MODELING, IDENTIFICATION AND CONTROL, 1989, VOL. 10, NO. 3, 135–141
doi:10.4173/mic.1989.3.2

The role of computational efficiency in process simulation

KURT STRAND†, HÅVARD NORDHUS†,
PETER SINGSTAD† and MAGNE HILLESTAD‡

Keywords: *Dynamic simulation, integration algorithms, steady state solvers, simulation software.*

It is demonstrated how efficient numerical algorithms may be combined to yield a powerful environment for analysing and simulating dynamic systems. The importance of using efficient numerical algorithms is emphasized and demonstrated through examples from the petrochemical industry.

Based on state of the art numerical algorithms, a simulation system has been developed, and its major characteristics are described. The system was originally developed to enhance control system synthesis and testing for a polyethylene reactor, but is presently used for other processes. The general features and flexibility of the system makes it ideally suited for analysing and simulating industrial processes and dynamic systems in general.

1. Introduction

Process simulation analysts have a wide range of options for computer based simulation tools today. The list of simulation software and application programs for both steady state and dynamic simulation is long, and is steadily increasing. They range from small programs running on a Personal Computer to comprehensive simulation programs, such as the CSSLs. It may be difficult for non-specialists to choose among this variety of computer based simulation tools, and to emphasize their most important features. The pace in which programs are developed makes it difficult to stay up to date, even for those who work in this field.

Computer simulation may serve a variety of purposes, and the characteristics/attributes of the problem as well as the features of the simulation programs influence the choice of software. It is beyond the scope of this paper to go into details about the particular attributes of the variety of tools for process simulation on the market, general numerical aspects related to process simulation will be treated.

The solution strategy applied in most simulation software for dynamic analysis may be classified into two types, modular integration and equation-oriented. The modular integration strategy is characterized in that the unit process models are calculated independently of each other, and interconnections and recycles in the process are handled by a final guessing and iterating on stream variables. Different numerical algorithms may then be applied inside each unit model, which therefore allow different time scales. However, the modules must be coordinated so that the

Received 1 August 1989.
† SINTEF, Division of Automatic Control, N-7034 Trondheim-NTH.
‡ STATOIL, Research and Development, Refining and Petrochemicals, N-7004 Trondheim.
This paper was presented at SIMS '89 (Scandinavian Simulation Society) 31st Annual Meeting, Bergen, Norway, May 31–June 2 1989.

interconnection variables are sufficiently accurate. The equation-oriented strategy is a simpler and more straightforward approach, in the sense that all equations are solved simultaneously. Different time scales or numerical algorithms for each process unit model are then impossible. However, modular model input and the equation-oriented solution strategy may be used together (Perkins 1986).

Simulation models may be classified according to their characteristics/attributes

model form/linearity
model size (number of equations/unit operations)
dynamic and/or steady state
lumped and/or distributed parameter systems
eigenvalue range (stiffness)

For dynamic systems where a lumping of system parameters provides satisfactory representations, the resulting mathematical models will typically be ordinary differential equations (ODE's) in the time domain,

$$\frac{d}{dt}\boldsymbol{x}=\boldsymbol{f}(\boldsymbol{x},t) \tag{1}$$

or, for some cases, a system of coupled differential and algebraic equations (DAE-system),

$$\frac{d}{dt}\boldsymbol{x}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t) \tag{2a}$$

$$\boldsymbol{0}=\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},t) \tag{2b}$$

For most process simulation problems, DAE-systems appear naturally. This is due to the presence of thermodynamics relations, such equations of state (EOSs). In some cases, as much as 70–90% of the CPU-time is spent in thermo-physical property routines. When comprehensive thermo-physical property models, taking into account highly non-ideal fluids, are used, the potential for saving CPU-time is substantial. By introducting local thermodynamic models, more economic use of these comprehensive TP-models is accomplished without a loss of accuracy. The local TP-models may be of the explicit form, $\boldsymbol{u}=\boldsymbol{u}(\boldsymbol{x},t)$, and substituted into eqn. (2*a*), the problem may be transformed into a system of ODE's.

Systems which extend over major regions in space and where a direct lumped representation is inappropriate, are termed distributed parameter systems. Such phenomena are typically described by integral or partial differential equations. Different mathematical techniques may be used for approximating such systems. For time-dependent (dynamic) systems, semi- and full-discretizing methods are distinguished between. For semi-discretizing methods or 'methods of lines', the approximation is made in two steps, i.e. first discretize the space dimensions, and secondly integrate the resulting ordinary differential equations (ODEs). For full-discretizing methods, the space and time dimensions are discretized simultaneously. Any finite difference or finite elements methods, may be either full-discretizing methods or 'methods of lines'. Using a 'method of lines' has the advantage that standard integration algorithms for ODE's may be applied, and eventual interconnected lumped parameter models may be solved simultaneously. The following example will be used to demonstrate important issues in process simulation.

