Korean J. Chem. Eng., **15**(3), 231-242 (1998)
REVIEW

FRAMEWORK OF DYNAMIC SIMULATION FOR COMPLEX CHEMICAL PROCESSES

Min Oh† and Il Moon*

LG Engineering Co. Ltd., Seoul
*Department of Chemical Engineering, Yonsei University, Seoul
(Received 24 June 1997 • accepted 4 April 1998)

Abstract – The current state of the art of a modelling and dynamic simulation system for complex chemical and biochemical processes is discussed. Process modelling activity involves modelling a physical plant and external tasks imposed on the plant, and details of both aspects are discussed. Typical software structure is concerned with a model builder, result analyser, translator, solution methods, model library and external software interface. Some of them are explained in moderate depth. Recent progress of functionality and numerical methods is presented. Numerical methods incorporating symbolic and structural techniques improve accuracy and efficiency. In order to illustrate benefits of employing dynamic simulation tools, one typical chemical process consisting of a mixing tank, tubular reactor and gas absorber is chosen and dynamic simulation is carried out. Taking into account the work in this paper, some suggestions for future development of a unified framework of a modelling package are made.

Key words: Process Modelling Tool, Dynamic Simulation, Discontinuity, Distributed Process, Numerical Methods, Software Structure

INTRODUCTION

Dynamic simulation is the activity of analysing and predicting the time transient behaviour of the physical process of interest that is usually described in terms of mathematical equations. In the chemical process industry, dynamic simulation serves an important role from the early stages of process design to plant commissioning and operation.

For example, in the design of control systems of a given chemical process with the collaboration of process and control engineers, many problems arise since chemical engineers are usually concerned with design issues reflecting steady state conditions. On the other hand, control engineers consider dynamic behaviour that is usually based on rough and qualitative knowledge. This might lead to an improper design of control systems. Even for the case of revamping the control system performing unsatisfactorily, it is not surprising that the system is modified according to the observations made in the startup or various state of operations. This procedure is no more acceptable if the decision on a certain process concept hinges on the outcome of the control. From this example, we conclude that the chemical engineer should have a better understanding of the dynamic response of the system of interest for better design and many other applications. In general, applications of dynamic simulation include the synthesis and analysis of chemical process control structure, startup and shutdown of the plant, safety assessments, optimisation and operator training facilities [Wozny and Jeromin, 1994].

In order to achieve a certain goal from dynamic simulation, two activities are mainly involved: building a well-posed mathematical model and a solution method to deal with it. However, this task is in many cases quite difficult or almost impossible for a non-expert and if not the case, very time consuming. Process modelling tools provide a user with a high level declarative language to build mathematical models and support reliable solution methods associated with them. The range of what is both desirable and practically feasible in process modelling has been expanding significantly in recent years. This trend is partly due to the realisation of the potential benefits of increased modelling realism and partly to rapid advances in computer hardware and numerical software.

Considering the increasing importance of dynamic simulation and tools to perform such activities, this paper discusses some of the significant issues concerned with such modelling tools for dynamic simulation. This is then followed by a brief description of the underlying numerical methods and relevant techniques such as symbolic and structural manipulation. In order to illustrate the capability and flexibility of modelling packages for dynamic simulation, one typical chemical process consisting of a mixing tank, tubular reaction and gas absorber is chosen and dynamic simulation is performed. Finally, based on the discussion made here, some suggestions for future development are given.

PROCESS MODELLING OF CHEMICAL PROCESSES

Mathematical modelling describes a given process in terms of mathematical expressions. This activity is usually involved with conservation rules, such as mass and energy balances, and other relations regarding physical properties and connectivity of dif-

†To whom all correspondence should be addressed.
E-mail: i28019@lgen.co.kr

ferent unit operations. It is also accompanied by intrinsic discontinuous behaviour of a physical system. This, in fact, defines the activity of process modelling in a traditional sense. However, the dynamic response of a physical system is decided not only by the intrinsic behaviour of a physical system, but also by external actions imposed by a controller or an operator. It is therefore concluded that process modelling for dynamic simulation should include both a physical system and external action associated with the operation of the process.

