N94-14628

ASTEC and MODEL: Controls Software Development at Goddard Space Flight Center

John P. Downing , Goddard Space Flight Center
Frank H. Bauer, Goddard Space Flight Center
and
Jeffrey L. Surber, Fairchild Space Company

Abstract

The ASTEC (Analysis and Simulation Tools for Engineering Controls) software is under development at the Goddard Space Flight Center (GSFC). The design goal is to provide a wide selection of controls analysis tools at the personal computer level, as well as the capability to upload compute-intensive jobs to a mainframe or super computer. In the last three years the ASTEC (Analysis and Simulation Tools for Engineering Controls) software has been under development. ASTEC is meant to be an integrated collection of controls analysis tools for use at the desktop level. MODEL (Multi-Optimal Differential Equation Language) is a translator that converts programs written in the MODEL language to FORTRAN. An upgraded version of the MODEL program will be merged into ASTEC. MODEL has not been modified since 1981 and has not kept pace with changes in computers or user interface techniques. This paper describes the changes made to MODEL in order to make it useful in the 90's, and how it relates to ASTEC.

Introduction

Several programs have been developed at NASA's Goddard Space Flight Center (GSFC) in recent years. These include the Interactive Controls Analysis (INCA) program [1] starting in 1981, and the Windowed Observation of Relative Motion (WORM) program [2] starting in 1986. An important earlier effort is MODEL (Multi-Optimal Differential Equation Language) [3] developed in the 1960's and 1970's. In the last three years the ASTEC (Analysis and Simulation Tools for Engineering Controls) [4]software has been under development. ASTEC is planned to be an integrated collection of controls analysis tools for use at the desktop level. Planned conversions of INCA and WORM to PC/Macintosh programs will be part of the ASTEC system. MODEL is a translator that converts programs written in the MODEL language to FORTRAN. An upgraded version of the MODEL program will be merged into ASTEC. MODEL has not been modified since 1981 and has not kept pace with changes in computers or user interface techniques. This paper describes the changes made to MODEL in order to make it useful in the 90's, and how it relates to ASTEC.

ASTEC

ASTEC is being written to satisfy the requirements of the GSFC Guidance and Control

Branch. As such, it must run on the computer equipment used in the branch. Currently desktop units consist of PC's, Macintoshes, and an occasional Tektronix or X-Windows terminal. Mainframe capabilities consist of VAX 8830 and IBM RS6000 computers.

ASTEC is designed to meet the continuing needs of GSFC engineers, where high order and complex systems are the rule and not the exception, and where tried and true classical methods predominate. Because spacecraft repair is very expensive if not impossible, it is important that analysis methods be exhaustive rather than quick, and that algorithms contain no shortcuts which may compromise analysis results. There is also a high demand for a modern, friendly user interface, since many of the engineers use the Macintosh or Microsoft Windows environment.

It has long been planned to port INCA and WORM from the VAX/VMS to a desktop computer. Since PC's and Macintoshes predominate in our branch, they were chosen despite relatively poor performance in floating point operations. It is hoped that there will be performance improvements in the future and that some computations can be done on the VAX and the results downloaded later.

ASTEC Architecture

ASTEC will consist of several modules. Many of the currently implemented or planned modules are described below. The capabilities of ASTEC include classical control methods, simulation both linear and non-linear, multi-variable controls and matrix methods, and new experimental capabilities -- including dynamic locus and three dimensional frequency response. The following modules are under some state of development, and more may be added. It is hoped that by the time of this conference MODEL will be available from COSMIC in VAX, PC, and Macintosh versions, and that WORM will be available in a PC version. Note that the old VAX versions of WORM and MODEL have been available for some time.

ASTEC	Manage files and launch other jobs.
(PUEBLO?)	Block diagram editing (Prototyping Utilities Emphasis is Block LayOut).
MODEL	Build systems and launch analyses.
INCA	Transfer function and state space analysis.
WORM	Plot results.

Input to some of the ASTEC modules can come from either the user or, more importantly, from other modules. Thus, for example, as user could build a block diagram model of his system using the PUEBLO program. ASTEC could use this data to generated a simulation in the MODEL language, which could be translated, compiled, linked, and executed. The results could be plotted by the WORM package.