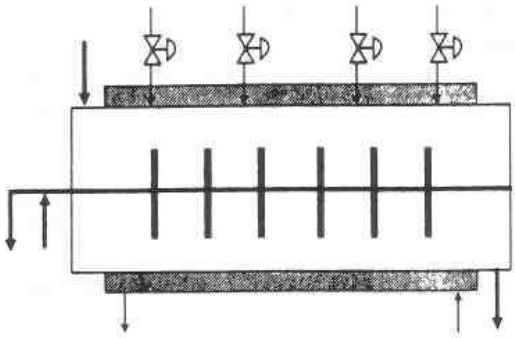

Figure 1. A stirred bed polymer reactor.

2. Simulating a polymer reactor

In this polymer reactor example a 'method of lines' is used, which is based on a finite volume formulation. When discretizing distributed parameter systems, such as the horizontal stirred bed polymer reactor in Fig. 1, a large number of ordinary differential equations (ODE's) results. For this particular example, we have 344 ODE's.

2.1. *Dynamic analysis*

In general, systems of ODE's which originate from a 'method of lines', are characterized by sparse or banded Jacobians, and by eigenvalues which differ by orders of magnitude. For this polymer reactor model, the ratio between the largest and smallest eigenvalue is approximately 10^7. The smallest time-constant is $\approx 0{\cdot}2$ msec, while the largest is $\approx 0{\cdot}5$ hours, so this system may be considered as a 'stiff' numerical problem. The concept of 'stiffness' is somewhat fuzzy, and its definition may vary. However, it usually refers to the fact that the time scales of the fastest and slowest modes are very different.

Numerical integration methods may grossly be divided into two classes, explicit and implicit methods. Explicit methods, such as Euler, Adams Bashforth, explicit Runge Kutta etc., are characterized by limited stability regions. For 'stiff' numerical problems, this implies that the step size will be restricted by, and be in the order of the largest eigenvalue. Implicit methods have the advantage of being numerically stable for almost any combination of eigenvalues and step size. However, a disadvantage of using implicit methods is the fact that the Jacobian, $\boldsymbol{J}=\partial \boldsymbol{F}/\partial \boldsymbol{x}$, must be provided. For large systems of equations this may involve much computational effort (dim $\boldsymbol{J}=NxN$, where N is the system order). The most frequently used implicit methods for solving engineering problems are Gear's BDF-methods (Backward Differentiation Formulas).

In the following table, different numerical methods are compared by simulating the reactor model. A step change in a back mixing parameter is introduced, and the model is simulated for 3 h, which is almost to steady state. The same error tolerances are used for all methods, and for the implicit methods, the Jacobian is estimated with internal approximations. It is obvious that explicit/semi-explicit (Adams Predictor/Corrector) methods are inappropriate for this reactor model. This is due to stability-restricted step size.

For the implicit methods, a certain portion of the total number of f-evaluations is used for Jacobian-estimation. The standard BDF-algorithm (LSODE, MF = 22), with full Jacobian, requires $N+1$ f-evaluations for each Jacobian-evaluation. The BDF-algorithm with sparse matrix handling (LSODES, Hindmarsh, 1983), requires

Method	Adams (PC) variable order	Standard BDF (Gear) full matrix	BDF with sparse matrix handling
Number of steps	$\approx 10^7$	85	92
Number of J-eval.	0	21	5
Number of f-eval.	$\approx 10^7$	7342	704
CPU-time (sec)[1]	$\approx$ days	1416	138

[1] On a VaxStation 3500.

Table 1. Efficiency of different numerical algorithms in solving the polymer reactor problem.

$N+1$ f-evaluations once only to determine the sparsity structure, and otherwise only a limited number of f-evaluations (≈ 50 in this example) are needed. The large difference in the total number of f-evaluations is mainly due to the fact that almost 94% of the elements of the Jacobian are zero elements. Since the standard BDF-algorithm uses a full Jacobian, much wasted work is done. The difference in number of f-evaluations for the two BDF-algorithms may have been reduced, if the band-structure of the actual Jacobian had been accounted for in the standard BDF-algorithm. The two algorithms are comparable with respect to the number of integration steps. The number of J-evaluations differs with a ratio of four, which may be due to the different treatment of the Jacobian and to implementational differences.

