N90-23067

Six–Degree–of–Freedom Aircraft Simulation with Mixed–Data Structure Using the Applied Dynamics Simulation Language, ADSIM

Clare Savaglio
Applied Dynamics International
Ann Arbor, Michigan

Abstract

This paper presents a realistic simulation of an aircraft in flight using the AD 100 digital computer. We discuss specifically the implementation of three model features: (1) a large aerodynamic data base (130,000 function values) which is evaluated using function interpolation to obtain the aerodynamic coefficients, (2) an option to trim the aircraft in longitudinal flight, and (3) a flight control system which includes a digital controller. Since the model includes a digital controller the simulation implements not only continuous time equations but also discrete time equations, thus the model has a mixed–data structure.

Introduction

Real–time simulation of a realistic model is a cost effective way to design and test hardware. Model simulation is faster and safer than testing the actual system. Obviously, the computer system used to simulate a model is an important factor when considering simulation speed and simplicity of implementation, but the simulation language is often equally important. The ease with which a system can be modelled is strongly dependent on the simulation language.

In this paper we present a realistic model using the System 100. We try to show how conveniently certain modelling problems can be handled using ADSIM. We discuss three major features of the aircraft model which are common to many simulations. The first feature is the evaluation of aerodynamic functions with a large data base. This type of situation occurs in simulations where an analytic function does not exist and where function files dependent on several independent variables are given to describe a model. The second feature, the longitudinal trim technique, is applicable to systems where useful simulations must be run at an equilibrium condition. Finally, the case of mixed–data systems occurs whenever a digital and a continuous system are simulated. Although z–transform theory can be used to estimate errors of a dynamic plant with a digital controller, it must be assumed that the plant is linear. If this is not the case, the plant must be approximated by a linear system. A thorough analysis of the error and system dynamics can be gained through simulation. Even if the plant is linear, oftentimes a knowledge of the inter–sample behavior is desired, and simulation of the continuous plant and the digital controller is again required.

In the next section a brief discussion of the System 100 is given. The following section gives an overview of the aircraft model; the reference systems used, the orientation method, the external force model, and the control system model are each described. The last sections are devoted to three specific features of the aircraft model, demonstrating how they are modelled using ADSIM.

The System 100

The System 100 is an integrated simulation environment which consists of a general purpose computer and a digital simulation computer. The general purpose computer is one of the DEC VAX systems. The VAX front–end computer serves as a host for the Applied Dynamics AD 100 and the ADSIM compiler. Program preparation is done on the host computer. The AD 100 is the digital simulation compute engine and consists of up to seven parallel processors. The AD 100 is a totally synchronous, bus-oriented, multiprocessor system capable of performing 20 million floating–point operations per second (20 MFlops).

The simulation language, ADSIM, is equation driven and block oriented. Many key elements of a typical simulation, such as integration techniques, function generation, and control system nonlinearity functions, are built into the language. A control executive consisting of two programs provides the basis for implementing a model. The two programs are INTEXEC which runs on the host computer and SIMEXEC which runs on the AD 100. The executive controls such parameters as simulation time, frame time, and integration step size. The user's ADSIM program, consisting of blocks of code, is inserted into the SIMEXEC code at compile time. In this paper we mention two types of ADSIM blocks, the **REGION** block and the **DYNAMIC** block. **REGION** blocks may contain procedural code. The number of times and the order in which these blocks are executed are dependent on the specific name of the block. For instance, the code of **REGION sync2** is executed before the simulation run is begun while that of **REGION sync4** is executed before each step of the simulation run. Eight optional **REGION** blocks are available. The **DYNAMIC** blocks contain the model dynamics. The code must be nonprocedural. The model differential equations reside here, written as a series of scalar and first order differential and difference equations.

ADSIM offers an interactive environment which allows the program to be modified without recompiling. This environment is named **INTERACT** and allows instantaneous changes to simulation elements at run time which include all numerical values of the program, integration algorithms, integration step size, end time, sample time of a digital system and the speedup ratio with respect to real time.