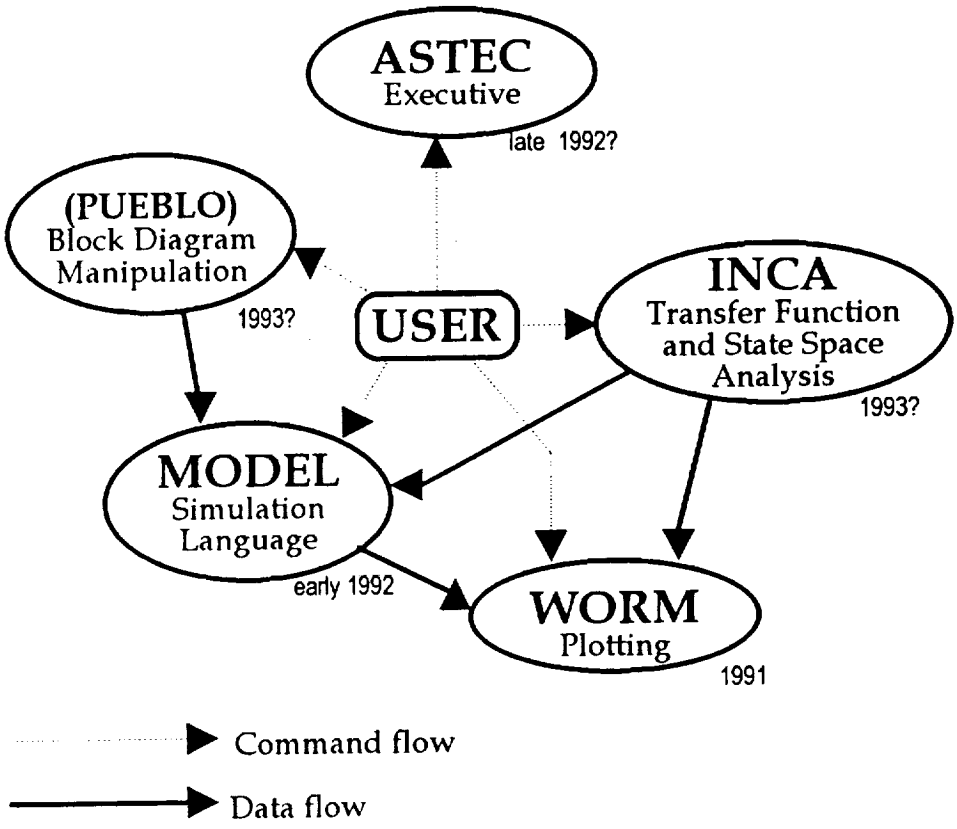


Figure 1. General ASTEC Architecture

While personal computers are quite good in the fields of graphics and user interface, they often fall short in the field of number crunching, especially if hardware floating point is not installed. For this reason a capability to transfer compute intensive jobs to a mainframe computer was deemed essential. Compute-intensive routines (such as MODEL) will be capable of dealing with text files only, allowing input and output data to be transferred between computers.

MODEL

The Multi-Optimal Differential Equation Language provides a means for generating numerical solutions to systems of differential equations using a digital computer. The notation of this language is similar to that used to describe physical systems by differential equations. Thus the learning process is simplified, programming becomes easier, and debugging is more readily accomplished. Programs written in the MODEL language are machine translated into FORTRAN-77 programs.

