SIMULATION IN EDUCATION

Keywords: *combined continuous/discrete simulation languages, differential equations, integration, numerical methods, random numbers*

GASP

by

François E. Cellier
Institute for Automatic Control
Swiss Federal Institute of Technology (ETH)
CH-8092 Zurich, Switzerland

and

A. Alan B. Pritsker
Purdue University and *Pritsker & Associates, Inc.*
West Lafayette, Indiana 47907

FRANÇOIS E. CELLIER received his PhD from the Swiss Federal Institute of Technology in 1979 with a thesis on combined continuous and discrete simulation languages. Since 1978 he has been a lecturer in the Department of Electrical Engineering of his alma mater. His main interests are in modeling and simulation methodology, and he has worked successfully as a "tool-maker" in the design of the GASP-V simulation package and—more recently—the COSY simulation language, a front end to GASP-V. He is the author of some 15 published articles and has received many invitations to hold seminar lectures, teach courses, and present survey papers at universities in Germany, Hungary, and the United States. He has also served on several international standardization committees for simulation software. Dr. Cellier is a member of IMACS and SCS.

A. ALAN B. PRITSKER is currently a professor of industrial engineering at Purdue University and president of Pritsker & Associates, Inc., a consulting firm specializing in simulation applications. He has held professorships at Arizona State University and Virginia Polytechnic Institute and State University, and was also director of the Systems Analysis Group at Battelle Memorial Institute. His educational background includes a PhD degree from Ohio State University in 1961. Dr. Pritsker is a member of AIEE, ASEE, ORSA, SCS, Sigma Xi, SIGSIM, TIMS, and the World Simulation Organization.

OVERVIEW

GASP is a digital simulation language that aids in the development of a much broader spectrum of dynamic models than most other languages can accommodate. GASP allows the variables that define system performance to be described by both equations and logical conditions. System descriptions written in terms of difference or differential equations are referred to as *continuous* models; system descriptions written in terms of discrete changes based on logical conditions are referred to as *discrete* models. GASP can handle continuous, discrete, and combined continuous and discrete modeling.

GASP, like most continuous simulation languages, includes numerical integration software for the solution of sets of simultaneous differential equations. It requires the user to write one line of FORTRAN code for each differential equation. In addition, GASP allows the user to write difference equations which can be independent of or related to the differential equations. Supporting routines are then included to compute the values of the system variables from the difference equations. The step size for the solutions is variable to allow for updating of status at frequent intervals when necessary and at long intervals during periods in which changes are slow.

The philosophy underlying GASP is similar to that employed by most model developers. Over the years, the common elements of simulation programs have been isolated, catalogued, and characterized. These common elements, written as FORTRAN or PL/I subprograms, make up the GASP simulation language. A student using GASP builds a model by employing GASP routines to perform common functions. To understand and use GASP, one must understand how to interface user programs with GASP functions.

The main tasks in teaching GASP are presenting the organizational structure, the syntax and semantics of the GASP routines, and the interface between GASP routines and user-written subprograms. These tasks are described below for continuous modeling.

THE USE OF *GASP* IN THE CLASSROOM

For purely continuous models, GASP uses the state-space representation:

$$\dot{\underline{x}} = \underline{f}(\underline{x}, t) \quad ; \quad \underline{x}(\underline{t}_0) = \underline{x}_0$$

Assuming that the student has had a course in differential equations, it requires about two hours to introduce the concept of state variables and to show the student how higher-order differential equations can be transformed into sets of first-order differential equations. The coding of a state-space model in GASP is straightforward, and it takes not more than an hour to introduce the GASP variables if the student is already familiar with FORTRAN IV.

BRIEF DESCRIPTION OF *GASP* SYNTAX (FORTRAN VERSION).

The GASP variable SS(I) is used to represent state variable I. The derivative of state variable I is defined by the GASP variable DD(I), that is

$$\text{DD(I)} = d\text{SS(I)}/dt$$

The immediately preceding values of the state variables and their derivatives are maintained as the GASP variables SSL(I) and DDL(I). With these GASP variables, a difference equation for state variable I would be written in subroutine STATE as

```
SS(I) = SSL(I) + DTNOW * RATE(I)
```

where DTNOW is the step size and RATE(I) is the rate of change of state variable I during DTNOW. This equation states that the value of state variable I at time TNOW is equal to the last value of state variable I plus the amount of change that would occur during time DTNOW. RATE(I) could be a function of other system variables. The executive function calls subroutine STATE every time a step is to be made so that the updating of time is implicit in the equation for SS(I).

