MODELING, IDENTIFICATION AND CONTROL, 1986, VOL. 6, NO. 4, 217–229
doi:10.4173/mic.1985.4.3

Perspectives in software design for dynamic process simulation†

S. MURTHY DIVAKARUNI‡

Keywords: *Software design, process simulation, integration schemes, steady state solvers, linear analysis.*

The process simulation analysts have a wide range of options for simulation packages today. It is often difficult to choose one simulation software package to meet all the requirements of the process industry. The revolutions in computer hardware and software combined with increased knowledge of the users makes it a challenge to design a single simulation software package that meets all of their demands.

In this paper, the needs of the process simulation users and general features of the two most widely, but differently designed simulation packages ACSL and EASY5 are described. Based on the experiences of the Modular Modeling System (MMS) code users in the last three years, the shortcomings of the simulation languages in general are described. Additional desired features for these languages are briefly mentioned. The role of personal computers and engineering work stations in easing the burdens of the users for the code inputs/outputs design are outlined, and their advantages in the design of advanced and partitioned simulation languages are explained.

1. Introduction

Traditionally, the developers of the power plant design and analysis software always insisted in designing the numerical integration schemes themselves, as a part of the overall software development effort. This required the users to keep in pace with the latest developments in the numerical integration schemes and distracted their attention from the main efforts viz., the power plant analysis. The recent enhancements in general purpose simulation languages have earned their place and creditibility in the process simulation arena due to diligent effort by their designers to understand the needs of the users. Rapid education on the user's part created new demands from them. In addition to ever increasing competition among the commercial software vendor houses, the user demands posed new challenges to the software designers. The task of meeting the user demands has increased by another dimension due to the diversities seen in the hardware and migration towards smaller and powerful machines. In general, however, the use of general purpose simulation languages to perform the process simulation and analysis has increased significantly in the last decade.

In 1978, the Electric Power Research Institute (EPRI) made a deliberate choice to use the simulation languages in the development of the Modular Modeling System (MMS) code—the software package for the dynamic analysis of the fossil and nuclear power plants. Many simulation languages were screened to select the

Received 1 July 1985.

† This paper was presented at the International Seminar on Modern Methods in Dynamic Simulation of Industrial Processes, Trondheim, Norway, May 1985.

‡ Nuclear Power Division, Electric Power Research Institute, Palo Alto, California, 94303, USA.

most popular and the state-of-the-art software, based on the selection criteria that consisted of the perceived user needs. The Advanced Continuous Simulation Languages (ACSL) developed by Mitchell & Gauthier Associates, Inc. (MGA) and Engineering Analysis System (EASY5) developed by the Boeing Computer Services (BCS) were selected to form the framework for the MMS code development. MMS and the typical user applications with the code are discussed in Divakaruni (1985), where some design features of the process simulation software have also been presented. During the course of the last seven years, the MMS users have identified several shortcomings in these two most advanced simulation software packages and the code developers MGA and BCS have complied with their requests to a large extent, to alleviate the problems. Where it was necessary, EPRI worked with their contractors Babcock & Wilcox (B&W), BCS and Argonne National Laboratories to address the deficiencies in the code in an expeditious manner. The MMS code commercializer, B&W is further addressing the needs of the MMS user community especially in developing the user-friendly packages, and in improving the steady-state solvers and integration routines in the MMS simulation language.

The new applications for the power plant process simulation software MMS have given challenges to EPRI to make the code more versatile and fast running, which in turn required ACSL and EASY to be more efficient, faster, economical and portable. Some of these requirements are conflicting in objectives and required compromises in the modeling accuracies.

In the following section, the general and most desired features of simulation languages are provided and ACSL and EASY5 are used as examples.

2. General design features of simulation packages

A good simulation language provides four features which makes its utilization extremely attractive from both a programmer and a user standpoint: (1) translation, (2) input/output, (3) integration algorithms, and (4) linear analysis routines.

Translation

The simulation language serves as a pre-compiler to translate a user's source code into a FORTRAN program. This allows the source program to be extremely compact. It also allows the user and the programmer to take advantage of the functions and routines provided by the language like tables, limits, deadbands, random number generator, switches, etc. The translators convert the source program statements into an executable order providing great flexibility.