In this paragraph we note some ADSIM integration terminology since we will need the concepts to describe the implementation of the model. These terms are referred to as run-specs of the ADSIM program([6]). The frame time is the amount of time needed for the computer to solve the differential model equations. The step size, T, is the integration algorithm step used to calculate the system solution. The step time is the step size divided by the number of passes through the dynamical equations needed for the specific integration algorithm. For instance the fourth order Runge-Kutta method, (RK–4), requires four passes through the differential equations and thus the step time is equal to $\frac{T}{4}$. In order for the simulation to run at real–time the actual time it takes to integrate the equations should be equal to the simulated time, thus the step time should equal the frame time. As an example, suppose the model uses the RK–4 integration method, thus for real–time simulation the step size divided by four should be equal to the frame time. If the step time is larger than the frame time the program is running at faster than real–time.

Aircraft Model

The aircraft model is representative of a business jet. The model can represent many other types of aircraft simply by changing some of the simulation parameters. The jet aircraft is modelled as a rigid body, the center of mass chosen as the reference point so that the translational and rotational aspects of the motion can be analyzed separately. The state variables chosen to describe the translational motion are: total velocity relative to the atmosphere, angle of attack, sideslip angle, distance north, distance east and altitude ([1]). The rotational motion is described using the conventional aircraft Euler angles and the components of the aircraft angular velocity along body axes. Figure 1 shows a block diagram of the overall six-degree-of-freedom flight equations. Starting on the left are the state variables and ending on the right are the time rates of change of the state variables. The time rates of change are integrated over each integration step to obtain the state variables used for the next step.

A total of four reference systems is used in the model to conveniently describe the aircraft motion. The model orientation is best described using a Body reference system, a coordinate set fixed to the aircraft. A nonrotating flat-earth atmospheric reference is defined to enable the translations and rotations of the aircraft to be related to an absolute reference. The translational equations of motion are written using Flight Path axes since the total velocity, angle of attack, and sideslip angle are easily described in that frame. (The Flight Path frame is defined such that the total aircraft velocity is along the X axis.) The Stability frame, an intermediate axis set between the Body and the Flight Path frames, is used to describe the aerodynamic equations since force and moment data are usually obtained in that frame.

Only three angles are needed to describe every possible orientation of the rigid body model. Of the choices of well known methods to describe rigid body orientation, this aircraft model implements the method of quaternions. The method of Euler angles was not used since a singularity occurs when the aircraft is pitched ± 90 degrees. The method of direction cosines was not chosen since the method uses 9 parameters to define the model orientation, requiring 6 constraint equations. The method of quaternions only requires 4 parameters to describe the system orientation, and thus only one constraint equation is needed. ADSIM offers a convenient quaternion model, which, given the initial Euler angles (defined for a $3-2-1$ rotation), and the aircraft's present angular velocities, computes the present Euler angles and the direction cosines of the system. The external forces acting on the jet aircraft are modelled as a constant gravity force, aircraft thrust, and aerodynamic forces. Fourteen aerodynamic coefficients are used to describe the aerodynamics of the aircraft. The coefficients are dependent on the aircraft angle of attack, sideslip angle, deflections of the control surfaces, (aileron, elevator, rudder and flaps), and mach number.

The aircraft flight control system is described with a pitch, roll, and a yaw-damper control loop. Each control loop contains an actuator described by a proportional plus-rate feedback second-order system. In the case of the pitch and roll loops, a proportional plus rate feedback is subtracted from the autopilot angle input to drive the controller. In the case of the yaw-damper loop, the yaw rate is sent through a high-pass filter and subtracted from the autopilot rudder input to drive the system.