1. Modelling a Physical System

It has long been recognised that a natural mathematical description of the transient behaviour of lumped parameter processes is in terms of mixed systems of ordinary differential and algebraic equations (DAEs) [Marquardt, 1992]. Examples include a perfect mixing tank, continuous stirred tank reactor, tray columns etc. It is a fact that lumped parameter processes result from over-simplification of a given system. For instance, for the sake of simplification, a perfect mixing condition is usually introduced to model a mixing tank. It is, however, obvious that the behaviour of a mixing tank is far from perfect mixing, especially for an industrial application. Instead, the mathematical description of such a process in terms of a dispersion model shows more accurate behaviour of the system.

Apart from lumped parameter processes, a significant number of unit operations in chemical and biochemical processes take place in distributed parameter systems in which properties vary with respect to one or more space dimensions as well as time. Examples include packed bed tubular reactors, packed bed absorption, adsorption and packed distillation column etc. In other types of unit operations, some of the properties of the material are characterised by probability density functions instead of single scalar values. Examples include crystallisation units [Pantelides and Oh, 1996] and polymerisation reactors, in which the size of the crystals and the length of the polymer chains, respectively, are described in terms of distribution functions. The form of the latter may also vary with both time and spatial position. In fact, most complex processes typically involve a combination of both distributed and lumped parameter unit operations. The mathematical description of distributed unit operation models usually involves partial differential equations (PDEs) expressing the physical laws of conservation of mass, energy and momentum. In addition to these, models may involve algebraic relations that characterise phenomena, such as phase equilibria, which operate on much smaller time scales than those described by PDEs. Algebraic equations (AEs) may also be used to express relationships between variables, such as the definition of enthalpies in terms of temperature, pressure and composition. Finally, population balances [Ramkrishna, 1985] carried out on systems involving properties characterised by probability distributions very often lead to the introduction of integral terms in some of the equations. Overall, then, we are faced with mixed systems of integral, partial differential and algebraic equations (IPDAEs).

The effects of uncertainty on process design and operation have been receiving substantial attention in recent years, and effective techniques for managing it are beginning to emerge [Grossmann and Straub, 1991]. A stochastic description of a physical system can arise if model parameters of a deterministic model are regarded as stochastically distributed. This leads to the inclusion of distribution in engineering models and can be applied for bubble column, particle size distribution in a solid handling process, crystallisation and catalytic activity, deactivation of catalytic sites etc. For transport processes in multiphase systems a stochastically based model, using a technique called spatial averaging or volume averaging, can provide new insights into the origin of the constitutive equations for fluxes and into the dependence of transport coefficients. This can result in a better correlation of data [Hofmann, 1988]. In batch processes, stochastic variability typically arises from small variations in initial conditions (e.g., feed stock composition and temperature) and operating procedures, as well as from equipment failures and other unexpected reductions in resource availability and noise in the measurements used for monitoring and control purpose [Watzdorf et al., 1994]. Beyond the purely technical level, uncertainty is also introduced by the unpredicted nature of the production demands imposed on a given multipurpose batch plant. All such stochastic behaviour can usually be modelled using a certain number of probability distributions; e.g. uniform, triangular, normal, etc. [Kampen, 1981].

Complexity increases when a mathematical description is coupled with intrinsic discontinuity of a physical system. Although in their simplest form, models for unit operations are described in terms of continuous operations, many such models also involve one or more discontinuities. These typically arise from thermodynamic (e.g. phase) or flow (e.g. from laminar to turbulent regime) transitions, or from irregularities in the geometry of process vessels (e.g. overflow pipes or weirs) [Barton and Pantelides, 1994].

