https://ntrs.nasa.gov/search.jsp?R=19900013736 2018-02-23T21:18:38+00:00Z

N90-23052

ASTEC -- CONTROLS ANALYSIS FOR PERSONAL COMPUTERS

John P. Downing, NASA/Goddard Space Flight Center

Frank H. Bauer, NASA/Goddard Space Flight Center

Christopher J. Thorpe, Fairchild Space Company

Abstract

The ASTEC (Analysis and Simulation Tools for Engineering Controls) software is under development at 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 supercomputer. The project is a follow-on to the INCA (INteractive Controls Analysis) [1] program that has been developed at GSFC over the past five years. While ASTEC makes use of the algorithms and expertise developed for the INCA program, the user interface has been redesigned to take advantage of the capabilities of the personal computer. This paper describes the design philosophy and the current capabilities of the ASTEC software.

1 *Introduction*

The ASTEC software is being developed as a comprehensive controls tool for the 1990s. An attempt is being made to provide a broad framework upon which many different techniques can be added. ASTEC will eventually consist of over a dozen separate programs, most of which can have more than one copy running at a time. This paper will first summarize the recent history of controls software development at GSFC, and then describe the design philosophy and current capabilities of ASTEC.

2 *LSAP, INCA, and ASTEC*

The LSAP (Linear Systems Analysis Program) was developed by D. J. Duven at Iowa State University, and enhanced by T. P. Weis, C. J. Herget et al. at Lawrence Livermore Laboratory [2]. It was received at Goddard in 1981 and was perceived as a useful engineering tool. It was soon ported to the VAX/VMS computer and a series of enhancements was begun. Eventually the program was entirely rewritten and expanded. It was rechristened INCA, and published in 1985 through COSMIC. A greatly enhanced version was released in early 1988. The program has been reasonably popular being used at approximately 40 government and industrial sites. Several operational spacecraft have been developed or analyzed using the INCA program, the best example the Advanced Tiros-N (NOAA F-G-H) series spacecraft [3]. Most recently, state methods have been added to INCA. The routines used have been drawn from the SAMSAN library [4].

However, a number of limitations in the software have become apparent, and various improvements have been proposed both at Goddard and from outside sources. These include:

- Expansion of the multi-variable and state space capability.
- Simulation capablility, both linear and non-linear.
- Speeding the production of plots by improving terminal drivers, and adding more drivers for the ever-increasing variety of terminals.
- Conversion from VAX/VMS to other computers, including Unix, IBM-PCs, Macintosh, and Sun/Apollo workstations.
- A block diagram user interface.

While certain improvements (first three) are feasible, others would necessitate a complete reworking of the program. Any extensive changes would require modification to the user interface and command structure, and thus cause relearning and compatibility problems for many existing users. For these reasons it was decided to consider INCA a "mature" program. Only additions would be made and the basic structure would not be changed.

A new software system was desired for the needs of the future. An attempt has been be made to remove as many of the existing restrictions as possible. The following goals were established:

- Maintain most of INCA's capabilities.
- The main user interface would be based on block diagram construction and manipulation. A high level of "user-friendliness" is desired throughout.
- For enhanced graphics capability and speed, the main program should be at personal computer / workstation level. Use of the program would not then be restricted to the often unavailable or overloaded VAX computers.
- Develop a capability to upload compute-intensive jobs to other (faster) computers.
- Develop a capability to generate compilable simulations, similar to the ACSL [5] or MODEL [6] programs.
- User expandability.
- Capability to generate viewgraphs, both of results and system block diagrams.

3 Overall Design

We shall now describe the environment chosen for the initial development of ASTEC. The main program is to run on a IBM-PC or compatible under Microsoft Windows. Uploadable programs will be written in Ada for maximum portability. Versions of these Ada programs will also exist on the PC. ASTEC will consist of several independent program modules instead of a large monolithic design as in INCA. This design will keep the complexity at a more moderate level. The main development tool will be the block diagram manipulator, which will create "jobs" to be executed by other program modules in the background. The results of these jobs may be examined by display and analysis plotting programs.