To define a state variable by a differential equation, the GASP user would write a defining equation in terms of DD(J). For example, if the differential equation for state variable j is given by

$$dy_j/dt = Ay_j + B$$

the corresponding GASP statement is

```
DD(J) = A * SS(J) + B
```

where SS(J) represents y_j.

In this situation, subroutine GASP calls subroutine STATE many times within a step in order to obtain estimates of the derivatives DD(J) within the step. These estimates are used to compute SS(J) at TNOW from the equation

$$\text{SS(J)} = \text{SSL(J)} + \int_{\text{TTLAS}}^{\text{TNOW}} \text{DD(J)}dt$$

where TTLAS is the time at the beginning of the step and corresponds to the time at which SSL(J) was computed. In GASP IV, the integration method is a Runge-Kutta-England numerical algorithm which maintains a user-prescribed single-step accuracy. If the accuracy is not maintained, the step size is reduced and the integral is recalculated. This process is repeated if necessary until a user-specified minimum step size DTMIN is reached.

In GASP V, users can provide their own integration algorithms or select one of the following: Runge-Kutta-Fehlberg (5th and 8th order), Runge-Kutta-Simpson (4th order), Gear-Kahaner (variable order), Adams (variable order), Impex-2, Euler, or Fowler-Warten. In addition, subprograms for solving partial differential equations by the method of lines are available, and subprograms for logic, memory, and generator functions are included. Examples of logic functions in GASP V are input switches, flip-flops, and gates. Memory functions included are hysteresis and delays. Generator functions include step, ramp, and impulse functions. In contrast to purely continuous simulation software, GASP V resolves all discontinuities internally by means of event handling.

Continuous models become complex when parameter values (or the structure of the model) change at threshold values determined by logical conditions. GASP provides the user with mechanisms to express the conditions and the discrete changes. Another two hours of instruction (after the initial two hours) are required to introduce students to these concepts. Elements that are difficult to model with continuous simulation software (such as the looseness of a gear) are easily modeled with GASP since it permits discontinuities in state variables. In GASP, state conditions and threshold values may be a function of system variables. When a state variable crosses a threshold value, a *state event* is said to occur. Discrete system changes and modeling changes can be made at state-event times. In GASP, discrete changes in system variables are modeled through the process of event scheduling.

Several reasons for choosing GASP for a continuous simulation course may be mentioned:

(1) The simulation course should prepare the student for real-life applications. Since most so-called "continuous" engineering models are models of systems that are really discontinuous, a continuous simulation course should describe combined continuous/discrete simulation. GASP is the only language that does this and is well-documented and widely distributed.

(2) After completing their studies, students are expected to apply their knowledge on the job. Although the difficulties involved in learning a new simulation language are not insurmountable,

most engineers are reluctant to expend the effort. Since GASP is a subroutine package coded in ANSI FORTRAN IV, it is machine-independent and can be implemented on any computer with a FORTRAN compiler. Thus, students will either find that GASP is already available at their companies or can be ordered and implemented without severe problems.

(3) A university-level simulation course should do more than teach the utilization of a particular program. The student not only should learn how to model a system, but also should learn the mechanisms of a simulation program. For this latter task, the simulation program needs to be transparent. Because of its outstanding documentation,[2] GASP is as transparent as any home-tailored FORTRAN-IV-coded simulation program, but it also has the advantage of relieving the user of the difficult task of structuring the simulation program appropriately.

CHARACTERISTICS OF *GASP*

GASP is available at over 400 installations. GASP IV requires 26 000 words of core storage on the CDC 6500. The cost of a permanent license to use GASP IV is $300 for industry or government ($800 for GASP V) and $100 for academic institutions ($250 for GASP V). The computers on which GASP IV operates are AMDAHL V-370; Burroughs 5500, 6700; CDC 3200, 6400, 6500, 6600; CYBER 73, 74, 170, 175; Data General Eclipse Minicomputer; DEC System 10; General Automation SPC 16/65; Harris SLASH/4; HP 2100, 3000; Honeywell 600, 605, 635, 3200, 6000; IBM 360, 370; KL 1090; Lockheed SUE; SEL 32; TR 440; UNIVAC 70, 90, 1106, 1108, 1110, 8003; VAX 11/780; and XDS Sigma 7, Sigma 9.

REFERENCES

1 CELLIER, F.E. BLITZ, A.E.
GASP V: a Universal Simulation Package
Proceedings 8th IMACS Congress on Simulation of Systems North-Holland Publishing Company
Amsterdam 1976

2 PRITSKER, A.A.B.
The GASP IV Simulation Lanugage
Wiley New York 1974

3 PRITSKER, A.A.B. YOUNG, R.E.
Simulation with GASP-PL/1
Wiley New York 1975