Aerodynamic Function Interpolation

Function evaluation is an important task for real-time simulation. In the aircraft model there is a large aerodynamic data base. Two variables, density and the reciprocal of the speed of sound, are dependent on only one variable, the aircraft altitude, and 14 aerodynamic coefficients, (such as the lift, side-force

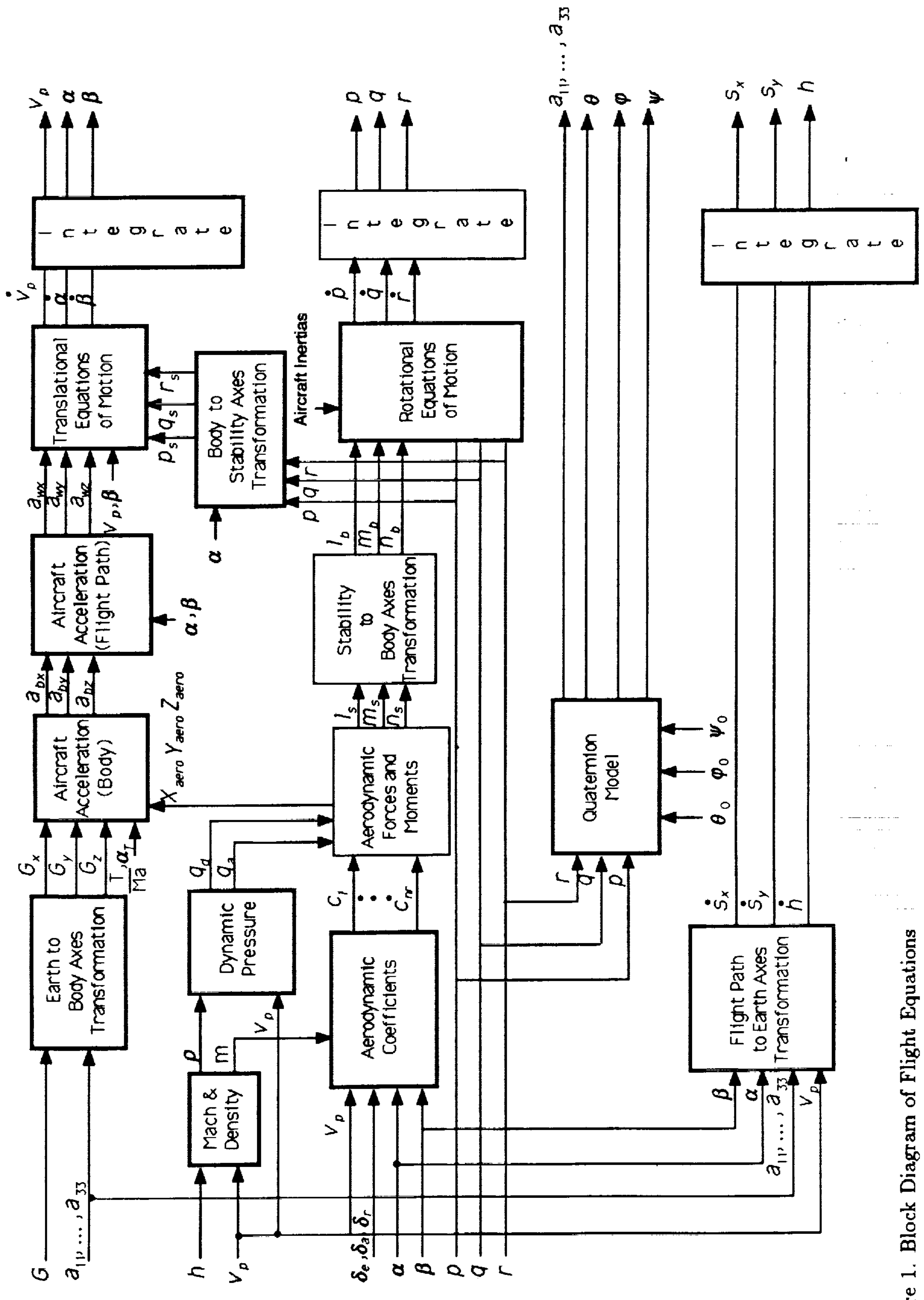

Figure 1. Block Diagram of Flight Equations

and drag coefficients), are dependent on up to four independent variables. ADSIM provides function generation capabilities for both equally and arbitrarily spaced functions on the AD 100. ADSIM allows multivariable functions with up to seven independent variables. Basically the way ADSIM evaluates a function of some independent values is to first use a binary search technique to determine between which given independent data points each input variable lies. Next interpolation techniques are used to evaluate the function at the given independent variable values. In order to perform function evaluation, independent variable and function tables must be declared. The aircraft simulation code containing function declarations for the density, and drag and lift coefficients is shown below ([4]).

```
(* Function generation breakpoint
 table definitions *)

INTERPOLATION_INTERVALS
            h_data(129 OF 129)
INTERPOLATION_INTERVALS
              a_data(33 OF 33)
INTERPOLATION_INTERVALS
              m_data(65 OF 65)
INTERPOLATION_INTERVALS
               de_data(3 OF 3)
INTERPOLATION_INTERVALS
               df_data(3 OF 3)

(* Function generation
              definitions *)

INTERPOLATION_FUNCTIONS
       air_density(h_data)
INTERPOLATION_FUNCTIONS
lift_coeff(a_data,m_data,
          de_data,df_data)
INTERPOLATION_FUNCTIONS
drag_coeff(a_data,
    m_data,de_data,df_data)
```

```
(* Assign data files to
  breakpoint tables and functions *)

FILES
 h_data   =    'h_sixdof.bpt',
 a_data   =    'a_sixdof.bpt',
 m_data   =    'm_sixdof.bpt',
 de_data  =    'de_sixdof.bpt',
 df_data  =    'df_sixdof.bpt'

FILES
 air_density = 'rho_sixdof.fun',
 lift_coeff  = 'liftc_sixdof.fun',
 drag_coeff  = 'cd_sixdof.fun'


(* The dynamic block containing the
continuous time equations *)

DYNAMIC Continuous
.
.
(* Evaluate air density *)

rho      =       air_density(h)
.
.
(* Aerodynamic coefficient
        evaluation *)

cl       =     lift_coeff(a,m,de,df)
cd       =     drag_coeff(a,m,de,df)
.
.
END DYNAMIC continuous
```

Longitudinal Trim

The model contains an option to trim the aircraft in longitudinal flight. The aircraft is assumed to be in symmetric level flight. Longitudinal trim is accomplished with the function, **NEWTON_RAP**, supplied by ADSIM. For a given flight regime, (aircraft altitude and speed), **NEWTON_RAP** determines the angle of attack, thrust magnitude, and elevator deflection to cause the longitudinal accelerations and pitch moment to be zeroed.

