MODELING, IDENTIFICATION AND CONTROL, 1986, VOL. 7, NO. 2, 71–81
doi:10.4173/mic.1986.2.2

Survey of existing systems for the dynamic simulation of industrial processes†

J. D. PERKINS‡

Keywords: *Dynamic simulation, mathematical software, CSSLs, flowsheeting.*

Dynamic simulation has the reputation of being a costly exercise. Tools to help reduce the development costs of dynamic models are discussed in this paper. Mathematical software libraries and CSSLs are briefly discussed from a user's point of view. Dynamic flowsheeting packages are classified, and one particular equation oriented package, SPEEDUP (Sargent, Perkins and Thomas (1982)) is discussed in detail.

1. Introduction

Several papers in this conference have sought to establish the importance of dynamic simulation in the engineering of large processes (Perkins, Howell and Wong. (1985), Womack (1985), Divakaruni (1985)). Nevertheless, progress in the consideration of process dynamics in industry seems rather slow. One reason for this is the high cost associated with the performance of a typical industrial dynamic simulation study. In this paper, some of the software tools available to help reduce the costs of development of plant simulations will be reviewed.

It is helpful to begin with a discussion of the attributes of process simulation problems. This discussion will provide a context within which to discuss the various software solutions available. A list of key attributes is shown in Table 1. Most simulation problems are described by mixed systems of non-linear differential and algebraic equations, and typically somewhere between one hundred and ten thousand equations and variables are involved. The problem is highly structured, in that each equation contains only a few of the problem variables, but the structure is not regular. Many problems are stiff, that is they involve widely differing time constants (see Norsett (1985)), and therefore require special integration algorithms to be used. Distributed parameter phenomena are quite common, as are pure time delays. Discontinuities of two kinds can occur: explicit, where the change occurs at a known time in the simulation; and implicit, where a change may occur depending on the state of the system.

Each of these attributes requires special precautions in order to achieve a solution. Their combined effect is to make process simulation problems rather difficult to solve.

The tools available to help engineers in the solution of simulation problems fall into three categories. First, there are libraries of mathematical software, implementing the algorithms discussed by Norsett (1985) at this conference. Second, there are

Received 20 February 1986.

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

‡ Department of Chemical Engineering, Imperial College, London SW7 2BY.

Differential *and* algebraic equations
Non-linear
Large (100s–1000s of equations and variables)
Structured
Stiff
Time delays, and distributed parameters
Discontinuities

Table 1. Attributes of process simulation problems.

the Continuous System Simulation Languages (CSSLs) which have been excellently reviewed by Rimvall (1985). Third, there have been efforts to provide for process engineers interested in dynamic simulation the kind of facility available for steady-state simulation (i.e. Dynamic flowsheeting packages). This last category will be the main subject of this paper, but before turning our attention to dynamic flowsheet simulators, a few comments will be made about numerical software and CSSLs from a user's perspective.

2. Some remarks on mathematical software and CSSLs

There is no longer any excuse for engineers to program their own versions of standard integration algorithms. High quality codes now exist implementing all the standard algorithms, and recommended software has been listed by several authors (see Carnathan and Wilkes (1980) and Norsett (1985)). The use of standard codes of this type reduces the work necessary to develop a dynamic simulation. Nevertheless, the user of such codes needs to pay close attention to a large number of issues, and to be knowledgeable in high level language programming and in numerical methods (in order to understand diagnostics generated by the codes).

The issues to be considered arise from the attributes of a typical problem listed in Table 1. Many codes are for differential equations only: where such a code is all that is available, the user must devise a procedure to solve the algebraic equations for the algebraic variables. He must convert the model from the form

$$\frac{dx}{dt} = f(x,y) \tag{1}$$

$$0 = g(x,y) \tag{2}$$

to the form

$$\frac{dx}{dt} = f(x,\ y(x)) \tag{3}$$

by rearranging eqn. (2) into procedure to calculate y given x

$$y = y(x) \tag{4}$$

This can represent a significant task, even for a model in the form of eqns. (1) and (2). For some models (so called high index models, Norsett (1985)) obtaining this form is non-trivial, involving not only algebraic manipulations but also differentiations of the original equations.