Input/output

The simulation language provides a run-time executive which includes many user convenience features for printing and plotting outputs either as print plots or line plots.

Integration algorithms

Simulation of a power plant generates a stiff mathematical model. That is, a model which includes a range of response modes (eigenvalues) whose characteristics

response times vary widely. The range in a typical power plant model includes modes with response times of 10^{-5} sec to modes with response times of 10^{5} sec. This severely complicates the numerical integration because while the user is not interested in the fastest response modes (less than a few tenths or hundredths of a second) the integration algorithm must at least provide stable results for those modes. Special integration algorithms have been developed to address models with these characteristics; the most popular is an algorithm developed by W. Gear (1971).

The simulation language includes this algorithm, or a variation, and thus provides an integration technique specifically tailored for models similar to power plant models.

Linear analysis routines

While transient response analysis is the major use of a plant model, it is not the only use. Particularly in the area of control system design and analysis, a great deal of the valuable information is available from analysis of the linear model. The linear model can be used to generate root-locus diagrams, Bode plots, Nyquist plots and perform, stability margins analysis, and optimal control design.

A clear advantage from the MMS code development standpoint was that by building the code in an existing simulation language, these four features were available without development costs to the project. Still another advantage to use of a simulation language was that (to the extent that the language is widely used), a built in user base already existed.

For these reasons EPRI decided to develop MMS in a simulation language framework. A number of simulation languages were evaluated. In addition to the four features discussed above, it was clear that the language must have at least the following three characteristics, (1) support, (2) transportability, and (3) MACRO capability.

Support

The language must be well supported by an organization thoroughly familiar with its operation. This support should include continuing development from which MMS could benefit.

Transportability

The language must be usable on several large mainframe computers, at least on large CYBER and IBM machines. It must be available for installation on other users machines with minimum transportability problems.

MACRO capability

The language must have MACRO capability to allow a FORTRAN program to be built as an in-line code and that allows MMS to generate all variable names.

3. ACSL vs. EASY5 comparisons and other simulation languages

Although ACSL and EASY5 were used to provide the features outlined above and to meet common objective of supporting the MMS user needs, the structures of

A	Operation	Languages ACSL	EASY5	DSNP	DSL	DARE-P	SALT	CSMP
	Simulation capabilities							
	Change and define model parameters	YES	YES	YES	YES	YES	YES	YES
	Change integration technique	YES	YES	YES	YES	YES	YES	YES
	Change integration controls	YES	YES	YES	YES	NO	YES	YES
	Save and recover operating points	YES	YES	YES	YES	YES	YES	YES
	Print output results with time	YES	YES	YES	YES	YES	YES	YES
	Plot output results with time	YES	YES	YES	YES	YES	YES	YES
	Cross plot output results	YES	YES	YES	YES	YES	YES	YES
	Scan range of output	YES	YES	YES	YES	YES	YES	YES
	Restart capability	YES	YES	NO	NO	YES	YES	YES
	Modify parameters with time	YES	YES	YES	NO	YES	YES	YES
	Define blocks of execution commands	YES	YES	YES	YES	YES	YES	YES
	FREEZE states during simulation	NO	YES	NO	NO	YES	YES	YES
B	Linear analysis capabilities							
	Linear model generation	YES	YES	NO	NO	NO	YES	YES
	Steady state finder	YES	YES	NO	YES	YES	YES	YES
	Eigenvalues/eigenvectors	YES	YES	NO	NO	NO	NO	NO
	FREEZE states for linear analysis	YES	YES	NO	NO	NO	YES	NO
	Stability margin analysis	NO	YES	NO	NO	NO	NO	NO
	Frequency response analysis	NO	YES	NO	NO	NO	NO	NO
	Bode plot	NO	YES	NO	NO	NO	NO	NO
	Nyquist plot	NO	YES	NO	NO	NO	NO	NO
	Nichols plot	NO	YES	NO	NO	NO	NO	NO
	Transfer functions	NO	YES	NO	NO	NO	NO	NO
	Root locus analysis	NO	YES	NO	NO	NO	NO	NO
	Optimal controller design (Optional reduced order)	NO	YES	NO	NO	NO	NO	NO
C	Validation	Good-Excellent	Excellent	Good	Poor	Poor	Poor	Good
	Component models availability	MMS	MMS	Yes	None	Some	Some	None
	Controls	40 comp.	70 comp.	Minimum	Minimum	Minimum	Minimum	20 comp.
D	No. of utility users (approx.)	~50	~15	NONE	2–5	2–5	NONE	~20
E	Total users (estimate)	>300	>200	30	10	~10	1–5	~200
F	Commercializer support	MGA	BCS	NONE (F&A)	NONE	University of Arizona	NONE (ANL)	NONE NOW (IBM)