The function **NEWTON_RAP** uses the Newton Raphson method, a successive approximation process, to determine the trim values. This method has quadratic convergence. Although only a local method, the Newton Raphson algorithm works very well for the aircraft flight model, because the aircraft equations are well behaved and an inital guess for the trim values with some physical insight causes the method to converge within about four iterations.

The function **NEWTON_RAP** is called in the **REGION sync2** block of ADSIM, thus the equilibrium values can be determined before the simulation run and the aircraft control is initialized for a trimmed flight.

Simulation of Mixed–Data System

The simulation of both a continuous and a digital system presents special modelling problems, such as simulation of the computational delay times for the Analog to Digital (A/D) and Digital to Analog (D/A) conversions, and the simulation of sampling times. A special execute pair exists in ADSIM so that these modelling questions need not be re-solved for each model containing digital and continuous systems ([3]). The special execute pair named **EXECUTE mixed_data** controls the execution of the integration, the implementation of the delay times, and the order in which the dynamic systems are solved. Note that the continuous time equations should be solved on a frame-by-frame basis while the discrete time equations should be solved only every integer multiple of the frame time. This integer is dependent on the sampling rate of the digital system.

The execute pair for mixed–data system simulation is used in ADSIM programs along with two dynamic blocks. Simulations of this type are easily partitioned into two or more subsystems. One subsystem consists of algebraic and difference equations that represent the digital/discrete-time control laws. The other subsystem consists of the algebraic and differential equations that describe the behavior of the continuous time system. When creating the control structure for simulation of mixed–data systems, the following factors were considered:

- Numerical integration of discontinuities introduced by the digital controller residing in the continuous time subsystem.
- Selection of numerical integration method(s).
- Adjustment of either the sample period or integration step size.
- Proper padding of simulation frames in order to maintain real-time.
- Representation of computational delay in the digital subsystem.
- Representation of errors introduced by A/D and D/A conversions.