Large, structured problems require the use of sparse matrix software, and it is the user's responsibility to set up the structure of his problem correctly for the

sparse matrix codes. The presence of time delays requires special action. Either this phenomenon can be represented as a distributed parameter system, in which case the correct advection equation must be set up and discretized by the user, or the delay may be dealt with by storage and retrieval of past values.

Algorithms for integration assume smoothness of the solution trajectories. The presence of a discontinuity may require special action (see Norsett (1985)). Essentially, integration must be stopped and restarted at each discontinuity. This is straightforward for explicit discontinuities, since the value of the independent variable at the discontinuity is known. For implicit discontinuities, iteration may be necessary to determine the time where the change occurs. Some codes include features to deal with discontinuities routinely; most do not. In the latter case, the user must deal with the situation himself.

In view of all these complications, it is not surprising that simulation has a reputation for being difficult and expensive! The objective of CSSLs is to reduce the complexity by removing many of these responsibilities from users. The user's responsibility is to present his problem to the CSSL essentially in algebraic terms. All other issues are handled automatically. However, it is usually the case that algebraic equations must be presented in explicit form (i.e. like eqn. (4) rather than eqn. (2)). Facilities for the automatic solution of sets of algebraic equations tend not to be a part of CSSLs, so eqn. (2) is not too easy to deal with. In addition, it can be quite difficult to establish initial conditions for the model. Many simulation problems are set up as perturbations from an initial steady-state. Thus, in principle they may be initialized by solving

$$f(x, y) = 0 \tag{5}$$

$$g(x, y) = 0 \tag{6}$$

Some CSSLs contain automatic procedures to enable this kind of initialization; the majority do not, implying that it is often almost as much effort to initialize a model as to write the model itself.

3. Dynamic flowsheeting packages

Several workers have sought to provide for dynamic simulation facilities analogous to those provided by steady-state flowsheeting packages. The important attributes of these systems are shown in Table 2.

The Table lists those attributes common to all packages in this class. There are also significant differences between the packages, most importantly in terms of the way models are represented in the system and the implied solution strategy. In steady-state packages, two distinct strategies have emerged. In the first (both historically and in terms of frequency of use), each process unit is represented as a pro-

Engineering—oriented input
Libraries—process models
physical properties
numerical methods
Process-oriented diagnostics

Table 2. Attributes of flowsheeting package.

cedure which calculates output streams from the unit given input streams to the unit, and various unit characteristics called equipment parameters. A process model consisting of units connected together is solved by executing the procedures in sequence using the results of one calculation as input to the next. Recycles in the plant are handled by tearing of streams, i.e. guessing and iterating on their values. Thus strategy is called *sequential-modular.*

The *equation-oriented* strategy represents units as sets of describing equations. A process model is assembled from these unit equations and the equations are solved simultaneously (exploiting the problem structure appropriately).

It should be noted that the two approaches have few implications in terms of the way problems are described by users. Both have equally 'modular' input languages. Also, today's steady-state simulation packages tend to contain features from both approaches blended in various ways (see Biegler (1983), Perkins (1983)).

The development of dynamic flowsheeting packages has mirrored to some extent that of steady-state simulations, although as we shall see the classification is not quite as straightforward. The equation-oriented category is analogous to the steady-state case: models are represented as sets of (differential and algebraic) equations and integration is by simultaneous solution of all model equations. However, even within this category there are differences in the way algebraic equations are represented and handled. The early attempts were based on the CSSL approach, and thus required the algebraic equations to be written explicitly in the algebraic variables (cf. eqn. (4)). DPS (1974) is one example of this type of system. Later packages e.g. ASCEND (Kuru (1981)) and SPEEDUP (Sargent *et al.* (1982)) allow arbitrarily complex algebraic equations with no implication of calculation strategy.