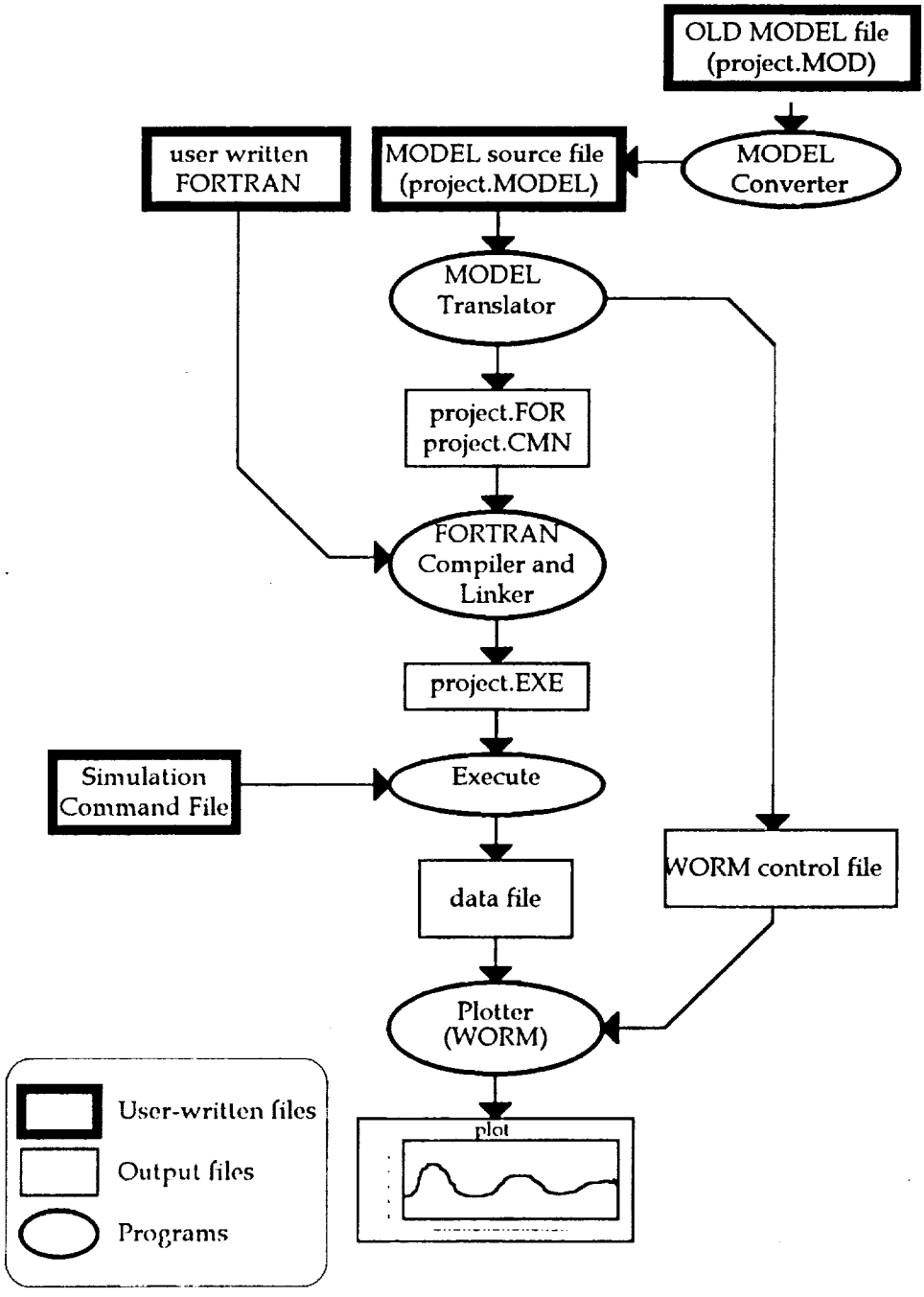


Figure 2. MODEL data flow diagram

MODEL is currently implemented on the VAX computer using VMS, on PC's under Microsoft Windows, and on the Macintosh. The VAX version is capable of automatically generating source files for the WORM plotting program. This feature will allow users to plot their data using the names assigned in MODEL.

Since the MODEL program is a translator, an additional translate step is added to the normal compile/link/run sequence. A data flow diagram is shown in Figure 2.

Language Features

A MODEL program is composed of Model statements. The basic MODEL statement is a differential equation. Equations can be entered in any order. The quotation mark (') is used to indicate a derivative, allowing the equations to be entered in a reasonably familiar way. Variables with quotation marks are derivatives of state variables. State variables can also be derivatives of other state variables. In this case multiple quotations marks are used.

Other MODEL statements include DEFINE statements to control the simulation, OPERATOR and FUNCTION statements to create an interface to user written subroutines, and comments to allow user documentation. MODEL uses a free-form line format. Multiple statements on one line are separated by semicolons (;). A statement may be continued to the next line by using ellipses (...). Two minus signs together indicate a comment--The rest of that line is ignored.

An simple MODEL program is the damped harmonic oscillator:

```
x"=z*x'+(k*x)
x(0)=10
x'(0)=0
z=-0.1
k =-0.2
WRITE (T,X",X',X)
DT=0.01; TFIN = 40.0 -- Time Step, Finish time
DEFINE FILE WRITE 0.1 ASCII TEST.OUT F14.6
```

Variables and Operators

MODEL variables must be one of seven types.

SCALARS : a single floating point number.
VECTORS : position, velocity, force, torque, magnetic fields, etc.
TENSORS : either a rotation matrix or Inertia tensors.