Table 1. Simulation language comparison chart

the two packages are widely different. The major differences arise from (1) model structure, (2) nomenclatures, (3) model schematics and of course the capabilities.

ACSL is block oriented language and follows the Continuous System Simulation Language (CSSL) specifications. The model is housed in three blocks: INITIAL, DYNAMIC and TERMINAL. INITIAL block is used to define the variables prior to simulation; DYNAMIC block is the main simulation block and TERMINAL block is used to wrap up the simulation model. The model consists of free format FORTRAN-like statements and FORTRAN subroutine like invocations of any of 112 MACROS. Generally, these MACROS are MMS modules or FORTRAN equivalents. ACSL allows the user to define MACROS within the program. Thirteen MACRO directive statements provide branching logic and statement generation loops to allow the amount and type of code generated to be controlled by the MACRO directive statements. FORTRAN subroutines, if needed, may be appended to the program. ACSL capability allowing six characters for variable names has been exploited to define the MMS variables very effectively.

EASY5, on the other hand is component oriented language unlike ACSL. Although it does not follow CSSL specification, it is very widely used and is a simulation language with exceptional capabilities. The code is supported by a large group of numerical analysts at BCS on a continuous basis. The EASY5 model consists of definition of location in a schematic and interconnections for any of 48 standard EASY5 components or the MMS modules and imbedded FORTRAN statements. EASY5 has additional 72 standard components which are largely of control hardware. EASY5 provides eleven matrix operations, e.g., addition, multiplications, inversion etc. EASY5 variable names are seven characters in length, with blanks allowed. All EASY5 names are automatically generated by appending a three character engineering quantity identifier, a two character component name and a one or two character user defined component identifier. Further, EASY5 automatically generates a schematic representation of the model from user specified component locations showing all interconnections. EASY5 also allows the user to define MACROS within the program, which may be saved as part of a user-defined MACRO library for utilization in subsequent EASY5 models.

The other major difference between ACSL and EASY5 are in the simulation and linear analysis capabilities. Table 1 summarizes these languages comparisons. Additional languages are also included in the table and comparisons are provided based on author's knowledge.

Because MMS was developed in two simulation languages ACSL and EASY5, there were opportunities to compare the performances of the two languages in running a fairly difficult nuclear power plant transient. In Smith *et al.* (1983) the steps involved in executing a 840 second duration transient are elaborated and the CPU time required to generate the results were given. The EASY5 model outperformed the ACSL model in overcoming the integration difficulties and in reaching the steady state. In many instances and one in particular (Smith *et al.* 1983), running a natural circulation transient in the nuclear power plant, the ACSL steady state finder had difficulties in coping with the low pressure drops encountered in the primary loop during the natural circulation conditions. The transient pressure decreased from 2250 psi to the natural circulation pressure conditions in five minutes and remained essentially constant thereafter.

The major reason for any simulation language superiority lies in the robustness of the integration and steady state solvers. In this regard, EASY5 proved to be a better language, in handling these difficult power plant transient conditions.

4. Experiences with ACSL and EASY5

The willingness on the part of BCS to incorporate the MACRO capability eased the burden on EPRI's part to keep the MMS code architecture intact. Special purpose translator was developed to translate the ACSL modules into EASY5 modules at minimum cost and this allowed to perform code validation and verification activity in one language.

The model generation capability and advanced diagnostic aids to locate the implicit algebraic loops in EASY5 provided comfort to the MMS users in debugging the plant models.