Within the sequential-modular category, there are also two types of simulation strategy. The different architectures are shown in Fig. 1. The first architecture is truly sequential-modular, since all calculations associated with a unit are performed within the unit module, including integration of any differential equations associated with the unit. This is the class of simulator discussed by Brosilow elsewhere in this conference (Brosilow (1985)). The second class of modular system has many features in common with the equation-oriented approach in that all differential equations in a process model are integrated simultaneously by one global algorithm. The function of the modules is, given values for the input streams, equipment parameters (which may now be time-varying) and also the current values of the state variables for the unit model, to calculate the current output streams, and values of the right-hand sides of the state equations (1) for the model. An example of a package in this category is DYNSYL (Patterson and Rozsa (1980)).

Whilst there has been research into the properties of the different approaches to dynamic simulation, very few dynamic flowsheeting packages have been developed to a commercial standard. Some exceptions are referenced in the discussion above. In the remainder of this paper, a project is discussed the aim of which is to develop a dynamic flowsheeting package for industrial use.

4. SPEEDUP: an equation-based dynamic flowsheeting package

The SPEEDUP project began in the early 1960s at Imperial College. It represents an attempt to provide an integrated framework for process simulation, so that process engineers may perform steady-state and dynamic simulation and optimization calculations using the same package. In addition, facilities exist for control system analysis based on linearizations of the non-linear SPEEDUP model. (See

