Reviews of Papers in the Computer Field

DONALD L. EPLEY, Paper Reviews Editor
D. W. FIFE, A. I. RUBIN, R. A. SHORT, H. S. STONE
Assistant Paper Reviews Editors

Please address your comments and suggestions to the Paper Reviews Editor: Donald L. Epley, Department of Electrical Engineering, University of Iowa, Iowa City, Iowa 52240.

A. SIMULATION


The authors analyze the behavior of formulas for the numerical integration of ordinary differential equations by considering the frequency response of the “integrators” in the system being simulated. This approach will appeal to engineers, particularly those with an analog background, as being a familiar way to consider the departures from ideal performance of the “computing elements” used as building blocks in simulation. The method has the advantage that the quality of different formulas can be compared independently of any particular problem and irrespective of whether the problem is linear or nonlinear. It does not, of course, give directly the errors in the overall solution, merely the errors injected by the integrating components, but the necessary theory for deducing solution errors from component errors is available if needed.

The authors restrict their attention to multistep formulas of either the simple predictor type or the solved-corrector type. They do not consider the usual practical case where the corrector equation is only approximately solved by using a predictor to give an initial estimate and one iteration of a corrector to refine that estimate, nor do they consider formulas of the Runge-Kutta type. They do not bring out the point that their method is not in fact applicable to these other types of numerical procedure. It is only for the particular types they consider that one can associate a single transfer function with each of the integrators independently of the system in which the integrators are embedded. (In the predictor-corrector case, one would need to consider the integrators as having two different transfer functions between which they alternate, one transfer function applying during the predict and the other during the correct phase; these do not combine into a single effective transfer function independent of the rest of the problem.)

This lack of generality of the integrator transfer function method of analysis no doubt explains why it has not already been adopted by other workers in this field.

Although Giloi and Grebe's method lacks generality it is very good for those types of formula for which it can be used, enabling one to get a quick feel for the order of performance obtainable and providing some guidance in the choice of formula.

An approach that is applicable to all types of formula and yet retains something of the flavor of Giloi and Grebe's method has been described by Baxter [1], Benyon [2], and others. In seeking greater generality as to type of formula one must give up the objective of problem independence. Furthermore, the analysis is feasible for linear constant-parameter problems only. (However, many nonlinear problems are locally linear with coefficients varying slowly enough to be considered constant, so the analysis gives some guide to performance in solving most problems.) By means of a partial fraction expansion the overall transfer function of a linear system can be expressed as a combination of first- and second-order transfer functions. Hence, it is sufficiently general to examine performance of a numerical formula in solving systems of up to only the second order and this greatly reduces the number of problem parameters that have to be considered. As in the Giloi–Grebe method, departures from ideal performance are considered in terms of errors in frequency response. The difference is that instead of the components whose errors are being examined being only pure integrators (transfer function 1/s) they also include first-order transfer functions with various time constants and second-order transfer functions with various natural frequencies and damping ratios.

The deficiencies of the Giloi–Grebe approach show up in their discussion of stability. They are able to establish the conditions for stability of the integrating elements themselves, namely that the roots of the polynomial \(D(z)\) (the denominator of their equation (7)) lie within the unit circle. However, this does not necessarily ensure stability of a system con-
taining these integrating elements. They are forced to introduce an arbitrary "stability margin" to allow for this and "leave it up to the user to determine what is required for his problem" even though their original objective had been to "get away from the unsatisfactory situation where simulation languages provide a number of integration formulas, from which the user can choose, but without having any criteria to guide his choice." Using the approach described in [1] and [2], on the other hand (which leads to the same stability analysis as has been given by Gray [3] and numerous others), readily gives the desired condition. For the class of formulas considered by Giloi and Grebe, the condition for stability of the whole system as distinct from just the integrating elements becomes that the roots of each of the polynomials $D(z) - \lambda_i C(z)$ lie within the unit circle. (Here, $C(z)$ denotes the numerator polynomial in their equation (7), $T$ is the step length, and $\lambda_i$ ($i = 1 \cdots n$) are the poles of the system being simulated.)

It is a little surprising that, having adopted an analog, block-oriented viewpoint, the authors have not made use of the concept of total dynamic error which is now generally used to describe the imperfections in frequency response of analog computing elements. Instead they use the magnitude and phase errors which are the separate in-phase and quadrature components of total dynamic error. It is now generally realized that it is usually only the total dynamic error that counts as neither of the separate components is more serious than the other, except in special cases (such as when the solution to a second order system is required to look right to the eye). Giloi and Grebe, however, regard the phase error as being more serious than the magnitude error. They are therefore led to search for formulas having zero phase error and thence to prefer predictors of even and correctors of odd order.

The authors' general method of attack is to be commended in spite of the shortcomings that have been mentioned and the article should prove of interest to all those concerned with the digital simulation of continuous systems.

P. R. BENYON
Commonwealth Scientific and Industrial Research Organization
Australia

REFERENCES


B. SWITCHING AND AUTOMATA THEORY


This paper concerns itself with rooted trees which have labeled nodes. The labels are taken from a stratified alphabet (each label is associated with a nonnegative number, the number of branches descending from it). The alphabet is divided into terminal and nonterminal labels. Tree generating grammars called regular systems are introduced. A set of trees serve as "axioms" and production rules of the form $\Phi \rightarrow \Psi$, where $\Phi$ and $\Psi$ are trees, allow successive replacement of subtrees $\Phi$ by $\Psi$. The "language" generated by such a system is the set of trees generable from the axioms containing terminal labels only. Such languages, when trees are written linearly (prefix or postfix form) are context free.

When the grammars are written linearly, they are semi-Thue systems, since production rules may be length decreasing. However, the restriction that all strings must map into trees (there must be some stratification assignment) assures that the language generated is context free.

Tree automata are tree analogs of finite state machines. A tree automaton acts on the set of trees over some stratified alphabet, accepting some subset. Intuitively, such an automaton begins at the sequence of endpoints, assigning a state to each endpoint. Moving up the tree, it assigns a state to each node as a function of its label and the sequence of states assigned to the nodes directly below it. If the state assigned to the root node is a designated (final) state, the tree is accepted. A tree automaton may be deterministic or nondeterministic, but for each nondeterministic tree automaton there is a deterministic one accepting the same set.

The author proves that the set of languages accepted by the tree automata is exactly the set generated by the regular systems. Further, the nonterminal labels are not required for generating this set.

All alphabets, trees, sets of production rules, and sets of automata states discussed are finite. The theorems, in general, require these finiteness limitations in their proofs.

The definitions, theorems, and proofs are concise and precise. The casual reader is warned, however, that the abundance of formalism, the sparsity of examples, and the lack of motivating discussion and intuitive explication will probably make for difficult reading for those not familiar with the subject matter.

This paper is essentially a collection of some of the major theorems from the thesis of the author. A review of the basis (taken from the author's abstract) appears in Computing Reviews, vol. 9, no. 2, February 1969. The review states the major theorems, and all but the last (on state minimization of tree automata) appear in this paper.

SAMUEL BERGMAN
Temple University


Stochastic (or, indifferently, probabilistic) sequential machines have been recognized to be important mathematical models for several classes of information processing systems with memory, in which random disturbances cannot be neglected. As examples it suffices to mention the