Let us discuss the selection of the integration method. Single-step predictor type methods which use the current and past derivatives to predict the new state values do so under the assumption of continuous derivatives. This assumption does not hold when simulating mixed-data systems. Strong discontinuites are introduced by the control command as determined by the discrete time system. Self-starting methods, such as the classical, multistep Runge-Kutta second, third and fourth order variety, do not rely on this assumption and thus these methods could be used.

However, the classical Runge-Kutta methods are not suited for real-time simulation since they require external inputs to the integration method before they become available in time. Applied Dynamics developed Runge-Kutta Real-Time methods which have similar accuracy and stability characteristics as the traditional versions but can be used for real-time simulations. Assume that we wish to solve the following vector differential equation:

$$\dot{x} = f(x, u(t)) \tag{1}$$

Where x is the state vector and u is the input vector. The following difference equation defines the second order Runge-Kutta Real-Time (RKRT2) integration method:

$$x_{n+1} = x_n + Tf(x_{n+\frac{1}{2}}, u_{n+\frac{1}{2}}) \tag{2}$$

where

$$x_{n+\frac{1}{2}} = x_n + \frac{T}{2} f(x_n, u_n) \tag{3}$$

We note from these equations that when using the RKRT2 integration method for the simulation of mixed data systems, we must assure the sample time is an integer multiple times the integration step size, T. The control structure of **EXECUTE mixed_data** will automatically adjust the sample time to insure this condition (only if the sample time is not an integer multiple of the frame time).

We now consider the simulation of the digital controller's computational delay. Actual microprocessor hardware does not produce control action to the continuous system instantaneously. The most common method is to compute the control effort and output as soon as the results become available. However, when simulating delay time, the lag must be an integer multiple of the continuous system integration step time. If an estimate of the computational delay for a particular piece of hardware is known, **EXECUTE mixed_data** will allow the effect of both fixed and time-varying computational delays to be simulated on the continuous system model. The error introduced by the limited accuracy of the A/D and D/A conversions can be modelled using a nonlinear control ADSIM function named **Quantizer** which generates a symmetric staircase function.

The general format for ADSIM programs using the control structure of **EXECUTE mixed_data** follows:

```
TITLE {Simulation Title}

DYNAMIC discrete

{ Difference and algebraic equations
     to represent digital controller.}

END DYNAMIC discrete

DYNAMIC continuous

{ Differential and algebraic equations
    to represent continuous-time plant.}

END DYNAMIC continuous

EXECUTE 'mixed_data'
```

We now describe the implementation of the digital controller in the aircraft model. In order to design the digital pitch control algorithm, knowledge of aircraft pitch behavior is needed. The longitudinal dynamics of aircraft are in general dominated by two phenomena; phugoid motion and short period pitching motion. Phugoid motion is a very slow and lightly damped oscillatory motion in the vertical plane induced by mismatch between lift and drag of the airframe for a particular flight condition. The short period pitching motion occurs when the aircraft pitch angle is not aligned with the flight path. The aircraft dynamics will drive the pitch angle

back toward the flight path. The short period pitching mode is much faster than the phugoid mode. The continuous system pitch dynamics were simulated and the frequency and damping ratio of the two motions were estimated from the transients. It was found that the phugoid motion exhibits a natural frequency of about 0.012 Hertz with a damping ratio of 0.1, while the short period pitching mode frequency is about 0.45 Hertz with a damping ratio of 0.35. The actuator is modelled by a proportional plus rate feedback second order system, its dynamics are characterized by a natural frequency of 5.0 Hertz and a damping ratio of 0.5. Using these estimates a transfer function representation was used to approximate the continuous system dynamics, where the input is the elevator deflection command and the output is the aircraft pitch angle.
A proportional-integral-derivative (PID) type of digital controller is implemented in the aircraft simulation. The digital controller transfer function, D(z), can be defined using z–transform notation and has the following form,

$$D(z) = \frac{a_1 + a_2 z^{-1} + a_3 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}. \qquad (4)$$