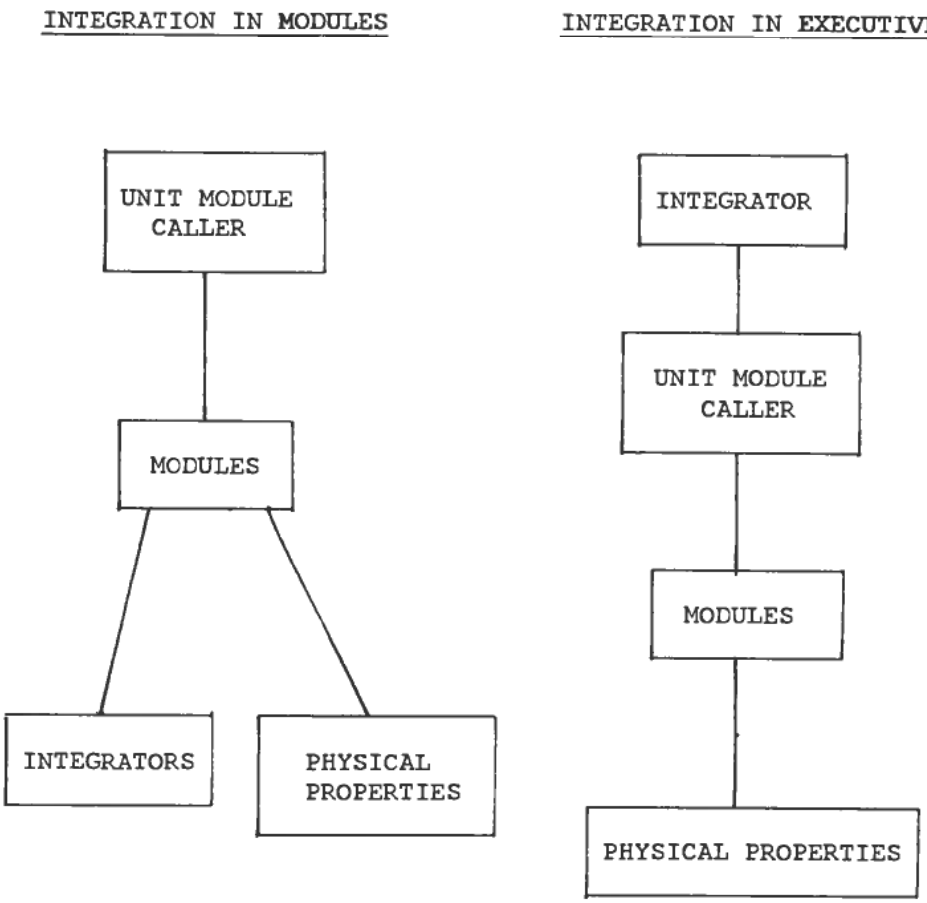


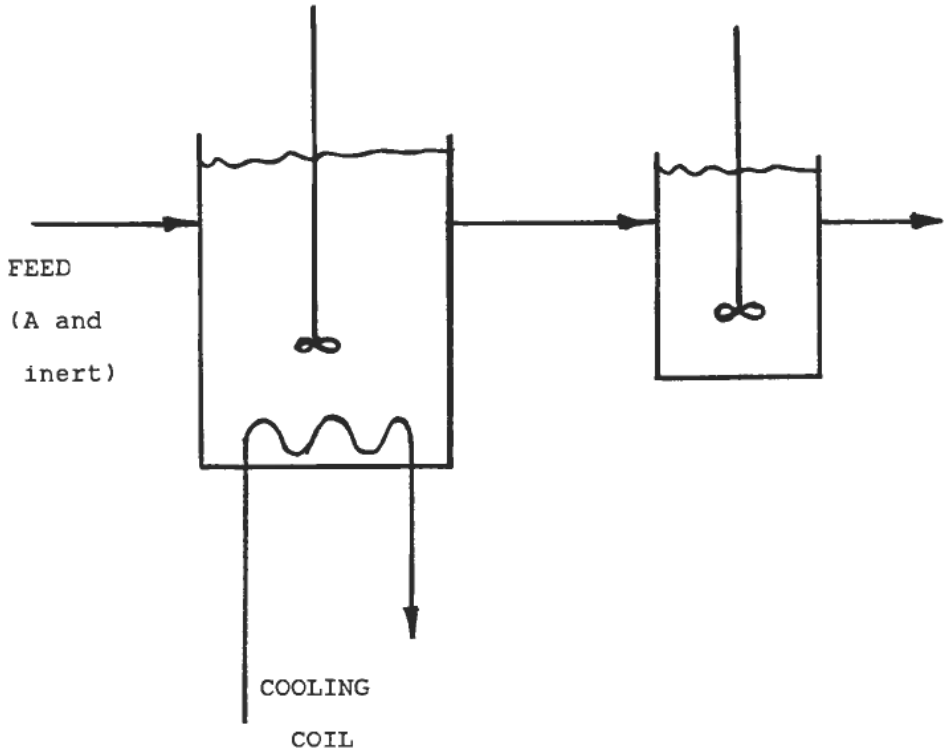
Figure 1. Sequential modular architectures.

Sargent *et al.* (1982) for a description of SPEEDUP and its history, and Perkins and Wong (1985) for a description of the control system analysis facilities.) The latest version was developed with financial support from Science and Engineering Research Council and British Technology Group.

The core of the SPEEDUP system is a data-base representing the current state of the design. Input information and commands for carrying out the various phases of the design are expressed in a special engineering-oriented language. The system interfaces with a data-bank which contains design data (plant models, physical properties, costs) and a library of FORTRAN sub-routines (physical property correlations, numerical procedures) used by the executive.

The information representing the current state of the design is of several distinct kinds, each of which is stored in its own section of the database. The sections we shall consider here are those used to specify the simulation problem:

FLOWSHEET	The flow diagram of the process, indicating the connections between the various units
MODEL	The describing equations for each of the unit-type occurring in the process
UNIT	The design specifications for each unit
OPERATION	The operating policy to be followed during the simulation.



Reaction: A → B (first-order exothermic)

Reactors initially filled with inert.

Figure 2. A continuous stirred-tank reactor system.

A brief description of some of the statements used to define the information is given below. The statements are illustrated by reference to a simple example; the start-up of a continuous stirred-tank reactor system. The flow-diagram of the system is shown in Fig. 2.

Flowsheet. The first step in the specification of problems to SPEEDUP is to define the process flow-diagram. Each unit in the flowsheet is named, and the connections between units are specified using link statements. The FLOWSHEET section for the reactor system is shown in Fig. 3.