BCS stiff integration routine is a modified gear algorithm unlike the ACSL algorithm and allowed larger time steps to be taken during a mild to severe transient. The variable-step variable-order basic and modified gear algorithms generally changed the integration order quite frequently during power plant transients and especially in larger models, it took extremely small time steps. Algorithms which could take advantage of the 'sparsity' and 'bandedness' of the Jacobian matrices would have been helpful in running large MMS models.

The steady state solver in the EASY5 package worked most of the time, although it required the user to acquire skills in correctly using the 'Freeze' option to isolate the effects of some MMS state variables to reach the steady state quickly. The 'TRIM' function in ACSL failed with large models and the 'FREEZE' function worked rarely and when used by some experienced code developers. The larger nuclear models generally had strong coupling between the component states. In a 90th order nuclear plant model, the Jacobian matrix has entries ranging from 10^6 to 10^{-11}, and various components can be identified and separated. The run time requirements in executing this type of large model were generally excessive. EPRI, its contractors, and the code commercializer Babcock and Wilcox and MGA are looking at ways to improve the steady state and integration routines in ACSL. The following paragraphs give descriptions of the work under progress in this area.

5. Enhancements update in ACSL and EASY5 for MMS applications

Steady state solvers

The two common methods employed for finding the steady state operating points for lumped parameter dynamic models are integration and gradient approaches. The first straight-forward method for obtaining a steady state operating plant is simply to integrate the differential equations until all states reach equilibrium conditions. This is an inefficient approach which can succeed in the absence of instabilities induced by the process model or numerical schemes. The other approach is a gradient search wherein model equations are numerically linearized about an initial guess $\bar{x}_i^0$ to find the Jacobian matrix A and subsequently determine a direction of search

$$\alpha\bar{x} = -A^{-1}\bar{F}_i$$

for a general set of equations

$$\frac{dx_i}{dt} = F_i(\bar{x}) \tag{1}$$

where

$$i = 1, \ldots n.$$

The improved guess to the desired operating point is obtained from

$$\bar{x} = \bar{x}^0 + \varepsilon\alpha\bar{x} \qquad 0 < \varepsilon < 1$$

The procedure is repeated until convergence.

The ACSL 'TRIM' function uses Newton–Raphson technique based on the second approach, although there are many improvements. In using TRIM function, MMS users found that the straightforward integration provided better initial guesses for subsequent application of the ACSL TRIM function.

EPRI together with Argonne National Laboratories reviewed four gradient methods from IMSL routines and applied them to two benchmark nuclear plant models. They were;

(1) Levenberg–Marquardt algorithm using steepest descent calculation,

(2) Quasi-Newton method approximating the Jacobian,

(3) Powell–Hybrid technique, and

(4) Secant method

The two MMS models used for comparison consisted of 42 and 91 non-linear differential equations. Both were characterized by real and highly damped complex pairs of eigenvalues which ranged from 10^{-5} or 10^{-6} to excess of 10^4. While both models integrated to steady state quite easily, only Levenberg–Marquardt model brought either model to steady state. The evaluation process was repeated with modified MMS models and after integrating the model with the gear integration for one second to allow the fast-dynamics to approach quasi-equilibrium. The evaluation results are given in the following Table 2 for the 42 state model.

Similar results were obtained with the 91 order model also. In case of TRIM function, the fraction of the linear step (FRACDL) was manually changed from 0·1 to 0·25, 0·5, 0·75 and 1·0 whereas the other routines were fully automatic in selecting the steps. While Marquardt model proved to be the most robust of all in solving the stiff non-linear equations to steady-state, secant and Powell seemed to have the best convergent properties.

Based on the above, Mitchell and Gauthier Associates, Inc. are modifying the current TRIM function in ACSL. While the TRIM function seems to work fairly well for linear systems to drive the unfrozen state variable derivatives to zero using the Newton–Raphson method, for some class of non-linear problems divergence results. MGA is planning to replace the TRIM operator with a better zero deriv-

Algorithm	Initial residual	Final residual	Required function evaluations
1. Integration to 1 sec.		1495	695
2. TRIM	1495	1·2	2077
3. Marquardt	1495	35	274
4. Quasi-Newton	1495	1483	Interrupted at 402
5. Powell	1495	8·2	58
6. Secant	1495	1·0	107
7. Integration to 200 seconds	—	3·3	1271

Table 2. TMI-1 42nd order model evaluation.

ative finder find that will transition to steepest descent steps in the case of difficulty, reverting back to Newton steps when the going gets easy again.