These results are by no means general, but they clearly illustrate that there are great savings in choosing the most appropriate algorithm. For most problems arising from distributed parameter systems, the BDF-algorithm with sparse matrix handling has been found to be superior over most methods. That may also be the case for models arising from large systems of interconnected units/components (sparse Jacobian). However, precautions must be taken if the structure of the Jacobian changes during the simulation (due to discontinuities for instance).

In this example the system was simulated almost to steady state. Often, it is the period shortly after transients which is of most interest to study. In such cases, explicit methods may be favourable, because implicit methods initially do much work to estimate the Jacobian.

2.2. *Steady state analysis*

In many cases only steady state behaviour is to be studied. The steady state solution may be considered as a particular solution of eqn. (1),

$$\mathbf{0} = \mathbf{f}(\mathbf{x}, t) \tag{3}$$

which yields:

$$\frac{d}{dt}\mathbf{x} = \mathbf{0} \tag{4}$$

It is not obvious, however, what is the most efficient approach, either a direct solution of the algebraic problem, eqn. (3), or be integrating eqn. (1) to steady state. It depends both on the solvers being used and on model characteristics, such as the eigenvalues, system order and degree of nonlinearity. As treated by Divakaruni (1985), in his comparison of the ACSL and EASY5 simulation languages, a number of approaches may be used to obtain steady state. One of the algorithms with the best convergence properties in his examples was the Powell-hybrid method from MINPACK. In the

Method	Powell-hybrid method	BDF with sparse matrix handling
Number of steps		35
Number of *J*-eval.	1	1
Number of *f*-eval.	346	434
Final residual	$5{\cdot}0 \cdot 10^{-5}$	$6{\cdot}3 \cdot 10^{-6}$
CPU-time (sec)[1]	841	82

[1] On a VaxStation 3500.

Table 2. Efficiency of different approaches in solving the steady state polymer reactor problem.

following example, the performance of this algorithm is compared with the integration approach using LSODES.

The Euclidean norm of the residuals ($\sqrt{\Sigma(dx_i/dt)^2}$) is initially 42 (for larger residuals, i.e. 3000 and 800, the Powell-hybrid method failed to converge). The major difference between these two approaches is seen in the amount of CPU-time consumed, which is mainly due to the fact that the Powell-hybrid method use the full Jacobian.

A general conclusion is that the integration approach does always provide a solution. So problems with solution-divergence, which unfortunately are found by most solvers of nonlinear algebraic equations, is avoided.

3. A new system for analysing industrial processes

On the basis of some of the above conclusions, a new system for analysing/simulating industrial (and other) processes has been developed. The system was originally developed to enhance control system synthesis and testing for a polyethylene reactor. At that time, the numerical capabilities of available software was considered to be unsatisfactory, and the flexibility of a self-developed system was emphasized. The numerical aspects of the simulation was assumed to be very important, since polyethylene reactor dynamics is extremely fast. This has been confirmed.

Since then, new features have been added, and the system has gradually evolved. Its favourable attributes are the numerical capabilities. State of the art numerical algorithms from a number of public libraries are included (LINPACK, EISPACK, MINPACK, ODEPACK and DEPAC), and new algorithms are added as required. The major attributes of the system are shown in figure 2. The system is entirely menu-driven, both with respect to the model configuration part, and to the analysis part.

Model configuration is menu-driven and the user may choose among a number of units in the model library when building process models. He may also add his own models. The following unit models are so far included in the model library:

polymer reactors
tanks
valves
separators
reciprocating compressors (both simple and advanced)
heat exchangers
pipes (both lumped and distributed)
pumps