Model. Each unit type has a MODEL section which gives the describing equations relating the input and output stream variables and the unit design parameters. An example of a MODEL section, specifying the reactors in our problem, is shown in Fig. 4. All text enclosed by the symbols #...# is treated as commentary by SPEEDUP. As with a FORTRAN sub-routine, the variables used in defining the model are specific to the model, and can be simple variables or members of arrays. As well as the describing equations themselves, we must also define the variables

```
FLOWSHEET
  FEED IS INPUT TO REACTOR 1
  OUTPUT OF REACTOR 1 IS INPUT TO REACTOR 2
  OUTPUT OF REACTOR 2 IS PRODUCT
```

Figure 3. The flowsheet section for the reactor system.

```
MODEL FIRST_ORDER_REACTOR
#   MODEL OF 1ST ORDER EXOTHERMIC REACTOR A→B   #
    SET NOCOMP, KEY
    TYPE
      TEMP_IN, TEMP_OUT, TEMP_COOLING_WATER AS TEMPERATURE
      PRESS_IN, PRESS_OUT, PRESS_DROP AS PRESSURE
      MOLE_FLOW_IN, MOLE_FLOW_OUT AS ARRAY (NOCOMP OF
      MOLEFLOWRATE
      T_MOLE_FLOW AS MOLEFLOWRATE
      X_OUT AS ARRAY (NO COMP) OF MOLEFRACTION
      BULK_DENS AS DENSITY
      NU AS ARRAY (NOCOMP) OF NOTYPE
      AREA, HEAT_TRANSFER_COEFFICIENT AS NOTYPE
      RATE_CONSTANT, GAS_CONSTANT,RATE AS NOTYPE
      HOLDUP AS NOTYPE
      REACTOR_VOLUME AS VOLUME
      ACTIVATION_ENERGY, DELTA_H AS ENERGY
      CP AS HEATCAPACITY
      STREAM
        INPUT    PRESS_IN.    TEMP_IN,    MOLE_FLOW_IN
        OUTPUT   PRESS_OUT,   TEMP_OUT,   MOLE_FLOW_OUT
      EQUATION
        PRESS_IN          =  PRESS_OUT + PRESS_DROP;
        T_MOLE_FLOW       =  SIGMA(MOLE_FLOW_IN);
X_OUT*T_MOLE_FLOW         =  MOLE_FLOW_OUT;
        RATE              =  EXP(-ACTIVATION_ENERGY/GAS_
                             CONSTANT/TEMP_OUT)
                             *RATE_CONSTANT*X_OUT(KEY);
        HOLDUP            =  BULK_DENS*REACTOR_VOLUME;
#     DYNAMIC EQUATIONS FOLLOW                          #
        HOLDUP*EX_OUT     =  MOLE_FLOW_IN - T_MOLE_FLOW
                             *X_OUT
                              + NU*RATE*HOLDUP;
HOLDUP*CP
        *  TEMPOUT        =  T_MOLE_IN - T_MOLE_FLOW
                             *X_OUT
;                             + NU*RATE*HOLDUP
HOLDUP*CP
        *  TEMP_OUT       =  T_MOLE_FLOW*CP*(TEMP_IN -
                             TEMP_OUT)
                             - RATE*HOLDUP*DELTA_H
                             - AREA*HEAT_TRANSFER
                               COEFFICIENT
                             *(TEMP_OUT - TEMP_COOLING
                               _WATER) ;
```

Figure 4. A MODEL of a continuous stirred-tank reactor.

associated with each input and output stream, and as in FORTRAN we must define the dimensions of any arrays of variables so that the appropriate storage can be allocated. The MODEL section therefore has various subsections:

SET. This section is used to declare those variables (e.g. array dimensions) which must be set in order for the model to be defined. In addition the SET section may be used to give values to physical constants appearing in the model.

TYPE. In this section, the variables used in the model are declared and arrays are dimensioned. Note that we may use parameters declared in the SET section as dimensions. The type associates a physical unit to the variable, together with default initial values and upper and lower bounds on the variable.

STREAMS. Here the variables associated with inputs and outputs to the unit are declared. Note that input and output streams do not have a fixed format, but can be defined to be any collection of variables in the model. Thus a model need not represent a plant unit, but is simply a collection of equations relating sets of variables. Furthermore, since the model is not a procedure, there is no directionality implied by the definitions of 'input' and 'output' streams. Therefore SPEEDUP can be used to handle any type of network, or interrelated system of equations. The nomenclature employed is simply to conform with the usual way of describing process flowsheets.