QUATERNIONS : are used to represent rotations.
CHARACTER STRINGS : are used to access external filenames.
MATRICES : an array of scalars, arranged in rows and columns.
SUBSCRIPT RANGES : used to create slices of matrices.

There is a simple relationship between the original program variables and the corresponding FORTRAN variables. MODEL variables are first truncated to 28 significant characters. If the variable is a state variable or derivative, an underline is appended, and then one or more 'P's to represent the order of the derivative.

x x
x' x_p
x" x_pp
x'(IC) x_pic
x''' x_ppp
t(0) t_ic

Variables may be manipulated by using operators. Operators may be unary or binary, and unary operators may be prefix or postfix. Each operator is given a priority. In complicated expressions the rules of precedence clarify the order in which operations are performed. Operations with equal precedence are performed from left to right. Expressions within parentheses are evaluated first and independently of preceding or succeeding operators. The operators in model are grouped in order of precedence, and are listed below. Note that certain operand types may be incompatible with certain operators.

DEGREES or RADIANS	Convert scalar to angle.
ARCMINS or ARCSECS	Convert scalar to angle.
.(period)	Vector element access.
^ or ** or ^- or **-	Exponentiation.
*	Multiplication.
/ or \	Division and left division.
DOT or *	Vector dot product.
CROSS or ><	Vector cross product.
+	Addition
-	Subtraction or negation.
||	Matrix column concatenation.
//	Matrix row concatenation operator.
==	Equivalence.
<> or != or ~= or |=	Non-equivalence
<	Less than.
>	Greater than.
<=	Less than or equal.
>=	Greater than or equal.
NOT or ~	Logical negation or inverse.

AND or &	Logical and.
OR or |	Logical or.
:(colon)	Matrix subscript ranging operator.
[]	Matrix indexing operator.

MODEL is equipped with built-in functions to support many function and non-linearities required for ease in simulation. Many of these will be familiar to users of FORTRAN, Pascal, or other programming languages. The basic trigonometric functions **SIN, COS,** and **TAN** are also available. These take an argument which MUST be of angle type, and return a scalar. Inverse trigonometric functions **ASIN, ACOS** and **ATAN** take a scalar argument and return an angle.

Other functions are used to represent various operations that are used in simulations. The **RANDOM** and **RANDOM12** generate random numbers. The **IF** function allows conditional assignment, like the C language ? operator. The **QUANT** function is used to implement quantization functions. The **LIMIT** function is used to implement limiters or limit functions. Additional functions such as **DEADZONE, BANGBANG, HYSTERESIS, BACKLASH** and **TRACKSTORE** are also available.

Other Features

The MODEL preprocessor is similar to the one in the C language. The #FOR statement is followed by a list of character strings. Each line after a #FOR statement is scanned for the at-sign (@), and if one is found, all at-signs in that line are replaced in turn by each character string. Lines without any at-signs are left alone. This process continues until a #ENDFOR statement is encountered. The INCLUDE statement is used to merge text from another file. Using the SYNTAX commands, the user can use his own routines in FORTRAN or other languages.

Run-time Command Language

The run-time command file is read by the generated simulation program to control the simulation. There are four types of statements in the run-time command language. The details of using a command file are implementation dependent. A simple command file is given below:

```
RESET
tfin = 60
RUN
PAUSE AT 30.0
k = 0.3
CONTINUE
STOP
```

The **RESET** statement returns all variables to their original values. The second line is a

variable change command. The format consists of a variable name and one or more values. The **RUN** command starts the simulation. The first line after the RUN command is a either the keyword **STOP** or a **PAUSE AT** command. If it is a PAUSE AT [time] command, it is followed a list of variable change commands to be given at that time. This allows the user to change parameters in the middle of the simulation.

Changes from first version of MODEL

For those familiar with the initial version of MODEL developed by Benjamin Zimmerman, the following describes changes made in the new version. These include:

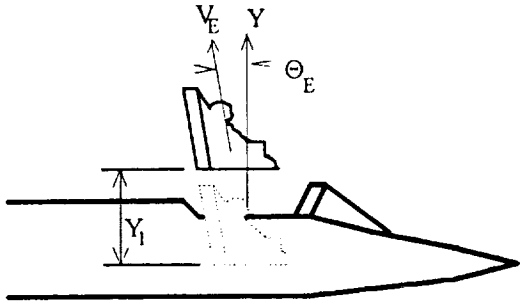
User defined functions and subroutines are now available.
Certain obscure relational operator definitions have been dropped. These are .EQ., .LT., .GT., .LE., .GE., .NE., */, /*, =/, =<, /=, =>, *=, and ><.
The IF statement has been changed to a function.
Elimination of conditional output.
The double comma (,,) may no longer be used to separate statements. Use the semicolon (;) instead.
New data types vector, tensor, quaternion character strings, matrices and subscript ranges. The old type is now called a scalar, and is now the default.
New mathematical operators have been added. These are ^, **-, ^-, \, MOD, DOT, CROSS, ||, //, ~=, !=, |=, ~, |, :, [] and others. Note that the colon has changed meaning from exponentiation to matrix ranging.
Multiple PLOT statements.
Automatic generation of WORM source files.
Several commands and that used to be abbreviated are now spelled out in full.
The END statement is no longer required.
The TAB statement and sampled variables not supported in the initial release.

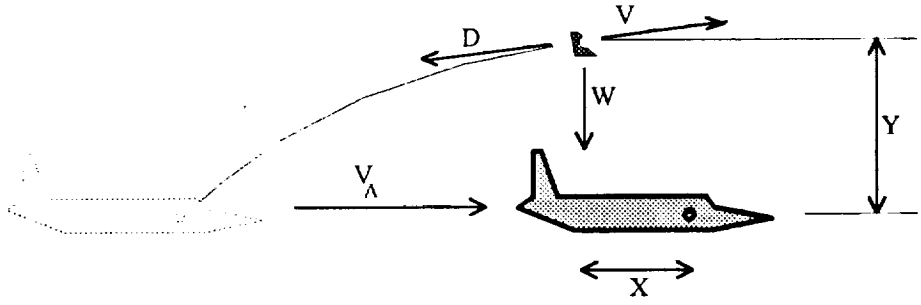
Example: Pilot Ejection Study

This study has been used as a standard of comparison for continuous simulation languages. This example is taken almost verbatim from the manual for the original Model program.

The purpose of this investigation is to determine the trajectory of a pilot ejected from a fighter aircraft and thus to ascertain whether he will strike the vertical stabilizer of the aircraft. Several combinations of aircraft speed and altitude are investigated since the drag on the pilot, which causes his relative horizontal motion with respect to the aircraft, is a function of both air density and velocity. The ejection system is so devised the pilot and his seat to travel along rails at a specified velocity V_E, at angle Q_E backward from vertical. The seat becomes disengaged from the rails at $Y = Y_1$.

Once the pilot and seat combination leaves the rails, it follows a ballistic trajectory which can be determined; however, since it is the relative motion of the pilot with respect to the aircraft (which is assumed to fly level with constant speed) that is important, we can formulate the equations so as to obtain this motion directly.





The governing equations are:

$X' = V \cos \Theta - V_A$
$Y' = V \sin \Theta$
$V' = 0$ $\quad 0 \le Y \le Y_1$
$V' = -D / m - g \sin \Theta$ $\quad Y > Y_1$
$Q' = 0$ $\quad 0 \le Y \le Y_1$
$Q' = -(g \sin \Theta) / V$ $\quad Y > Y_1$
$D' = \frac{1}{2} r C_D S V^2$

Constants (for all cases)

$m = 7$ slugs
$g = 32.2$ ft/sec^2
$C_D = 1$
$S = 10$ ft^2
$Y1 = 4$ ft

VE = 40 ft/sec
QE = 15

The initial values of V and Q (pilot's initial velocity vector at moment of leaving cockpit rails) are given by

$$V_0 = [(V_A - V_E \sin \Theta_E)^2 + (V_E \cos \Theta_E)^2]^{1/2}$$
$$Q_0 = \tan^{-1} [(V_E \cos \Theta_E) / (V_A - V_E \sin \Theta_E)]$$