The controller constants, a_1, a_2, a_3, b_1, b_2, were chosen using the transfer function approximation of the continuous system. These constants are easily refined at run–time using **INTERACT**.
As discussed in a previous section, the ADSIM language supports both continuous time derivatives and discrete time difference equations. Therefore the D(z) transfer function is simply converted to ADSIM notation and placed into the discrete dynamic block. The continuous dynamic block contains the equations for the continuous model equations. The prime symbol denotes the differential operator, $\frac{d}{dt}$, and the pound symbol denotes a shift in time, or a next-state variable. We note that the transfer function for a zero order hold need not be represented in the ADSIM program since the simulation control structure of the **Execute mixed_data** pair supples this effect implicitly. The relevant ADSIM code is shown below.

```
TITLE 'Six Degree-of-Freedom
          Aircraft Simulation'

COMMENT

This is an ADSIM program for simulating an
aircraft in six degrees of freedom.
The program runs on the Applied Dynamics
AD 100. Sixdof performs all the computations
required for the six-degree-of-freedom
aircraft equations of motion.
...

END COMMENT

...

(* Equations of motion of the
                  aircraft*)

DYNAMIC Continuous

(* Flight control systems &
                   actuators: *)

(* Elevator actuator *)

dei      =   Deilim*SAT(deicom*
             Ideilim)
udec     =   Kde*(dei-des-Cdedot*
             dedot)
dedot'   =   Udelim*SAT(udec*
             Iudelim)
des'     =   dedot
de       =   des+detrim

(* Roll control system *)
```

```
daicom  =    Kphi*(phii-phi-
             Cp*p)
dai     =    Dailim*SAT(daicom*
             Idailim)

(* Aileron actuator *)

udac=Kda*(dai-das-Cdadot*dadot)
...
...
...
r'=Inv_1mIxzsqxInv_IxxIz*(mz+
   IxzxInv_Iz*mx)

END DYNAMIC Continuous

DYNAMIC Discrete

(* Pitch control system *)

e_theta  = -(thetai-theta
            -Cq*q)
f_theta# = e_theta
g_theta# = f_theta

deicom  =
B1*u_theta + B2*v_theta +
A1*e_theta + A2*f_theta +
A3*g_theta

u_theta# = deicom
v_theta# = u_theta

END DYNAMIC Discrete

EXECUTE 'mixed_Data'
```

Speed of Simulation

The time needed for a single pass through the highly nonlinear dynamic equations is 138.1 microseconds, corresponding to a frame rate of about 7000 frames per second. Since accurate real-time simulations of aircraft require about a 10 to 30 second frame rate([5]), it would be possible to run this model up to 700 times real time.

Conclusions

We have presented an ADSIM model for the real-time simulation of an aircraft. We have described the modelling of three specific features of the simulation which are common to many simulation models. These features are a large aerodynamic data base requiring function interpolation, an option to trim the aircraft, and a digital controller. We have shown the model can be implemented and executed conveniently using ADSIM.

References

[1] Fogarty, L.E., Howe, R.M., *Computer Mechanization of Six-Degree-of-Freedom Flight Equations,* Simulation 11, October 1968.

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

[3] Zammit, S., Zwaanenberg, K., *Simulation of Mixed-Data Systems Using the Applied Dynamics Simulation Language ADSIM,* European Simulation Multiconference, June 1989.

[4] *Six-Degree-of-Freedom Flight Simulation,* ADI Application Report, Ann Arbor, MI, 1988.

[5] *Six-Degree-of-Freedom Flight Simulation Using the System 10,* ADI Application Report, Ann Arbor, MI, 1983.

[6] *ADSIM Reference Manual (Version 6.1),* ADI, Ann Arbor, MI, 1989.