2. Modelling Operations

The procedure employed for the operation of process plants has traditionally been considered to be outside the scope of process modelling, being perhaps more relevant to real-time control systems which provide facilities for expressing and implementing them. Considering that one of the main purposes of simulation activities is to achieve desirable process objectives, a plant and an operating strategy must be considered as two equally important facets of process modelling. Along with the intrinsic discrete characteristics of a processing system, most chemical unit operations experience external actions that lead to an introduction of discrete events. From the viewpoint of process modelling, an operating procedure comprises a set of actions that effect certain discrete changes in the underlying model of the physical behaviour. For instance, many computer control actions do indeed correspond to discrete changes in the values of the input variables. Further complications are introducing a quantity of a certain reactant in a reactor, or resetting the integral error of a proportional/integral controller to zero. More severe discrete actions that are faced in conventional processes are probably startup and shutdown procedures of the plant. During startup and shutdown, parallel or sequential unit operations take place in the process; this eventually causes frequent discrete events in the process. Another example of experiencing frequent discrete events is periodic processes such as pressure swing adsorption and thermal

swing adsorptive reaction [Oh and Jang, 1998; Oh et al., 1998]. In pressure swing adsorption processes, the pressure at the feed end and product end change drastically according the sequence of cycling steps. Other variables that are dependent on the pressure change accordingly. Of course, certain actions in an operating procedure will be performed only under certain circumstances (e.g. emergency handling). Overall, the model of the operating procedure must also determine whether each action actually takes place and its precise timing. Detailed dynamic modelling and simulation of a batch operation is more complicated than a conventional continuous process because of the need to model not only the physical behaviour of individual units, but also the complex sets of discrete control actions that are imposed on them, and the equally complex logic involved in co-ordinating their operation [Park et al., 1996]. The degree of discontinuity increases in the following order:

a. purely continuous process
b. continuous process with intrinsic discrete behaviour of a process
c. continuous process with digital control
d. startup and shutdown of a continuous process
e. periodic process
f. batch process

One of the prime motivations for developing dynamic models and using dynamic simulation is the accurate analysis of the effects of external control actions and disturbances imposed on the physical system; therefore, modelling external forces should be an integral part of the modelling activities of a physical system for dynamic simulation.

PROCESS MODELLING TOOLS IN CHEMICAL ENGINEERING

As we have discussed, the mathematical description of a chemical process is a mixed set of IPDAEs. Furthermore, relevant discrete events, such as digital control actions and start-up and shutdown procedures, make modelling and dynamic simulation very complicated. In addition, the sheer size of mathematical equations (usually tens of thousands) resulting from process modelling and high non-linearity is another source of difficulty in performing dynamic simulation successfully. From the discussion, it is clear that constructing a complex mathematical model from scratch and performing dynamic simulation using in-house solution methods is almost an unattainable task; and even if this is the case, a substantial amount of time is necessary.

In this respect, dynamic simulation tools enable a user to tackle such complicated problems without any deep knowledge of numerical methods and computer technology. This section contributes to the software structure of such a simulation tool and the progress it has made up to now.

1. Software Structure

A software package for dynamic simulation is comprised of a number of elements. Fig. 1 illustrates the typical software structure of such packages. It should be understood that the details of the software structure of a dynamic simulation tool may be different between developers; Fig. 1 presents a representative basic scheme of such software.

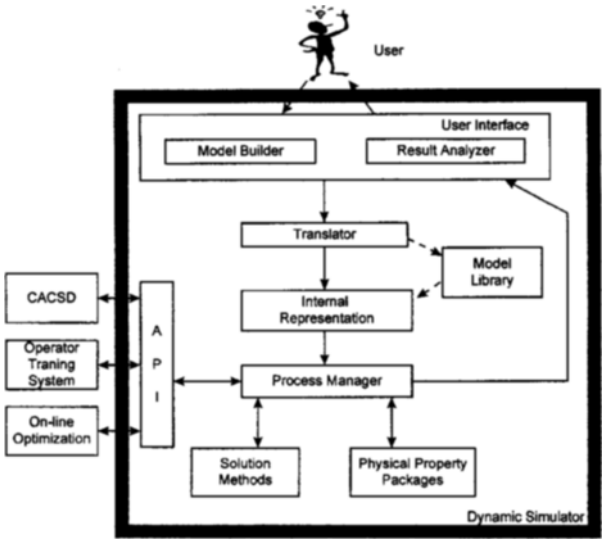

Fig. 1. Software structure of a dynamic simulation package.