Automatic 'freezing' of singular state variables and selection of linear interpolation step FRACDL are other desired features in ACSL's TRIM function. EASY5 currently does these and enables validity error print-out when the steady state solution is not physically meaningful.

Integration routines

Babcock & Wilcox, as the MMS code commercializer, is attempting to strengthen the library of integration routines to meet the MMS users needs. The simulation studies related to power plants, involve quite often discontinuities in the process variables, special events that inject significant process disturbances in various magnitudes and require large models in the order of 200–300 differential equations, Jacobians of which could be either dense or sparse. In a recent BWR full plant model development (approximately 230 states) with the MMS code on cyber 176 machine, special memory segmentations and partitions of the model into separate derivative sections were required to be able to execute the model. In another interactive application with ACSL, the MMS model was used as a driver to the colour graphic displays depicting the plant status information. Since the integration was taking very small steps, the simulation was much slower than the expected real-time plant behaviour. Several scenarios with different operator actions were simulated and replayed. Interactive model execution in real-time would have been more useful to evaluate effects of various simulated operator procedural actions during the course of a transient. Sparse matrix routines, and options to reduce the model order, which could enable the model to execute faster would have helped.

B&W evaluated sparse matrix routines and the integration methods for stiff ODES in general, and incorporated them in STACSL—the ordinary differential equation solver for ACSL version of MMS. Features included in this are sparse matrix treatment of the Jacobian matrix, incorporation of techniques for detecting and processing special events and discontinuities, and diagnostic capabilities for developing and debugging the MMS models. In STACSL, the Jacobian matrix J is treated as a sparse matrix by default. This significantly reduces the storage requirements and allows larger systems to be solved. A special grouping technique also allows Jacobian matrix calculation with fewer derivative evaluations. STACSL uses the Yale sparse matrix package to perform the sparse matrix related calculation.

In the BCS EASY5 package, the Gear algorithm for solving stiff ODES is modified along the lines of Hindmarsh solution approach. This BCS Gear runs up to 40% faster than stiff Gear on large models, synchronized for use with models simulating digital controllers and is claimed to 'gracefully' terminate if the model detects an invalid operating conditions. In MMS models using EASY5, the largest model run to date was in the order of 100 state variables. The large model handling capability was never tested.

Automatic parameterization

In MMS, substantial amount of user effort is spent parameterizing the modules at various initial conditions. This involves calculation of heat transfer parameters, flow resistances, volumes, areas, etc., only some of which change depending on the

Simplifications (cumulative)	Number of states	Cyber CP-seconds (total)	Run time ratios 1st 10 secs	(CP/sim) last 10 secs	% Error Introduced Final Value (45 s)	Percent error at 10 s
Original model (MMS-01)	57	193	10·75	2·88	Reference Case	
Reduce Thermal-Hydraulics to 4 nodes	49	109	4·08	1·48	+0·5%	+0·5%
Use point kinetics	41	47	3·29	0·82‡	+0·4%	−1·4%
Remove subcooled boiling	40	43	2·32	0·33‡	+0·9%	−5·5%
Reduce decay heat to 3 groups; reduce fuel T model to 2 rad nodes	33	33	1·78	0·28‡	−0·4%	−7·2%
Reduce fuel temp to 1 radial node	29	31	1·32	0·20‡	−2·9%	−15%

† Transient: Trip to natural circulation (no scram) of both recirc. pumps.
‡ Integration time step pegged at 0·5 s; long-term running time can be improved further.

Table 3. Effect of simplifying options on running time and accuracy for a typical Boiling-water Reactor transient.†

model initial conditions. EPRI developed a trial version of a procedure which automatically generates the parameterization given the physical and operating data for each module. These procedure routines can be used to generate parameterization inputs for the models by either ACSL or EASY5 versions. B&W is planning, in the future, to extend this trial version of automatic parameterization procedure to optimize parameters to match a desired model operating point. This combined with automatic model generation pre-processors will allow the user to parameterize the model, generate and initialize the model fairly quickly.