3.1 Hardware Architecture and Development Configuration

ASTEC was conceived as a platform for many types of analyses. Microsoft Windows was chosen as the baseline architecture. This operating environment has many advantages as listed below.

- Near universal availability of the IBM-PC and compatible systems. While there are additional requirements for Windows (Hard Disk, Hercules or EGA display, and mouse) these are of moderate cost.
- Supports Pascal, the language in which INCA was written.
- A friendly graphical user interface.
- Device independence, in that display, printer and mouse drivers are already written. A large portion of the INCA development effort was spent on device drivers.
- Wide popularity, ensuring device driver support for future display and printer enhancements.
- Easy port to OS/2, possibly one that can be automated.
- Possible port to Macintosh, though not so easy.
- Possible port to PM/X for Unix systems.
- Easy interchange of data with commercial Windows programs for improved generation of data and reports.

Problems with the Microsoft Windows environment include:

- Often is slow, especially on and 8088 based computer.
- Complicated programming environment.
- Possible successful legal action by Apple Computer.

The Ada language was selected for use in the compute-intensive routines. Since these routines must run on a wide variety of computers, it was felt that Ada gave the best combination of portability and modern software capability.

3.2 ASTEC Architecture

ASTEC will consist of several modules. Many of the currently implemented or planned modules are described below.

• Manager	(ASTEC) Manage files and launch other jobs.
• Modeler	(MODEL) Build systems and launch analyses.
• Control Panel	(PANEL) Control system parameters.
• Template Librarian	(LIBRN) Allow user design of building blocks.
• Job Scheduler	(SCHED) Schedule executable jobs (JOBxx).
• State Space	(STATE) Execute operations using state space methods.
• Plotter	(GRAPH) Plot results of JOBxx modules.
• Root Locus	(JOBRL) Execute root locus analysis.
• Contour Locus	(JOBRC) Execute root contour locus analysis.
• Dynamic Locus	(DYLOC) New capability to display root locus as poles and zeroes are adjusted in real time.

- Frequency Response (JOBFR) Execute frequency response analysis.
- Freq. Resp. GH* (JOBFS) Execute GH-star analysis.
- Negative Real Axis (JOBNR) Execute a negative real axis analysis.
- Bode-Siggy (JOBSG) Execute a Bode-Siggy analysis.
- Describing Function (JOBDF) Execute describing function analysis.
- Time Response (JOBTR) Execute time response analysis.
- Linear Simulation (JOBLS) Compile, Link, and Execute a linear simulation.
- Nonlinear Simulation (JOBNS) Compile, Link, and Execute a non-linear simulation.
- Mode Significance (JOBMS) Execute a mode significance analysis.

The user will interact primarily with Block Diagram Manipulation (MODEL) Module. In many ways this module is the heart of ASTEC, and it was one of the first developed. The capabilities of ASTEC include classical control methods, simulation both linear and non-linear, multivariable controls and matrix methods, and new experimental capabilities -- including dynamic locus and three dimensional frequency response.

3.3 Uploading of problems to Mainframes and Super-Computers.

While personal computers are quite good in the fields of graphics and interactiveness, they often fall short in the field of number crunching, especially if hardware floating point (a co-processor) is not installed. For this reason a capability to transfer compute intensive jobs away from the PC was deemed essential. The number crunching routines such as simulation and root locus evaluation were to be written as simple batch-oriented file processing programs, without any links to the Windows environment. These routines would read a text (ASCII) file containing the problem, solve the problem, and write out the results to another text file. The only other output would be periodic progress reports on the state of the computation. Such reports would be useful if the job was executing on the PC, or if the user wished to check on the status of the job on the remote computer.

As much as possible, it was desirable to have only one version of the source code for these job programs. Thus the code is written in or will be converted to the Ada language. Preliminary versions were written in Pascal for testing purposes.