1-1. Model Builder

The user interface consists of a model builder and a result analyser. The model builder provides a user with the means to map physical processes into mathematical equations. The mathematical model of a given process can mainly be constructed by a collection of built-in models from the library of a simulation package. Many commercial dynamic simulation packages (e.g. *HYSYS* [Hyprotech, 1995]) adopt the first approach and usually provide a user-friendly graphical editor to perform the modelling procedure rather easily. A mathematical model of complicated processes constructed in this manner allows well-posedness and, coupled with appropriate numerical methods, a user can easily obtain satisfactory results. However, it is argued that even for the same process, a mathematical description of a dynamic model can be different according to their applications. For example, a dynamic model for design purposes is usually much more detailed than for operator training purposes. Building a library to encompass all such applications is in fact very time consuming or even impossible.

For the second approach, a high-level declarative modelling language is utilised during the modelling procedure. Since a user is totally responsible for building a mathematical model based on conservation laws, reaction kinetics, phase equilibria, etc.; the well-posedness of the resulting mathematical model should be carefully checked. Despite the fact that a user (especially non-expert user) sometimes faces difficulties in constructing a mathematical model, this approach guarantees great flexibility, which allows research on the analysis and development of new processes and various applications. This flexibility is, however, obtained at the expense of a more complicated software structure and a very reliable and efficient solution code is necessitated. *Speedup* [Aspen Technology, 1994] can be regarded as a typical example to apply such a methodology. Some efforts are made to overcome the drawbacks of such approaches (i.e., inflexibility from the former and difficulty to construct a well-posed model). The *DIVA* modelling

package [Kroner et al., 1990] employs a hybrid model builder supporting two levels called "simulation expert" and "modelling expert". Through the simulation expert user interface, the simulation expert applies the graphical simulation model editor to set up the flowsheet and draw the topology of the plant. From a list of graphical representations of process units contained in the model library the user selects the desired process. In the modelling expert, users build their own mathematical models, problem definitions and simulation data. They are stored in a database and used on their own or invoked through the simulation expert.

1-2. Result Analyser

The dynamic simulation of a large system over a long period of time produces large amounts of result data. The task of actually collecting and physically archiving results produced by a complex simulation can be quite complex and computationally demanding. The results can either be stored in the form of ASCII files or displayed to the user during simulation or after it. When a complex simulation over a long time period is executed, the size of the ASCII file is enormous, and the time needed to write data into the file is not negligible. Taking these considerations into account, all data from the simulation is stored in a binary file, in which the size and computational demand is comparably reduced. This can also be converted into ASCII code as required. Visualisation in terms of 2D or 3D is inevitable for complicated dynamic simulation since it is unthinkable to analyse huge results only by means of ASCII data. Off-line visualisation (after the simulation) is a normal practice, but run-time graphical demonstration is a desirable aspect for many other applications such as operator training systems and on-line monitoring. In many cases, the analysis of the result of a simulation in order to understand the behaviour of a physical system involves only a number of variables. For instance, in order to characterise a catalytic reactor system, data for the reactor temperature and concentration are often sufficient. For a large system, for which the time wasted to deal with unnecessary data (e.g. storage, retrieving) is not negligible, selective monitoring and saving of variables or time duration of interest is advisable.

1-3. Translator

The structure and functionality of a translator is heavily influenced by the software structure of a modelling system. In the case of building mathematical models in terms of a combination of built-in library, the analysis of the model is omitted and only the parameters supplied by a user are checked and saved accordingly. However, when a user is fully responsible for constructing a mathematical model, the translator performs two major tasks : analysis and validation (whether the input is legal, meaningful and semantically correct) of the input file containing the description of the process model as well as a simulation problem coded in a given user interface; and the generation of an internal model representation [Fisher and LeBlanc, 1988].