Model reduction

The modular nature of MMS and the simulation package capabilities are allowing MMS users to explain new applications. One such application is the BWR plant analyser. Here, we reduced the BWR model complexity and evaluated the effects of simplifying assumptions on model accuracies. Table 3 gives a summary and the execution times associated with each case for a 45-second natural circulation transient as a result of recirculation pump trips, without scram. The run time was reduced by a factor of 6, with only 0·4% error introduced by reducing the thermal/hydraulic states, decay heat groups neglecting subcooled boiling and using a point kinetics model. This approach was used because ACSL did not provide provisions to perform automatic model reductions.

In another application for designing the digital feedwater control system for a BWR plant, the linearization routines in EASY5 to automatically reduce the order of the model was used. The results are not available at the present time, but the intent is to develop a real-time model for the controller during normal operating conditions and for developing a real-time closed-loop BWR test bed to check-out the controller performance before implementing it in an operating plant.

6. Simulation software design

The discussion, thus far, focused on the simulation languages used in the MMS code development and the deficiencies in the two languages ACSL and EASY5 were highlighted, from the user standpoint. The decision to use ACSL and EASY5 in the MMS code development was taken in 1978 and since then, many changes have taken place in the simulation software development area. There are noticeable differences among the commercial simulation packages available today resulting in a wide range of capabilities for solving specific simulation problems. The tendency on the part of simulation software vendors is to focus on improving characteristics related to input/output features, debugging or diagnostic aids, portability to new and small computers and documentation. While these aspects are very important, the modeler still suffers from poor steady-state solver and integration techniques. Often he is frustrated with situations to make multiple costly runs or model modifications in attempts to reach a solution with accepted computational speed and accuracy.

It is interesting to note how the commercial simulation packages have been developed in the U.S. From CSSL specifications of late 60s and the old CSMP code, CSSL-IV and ACSL have evolved. These codes still expect users to develop his own plant models. On the other hand, plant specific simulation languages such as DSNP. GSMP, GPSAP, SALT and SALT-D have also evolved in attempts to fix the short-

Improvements	Comments
1. Universal pre-processor	Development of a super-macro code which acts as a universal pre-compiler to codes like ACSL, EASY, DARE-P or CSMP. The super-macro code will develop input files for the translator of the selected language. The usefulness of the universal super-macro routines can be further enhanced if a general operating system like the UNIX structure is used in the code.
2. Data Base Management System (DBMS) or interface development	The simulation language should recognize that each user has limited interest in applications and will use only limited utility or analysis routines. The software design should structure it so that the user can access various programs independently, using the data base management system and keep the length of the overall program short.
3. Standardization of compilers, DBMS and graphics packages	Although the general purpose simulation package may not be able to run completely on a personal computer, the basic modules for analysis may be implemented on an IBM-PC (for example). The simulation itself may be performed on a mainframe or a mini computer. For that reason, the standard or most popular compilers and graphics packages for a super-macro would be essential so that the software can be tailored to user needs and executed efficiently. Similarly, some simple but essential features like automatic conversion of single to double precision variables should be offered in the support packages (e.g., integration routines).
4. Library of steady-state solvers and automatic selection/switching	The steady-state solvers in the current simulation packages normally work for the linear problems. The non-linear problems require good initial guesses of the state variables. The general purpose simulation package should have a library of steady-state solvers instead of a single solver based on one method. Also, it will be desirable to offer an automatic switching from one method to another in some classes of problems (example: in ACSL code, switch over from Newton–Raphson to steepest descent method) or based on convergence criteria.
5. Library of integration routines with automatic selection	Most simulation codes provide a library of integration subroutines for solving ODES and PDES. The capability to switch algorithms between stiff and non-stiff ODES is essential, but an automatic recognition of stiff problems and switching an integration routine is desirable. Along the same line, partitioning of state-space problems based on automatic recognition of fast and slow dynamics is helpful. Currently, in solving the power plant transient problems the Gear integration routine changes the order of the integration and the step size automatically, even during minor perturbations. Partitioning the problems and communication between the sub-models at the end of communication intervals may prove to be an efficient way, if properly implemented.