A disadvantage of this technique is the necessity to transfer data between the PC and the remote computer. Since numeric formats vary between machines, it was decided that all transferable data should be in ASCII format only. A second disadvantage was the requirement for a job scheduling program that could interact with the remote computer in a generalizable way.

3.4 Downloading of results for convenient analyses.

Once the remote job has been completed, the results must be retrieved and displayed. The result files must be transferred to the PC, and a plotting program started. These operations are ideally under the control of the scheduler module, which would poll the remote computer to see if a job were completed. Upon completion it transfers the file and notifies the user, and possibly even starts a plotting-analysis program.

4 Current status

ASTEC is currently being developed on two systems, MS-DOS/Windows and the Macintosh. More progress has been made in the Windows version. A progress report of the latest developments is summarized here. The status is reported circa June 1989.

4.1 Microsoft Windows environment.

Development using Microsoft Windows is simultaneously most frustrating and most rewarding. While the Pascal language gives the compile-time checking needed in a large project of this sort, the Windows development tools are designed for the C language, in which much of the checking is the responsibility of the programmer. Thus a lot of time was spent chasing runtime problems that should have been caught in the compile and link steps. There are also a number of outright bugs and omissions in the development tools. Better development tools are certainly needed. A "prelink" program was developed at GSFC to automate the generation of the various linker definition files.

The advantages of Windows programming make the frustrations worthwhile. It is almost trivial to create menus, and only slightly less so to create dialog boxes and "read" the mouse buttons. Graphics generation is intricate, but no more so than many other graphics toolboxes. And the fact that only one print routine is sufficient for a vast variety of printers is absolutely wonderful.

4.2 Macintosh environment.

The Macintosh development of ASTEC, as was previously mentioned, is somewhat behind that of the Microsoft Windows development. The Macintosh Programmer's Workshop (MPW) was chosen as the programming environment. The Programmer's Workshop is versatile, and it is primarily intended for the creation of stand alone applications. The Pascal used with the workshop is run as a tool, that is part of the MPW's shell and thus frees the programmer from having to initialize Mac Toolbox managers, and menus or events are preformed by the MPW shell.

5 Future plans

In the next year the remaining capabilities of the INCA program will be added to ASTEC. At that point the software will be submitted to COSMIC for publication and distribution. In the following year work will proceed on the simulation module. This is anticipated to be the most involved portion, involving new routines (i.e., not from INCA) and a new language (Ada).

6 Conclusion

The ASTEC system shows that personal computers can be used to perform sophisticated controls analyses. The use of user interface techniques pioneered in the business word (word processors and spreadsheets) can be used to make the life of the controls engineer easier as well. Also, there is no need to give up the power available in the mainframe environment. The algorithms used in ASTEC are equivalent to those used on mainframes, and are only slightly inconvenienced by the somewhat limited capacities of today's personal computers. When completed, ASTEC is a tool that will serve engineers in the 90's.

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

2. Herget, C. J. and Weis, T. P., Linear Systems Analysis Program User's Manual, Lawrence Livermore Laboratory, UCID-30184, Developed under contract W-7405-Eng-48.

3. Bauer, F. H. and Downing, J. P., Control System Design and Analysis using the INteractive Controls Analysis (INCA) program.

4. Frisch, H. P. and Bauer, F. H., Modern Numerical Methods for Classical Sampled Systems Analysis (SAMSAN Version 2) User's Guide, Program Number GSC-12827, COSMIC, University of Georgia, Athens, Georgia, 1984.

5. Advanced Continuous Simulation Language (ACSL) Reference Manual, Mitchell and Gauthier Associates, Concord, Mass, 1986

6. Zimmerman, B. G., Multi-Optimal Differential Equation Language (MODEL) User's Manual, Program Number GSC-12830, COSMIC, University of Georgia, Athens, Georgia, 1980.