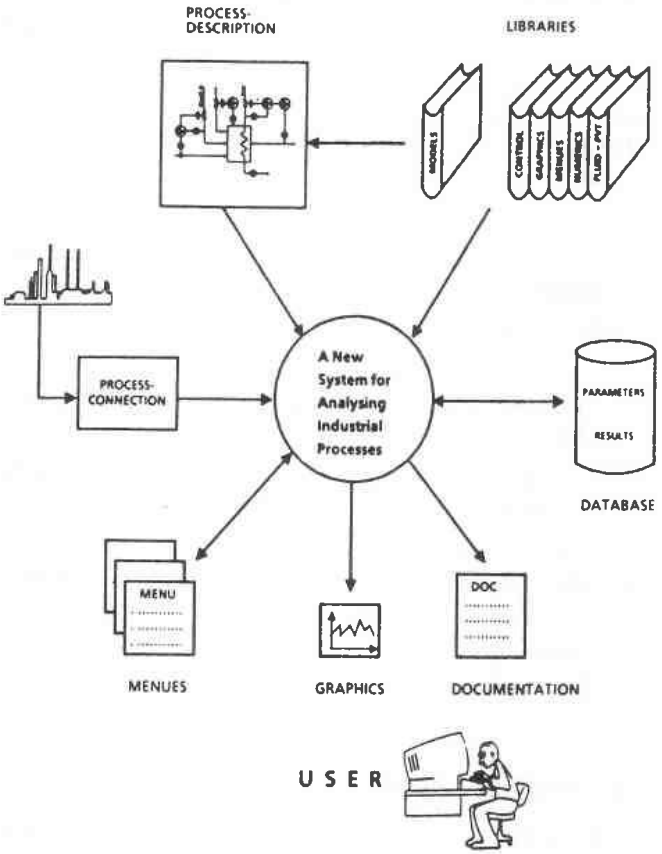

Figure 2. A new system for analysing industrial processes.

The models are classified according to the kind of boundary conditions required. The strategy for connecting unit models is based on this classification. Connectivity and process variables at boundaries between units are exchanged through a common database.

Control system configuration is also menu-driven, and the user may choose among the following items in the control system library:

SISO PID-Controllers
SISO discrete PID-Controllers
AD/DA-Converters
Noise generators
Filters

As indicated, discrete or sample data systems may as well be treated. This has necessitated a system for discrete event handling. Such a system has been included, which allows both explicit (time-dependent) and implicit (state-dependent) discrete events.

The system at present contains a controller documentation part, and a general documentation system is planned.

As illustrated in Fig. 2, the simulator has an interface to a process data base. Such an interface is planned for the Asea Brown Boveri's Masterpiece/Superview System, to be used at STATOIL's polymer reactors in Bamble. In this way, realistic on-line plant data

may be utilized in the system, either as boundary conditions, or/and as reference solutions.

The most important part of the system, is the the analysis part. It is possible to perform both dynamic and steady state simulation, and eigenvalue analysis. In the dynamic simulation part it is possible to set error tolerances and to choose among the different numerical algorithms. The following integration algorithms are available:

Adams Bashforth, order 1–4, fixed step
Adams Bashforth/Moulton (PC), variable order (LSODE)
BDF, variable order (LSODE)
BDF, variable order, sparse matrix handling (LSODES)
Runge–Kutta–Fehlberg-45
Runge–Kutta Prince/Dormand-87
Variable method, variable order, with root finding capabilities (LSODAR)

After a simulation, it is possible to

store results (for later presentation/comparison)
plot result (both on screen and HP-plotter)
continue/restart simulation, or any other analysis task

For steady state solutions, the MINPACK-solver HYBRD1 (Powell-hybrid method) has been adopted. This has shown to be a robust method for solving nonlinear algebraic equations. The method uses a combination of Newton–Raphson and scaled gradient techniques.

As demonstrated, it is important both for control purposes and for choosing the most appropriate numerical algorithm, to know the eigenvalues of the model. Therefore, routines from the EISPACK library has been adopted. Together with a linearization routine, this allows the user to evaluate model eigenvalues at any time.

There is also a possibility of 'freezing' the states of any unit model, for the purpose of analysing parts only of the entire system model.

4. Conclusions

The importance of using the most appropriate numerical algorithms is emphasized through examples from the petrochemical industry. It is found that integrating the dynamic model to steady state in some cases may be more efficient than solving the steady state model directly. Convergence problems are then avoided. It is demonstrated how state of the art numerical algorithms may be a basis for a powerful environment for process simulation.

Acknowledgments

We are grateful to Den norske stats oljeselskap a.s. (STATOIL), for financing the projects, from which this paper has originated.

References

Divakaruni, S. M. (1985). Perspectives in software design for dynamic process simulation. *Modeling, Identification and Control*, **6**, 217–229.

Hindmarsh, A. C. (1983). ODEPACK, a systematized collection of ODE solvers, in *Scientific Computing*, vol. 1 of *IMACS Transactions on Scientific Computation*, (North-Holland, Amsterdam), pp. 55–64.

Perkins, J. D. (1986). Survey of existing systems for the dynamic simulation of industrial processes. *Modeling, Identification and Control*, **7**, 71–81.