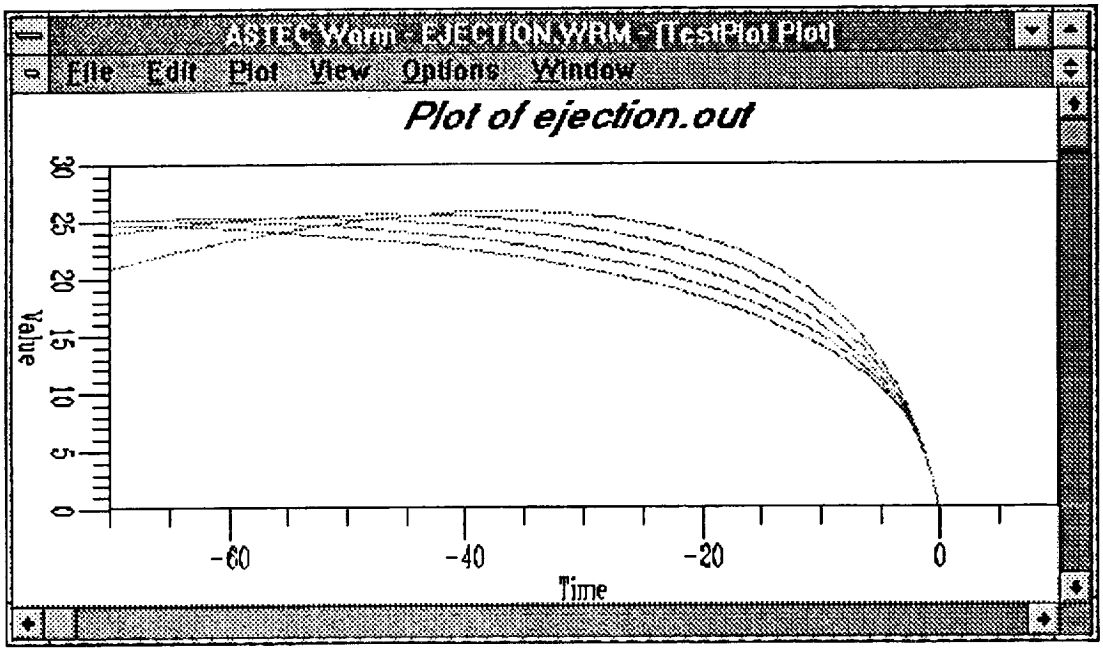
and further

$$X_0 = Y_0 = 0$$

The following quantities are to be printed every 0.002 seconds:

t, V, V', Θ, X, Y

```
------------------------------------------------
----------  PILOT EJECTION STUDY  ----------
------------------------------------------------
-----  EQUATIONS
x' = v*cos(th)-va
y' = v*sin(th)
clear = y>yl|y'<0
v' = IF(clear,-d/m-g*sin(th),0)
ANGLE th' = IF(clear,(-g*cos(th)/v),0) RADIANS
d = .5*rho*cd*s*v^2
v(IC) = sqrt((va-ve*sin(thd))^2 + (ve*cos(thd))^2)
ANGLE th(IC) = atan(ve*cos(thd)/(va-ve*sin(thd)))
x(IC) = 0; y(IC) = 0
-----  CONSTANTS
m  = 7     -- slugs
g  = 32.2  -- ft/sec^2
cd = 1
s  = 10    -- ft^2
yl = 4     -- ft
ve = 40    -- ft/sec
ANGLE thd = 15 DEGREES
-----  DATA
va = 900     -- ft/sec
rho = 2.3769e-3  -- slugs/ft^3
-----  OUTPUT
DEFINE FILE WRITE ASCII 0.02 EJECTION.OUT 6F12.5
WRITE (t,v,v',th,x,y)
-----  MODEL PARAMETERS
DT   = 0.002; TFIN = 2.0
```



Conclusion

The new MODEL programs is an attempt to take a sixties-vintage program and updata it for the nineties. When integrated into the other modules it of ASTEC, it should prove to be an extremely useful design tool. As a standalone program, it contains some features available in no other package currently available. It will soon be submitted to COSMIC for publication.

References

1. Bauer, F. H. and Downing, J. P., Interactive Controls Analysis (INCA) User's Manuals (4 Volumes), Program Number GSC-12998, COSMIC, University of Georgia, Athens, Georgia, 1985. Updated 1988.
2. Bauer, F. H. and Downing, J. P., Windowed Observation of Relative Motion (WORM) User's Manuals (2 Volumes), Program Number GSC-13232, COSMIC, University of Georgia, Athens, Georgia, 1988.
3. Zimmerman, B. G., Multi-Optimal Differential Equation Language (MODEL) User's Manual, Program Number GSC-12830, COSMIC, University of Georgia, Athens, Georgia, 1980.
4. Downing, J. P., Bauer, F. H., and Thorpe, C. J., ASTEC -- Controls Analysis for Personal Computers, Proceeding of the Third Annual Conference on Aerospace Computational Control, Oxnard, California, pp. 600-605, 1989.