EQUATIONS. Equations are written in the form

(arithmetic expression) = (arithmetic expression)

where the expressions follow FORTRAN conventions. Some extra facilities provided by SPEEDUP are illustrated in Fig. 4. The $ symbol implies a derivative with respect to time. Equations can be defined using vector arithmetic. For example, the equation

X_OUT*T_MOLE_FLOW = MOLE_FLOW_OUT ;

defines the relationship between the total molar flow, the molar flows of each component and the mole fraction of each component in the output stream from the reactor.

Thus, to provide a new model to the simulator, it is only necessary to provide describing equations. Since it is quite likely that it will be necessary to write one-off models for some units in almost all dynamic simulation studies, it is important that model writing be made as simple as possible.

The MODEL section describe the models which can be used in the simulation. The UNIT section is used to define which models will be used to simulate particular units, and also gives dimensioning information. An example of a UNIT section taken from our example is shown in Fig. 5.

Operation In this section, values are given to all known design parameters in the simulation. Also, initial conditions may be set for variables, or initial guesses for unknowns in steady-state problems. The OPERATION section for our example is shown in Fig. 6. The SET section is used to specify design parameters. The PRESET section may be used to specify initial conditions. Alternatively, the initial conditions may be established by a steady-state solution of the model.

```
UNIT REACTOR_2 IS FIRST_ORDER_REACTOR
SET NOCOMP = 3, KEY = 1
```

Figure 5. An example of a UNIT section.

```
OPERATION
  SET
    WITHIN REACTOR_1
      REACTOR_VOLUME = 1, PRESS_DROP = 0.5, PRESS_IN = 1.65
      TEMP_IN = 300.0,
      MOLE_FLOW_IN(1) = 0.5,
      MOLE_FLOW_IN(2) = 0.0,
      MOLE_FLOW_IN(3) = 2.278,
      RATE_CONSTANT = 1.7654E15,
      ACTIVATION_ENERGY = 1.0E5,
      DELTA_H = -5.957E4,
      GAS_CONSTANT = 8.314,
      NU(1) = -1,
      NU(2) = 1,
      NU(3) = 0,
      CP = 75.24,
      AREA = 10.0,
      HEAT_TRANSFER_COEFFICIENT = 62.705,
      TEMP_COOLING_WATER = 300.0,
      BULK_DENS = 55.6
    WITHIN REACTOR_2
      REACTOR_VOLUME = 0.1, PRESS_DROP = 0.2,
      RATE_CONSTANT = 1.7654E15,
      ACTIVATION_ENERGY = 1.0E5,
      DELTA_H = -5.957E4, GAS_CONSTANT = 8.314,
      NU(1) = -1, NU(2) = 1, NU(3) = 0,
      CP = 75.24, AREA = 10.0,
      HEAT_TRANSFER_COEFFICIENT = 0.0
      TEMP_COOLING_WATER = 300, BULK_DENS = 55.6
      PRESET
     WITHIN REACTOR_1
      X_OUT(1) = 0.0,
      X_OUT(2) = 0.0,
      X_OUT(3) = 1.0,
      TEMP_OUT = 300.0
     WITHIN REACTOR_2
      X_OUT(1) = 0.0,
      X_OUT(2) = 0.0,
      X_OUT(3) = 1.0,
      TEMP_OUT = 300
```

Figure 6. The OPERATION section for the reactors simulation problem.

The SPEEDUP system is programmed in PASCAL for ease of production, maintenance and portability and produces a FORTRAN program to perform the numerical calculations. The user interacts with a controlling executive, known as SPEEDIT, which allows entry and editing of the data and initiation of a run. SPEEDIT manages the files associated with SPEEDUP and controls the sequence of the simulation. The interfaces with machine dependent parameters are dealt with here and are well defined.

Supported by SPEEDIT the translator takes the SPEEDUP description of the problem and produces FORTRAN code which is compiled and then linked with libraries of physical property and numerical routines to perform the simulation. Facilities are provided to halt the execution after a specified time, when control

returns to the user, who may examine the results in graphical or tabular form and then restart the simulation, possibly having changed the values of some variables.