6. Large system solvers	Debugging a large simulation problem tends to be very tedious and expensive. Model partitioning, independent set-up and check-out of submodels at initialization and integration stages are necessary. Large data handling, table look-ups for properties onwards to memory extensions are sometimes necessary. An extensive debugging tool with pre- and post-processors may be reliefs for the users in this area. Model reduction routines may be helpful for redefining large simulation problems.
7. Special problem handlers	The simulation problems dealing with discontinuities, time delay, noisy systems are not very uncommon. The integration packages should exactly locate when the discontinuities occur.
8. Efficiency and user-friendly features	Many times the modelers are frustrated because the simulation stops and no diagnostic messages are given out. Some helpful diagnostics come from automatic model generation, recognition of undefined parameters, continuous integration step sizes, detection of special events like a discontinuity, algebraic loop, etc. The simulation software developers should provide default options to store automatically simulation results upon time outs. Also one should be able to suspend simulation and start with new parameters at the run time.
9. Pre-compiled models and blocks	There are many applications specific simulation packages like DSNP and SALT. One of the strengths of SALT code is that it provides pre-compiled modules and hence, the execution time tends to be minimal. Similar features can be incorporated at the simulation blocks level. For example, each block in ACSL can be developed fully so that the compilation time can be reduced.
10. Benchmark problems	Standard benchmark problems to evaluate several simulation software packages should be developed.

Table 4. Desired improvements in current simulation packages.

comings in CSMP and DSNP. Yet another family of simulation codes consists of DARE-P, EARLY DESIRE, DARE-ELEVEN, PSCSP and MICRODARE. Each of these codes have their own strong features and although most code vendors recognize the shortcomings in their codes, the existing user base and their market focus precludes them from making the codes 'truly' general purpose simulation packages. Based on the strong features in the codes mentioned here and the experiences gained in the MMS code development effort, the following ideas have been compiled. The objective here is to present ideas for the design of a software package that would be more acceptable by simulation software users.

Basically the simulation software design can be viewed from three aspects: (1) user's needs, (2) developer's potential to market the software to different analysts groups, and (3) simulation and analysis routines. The credibility of the language lies in the robustness of the analysis routines in the simulation software to handle small to large size problems described by either ODES and PDES. The analysis routines

may range from classical 'control' related routines to a wide spectrum of routines to include optimal control packages to solve the modern control problems and address noise analysis issues with fast-Fourier transform routines. Typically, the software vendors are trying to extend continuous system packages to handle the discrete systems also. The users, on the other hand, want to be able to utilize the same plant model for various applications, *and* with the latest available hardware system. Considering these trends among the users and developers of the simulation software, it is best recommended to use modularity at each stage and let a data base management system handle the interfaces between various routines. Table 4 lists improvements or typical advances one wishes to see in the current simulation packages.

7. Concluding remarks

The simulation languages have been batch processing oriented software, until recently. Unless problems are extremely large, engineers often prefer to work with engineering work stations today. From that aspect, it is desirable to design the simulation software for interactive use and ideally set up to solve a series of small problems or use different analysis routines to perform different evaluations. File manipulations, together with graphics usage have become essential tools and we can no longer overlook this predominant PC-culture. While the data base management systems, graphics monitors for post- and pre-processors developers might help meet the user-friendly needs, the smaller machines pose more challenges to develop packages with robust numerical routines.

For the mainframe and mini-computer applications, the focus of attention must be on improving the robustness of steady-state and integration solvers. Handling special events, methods to reduce the model generation and computation times with improved run-time commands are emerging.

With many simulation languages available today, the time has come to look at the possibility of developing a super-macro or super pre-compiler to the simulation languages, for use in codes like the Modular Modeling System. This super-macro code would select the modules in a given language (example: ACSL or EASY5 or any other) and create the pre-compiled FORTRAN code. Based on the application problem and size, the user can then set up and run the models on his choice machine. The advances in data base management systems and graphics can be taken advantage of, and even the universal operating systems can be addressed at least at a conceptual level.

References

Divakaruni, S. M. (1985). The application of simulation in large energy system analysis codes. *Modeling, Identification and Control*, **6**,

Gear, W. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice-Hall.)

Smith, L. B., Ummel, B., and Divakaruni, S. M., (1983). Effective utilization of two commercially available simulation packages in MMS code development. SCS Conference, *Vancouver, B.C., July 1983.*