Great care has been taken to provide efficient and robust numerical techniques for the mathematical problems involved, viz the solution of large sets of non-linear algebraic equations, and of mixed systems of differential and algebraic equations. Recent developments include the automatic generation of analytical derivatives, and their use in all the numerical algorithms (see Pantelides (1985) for a review of the latest numerical techniques; Perkins (1983) gives a summary of earlier research on numerical methods).

An evaluation of the steady-state capabilities of a development version of SPEEDUP has recently been published by EXXON (Gupta, Lavoie and Radcliffe (1984)). Industrial evaluations of the dynamic capabilities of the package are currently in progress.

5. Conclusion

Tools available to aid in the development of dynamic simulations of industrial processes have been reviewed. They are of three kinds, each seeking to reduce the amount of work necessary to develop a simulation by removing some responsibility from the user. Use of numerical software libraries removes the need to develop one's own implementation of standard algorithms. This is the lowest level of support. CSSLs remove responsibility for detailed programming skills by permitting model definition in algebraic terms. The greatest level of support is provided by flowsheeting packages, where a library of standard models is provided in addition to the earlier features.

One development of importance is the effort to provide a unified framework for all large-scale continuous simulation, both steady-state and dynamic. One example of such a system has been described in some detail in this paper. By providing one package for all an engineer's simulation needs, it is hoped to break down the artificial barriers between different categories of simulator.

References

Biegler, L. T. (1983). Simultaneous modular simulation and optimization. Presented at '*Foundations of Computer-aided Process Design*', Snowmass, Colorado, USA, June 19–24.

Brosilow, C. (1985). Modular integration. *Modeling, Identification and Control*, 1985, **6**, 3, 153–179.

Carnahan, B., and Wilkes, J. O. (1980). Numerical solution of differential equations—an overview. Presented at *'Foundations of Computer-aided Process Design'*, Henniker, New Hampshire, USA, July 6–11.

Divakaruni, S. M. (1985). Application of dynamic simulation in energy systems analysis. *Modeling, Identification and Control*, 1985, **6**, 4, 231–247.

DPS Users' Manual (1974). Information Technology Promotion Agency, Japan.

Gupta, P. K., Lavoie, R. C., and Radcliffe, R. R. (1984). An industry evaluation of SPEEDUP, Presented at *AIChE National Meeting*, San Francisco, November, 1984.

Kuru, S. (1981). Dynamic simulation with an equation based flowsheeting system, PhD Thesis, Carnegie Mellon Univ., Pittsburgh.

Norsett, S. (1985). Methods for solving systems of algebraic and differential equations. *Modeling, Identification and Control*, 1985, **6**, 3, 141–152.

Pantelides, C. C. (1985). Numerical methods for large scale continuous simulation, Presented at IBM Europe Institute *'Large Scale Simulation'*, Oberlech, Austria.

PATTERSON, G. K., and ROZSA, R. B. (1980). DYNSYL: a general-purpose dynamic simulator for chemical processes. *Comp. & Chem. Eng.* **4,** 1–20.

PERKINS, J. D. (1983). Equation oriented flowsheeting, Presented at *'Foundations of Computer-Aided Process Design'*, Snowmass, Colorado, USA, June 19–24, 1983.

PERKINS, J. D., HOWELL, J. M., and WONG, M. P. F. (1985). The significance of modeling and simulation in process design. *Modeling, Identification and Control*, 1985, **7,** 2, 57–70.

PERKINS, J. D., and WONG, M. P. F. (1985). Assessing controllability of chemical plants. Presented at PSE 85, Cambridge, UK, Mar 31–Apr 4.

RIMVALL, M. (1985). Evolution and perspectives of simulation languages following the CSSL standard. *Modeling, Identification and Control*, **6,** 4, 181–199.

SARGENT, R. W. H., PERKINS, J. D., and THOMAS, S. (1982). SPEEDUP: Simulation program for the economic evaluation and design of unified processes in *Computer-aided Process Plant Design* (Gulf, Houston), edited by M. E. Leesley.

WOMACK, J. M. (1985). Dynamic simulation in the process industries: case studies from MOBIL. *Modeling, Identification and Control*, 1985, **6,** 4, 201–210.