For the latter application, internal model representation is usually described in terms of a high level programming language such as Fortran (e.g. *Speedup* [Aspen Technology, 1994]) or C, which is then compiled and linked with numerical methods and physical property routines. In searching for the benefits of current developments of computer technology, the analysed input is sometimes translated into abstract data types such as binary trees, linked lists, etc. Unlike the former, this approach makes much use of modern technology in computer science and permits a very fast and efficient process during modelling and simulation activities.

1-4. Solution Methods

Since a general mathematical description of a complex chemical plant is a mixed set of IPDAEs accompanied by considerable discontinuity, a sophisticated dynamic simulation package should include reliable, and at the same time, efficient numerical solution codes to tackle such problems.

Rigorous mathematical equations arising in chemical processes usually include conservation laws, reaction and adsorption equations, phase equilibria and connectivity, since reaction and adsorption contain exponential terms which lead to highly non-linear systems. Numerical solution codes incorporating symbolic and structural techniques are very demanding. The number of equations of a practical system reaches tens of thousands and the matrix representing such system shows a very sparse pattern. Exploiting sparse techniques [Duff, 1980] is very beneficial in saving computational time and hardware resources. Considering that the role of numerical solution methods is of paramount significance, we discuss this issue in a separate section (section 4. Solution Methods) in more detail.

1-5. Interface to External Software

As will be discussed later, model-based dynamic simulation can be applied to various fields in process systems engineering. Dynamic optimisation, operator training system, computer aided control system design and on-line optimisation can be regarded as typical examples of the application of dynamic simulation.

In order to make those activities possible, simulation packages should be equipped with the functionality to communicate data to and fro between the simulation package and other application software. For instance, an operator training system does not necessarily involve numerical solvers or a model builder. Instead, a mathematical model concerning the process of interest is carried out in a simulation package and at every reporting time, simulation results are passed to the core engine of the operator training system [Cho et al., 1996]. And then the transferred data are analysed and displayed through the user interface of the operator training system. When intervention from a user is required (e.g. changing some operating conditions), the operator training system sends the simulation package a message to interrupt the present simulation task. Simulation resumes, reflecting new data from the operator training system. For the sake of reliable application, this activity should be performed in real time and provide a means to synchronise both processes.

2. Recent Progress

There has been significant progress in process modelling tools for simulation during the last decade. It is now a common practice to use simulation tools for design, revamping, predicting dynamic response and determining optimal operating conditions. Extensive review of software packages to support such tasks is given by Marquardt [1992] and Wozny and

Jeromin [1994]. Because it is not the aim of this article to review the functionality and usage of individual packages, the paper concentrates on technical breakthroughs in the field of modelling package developments.

2-1. Distributed Process Modelling

Currently available general-purpose software for the modelling and simulation of chemical and biochemical processes is primarily intended for lumped parameter systems [Marquardt, 1992]. Several packages, such as *Speedup* [Aspen Technology, 1994], *DIVA* [Kroner et al., 1990], *ASCEND* [Piela et al., 1991] and *OMOLA* [Anderson, 1991; Mattsson and Anderson, 1992], provide high level declarative modelling languages that allow mathematical statements of the transient behaviour of individual unit operations in terms of a mixed set of DAEs. In conjunction with efficiently dealing with the modelling activity of complex chemical processes, some packages adopt an object-oriented paradigm which is gaining increased popularity.

However, in such modelling packages, distributed processes are currently modelled by manual discretisation of the distributed dimensions, which reduces the mathematical system to a set of DAE with respect to time. This process is both difficult and error-prone, especially when advanced discretisation techniques are to be applied, and a high level of mathematical knowledge and skill is required to perform it in a satisfactory manner. Along with increasing requirements from the chemical industry to pursue more rigorous modelling, much attention has been focused on direct modelling of distributed processes in recent years. Two important issues are considered: the formalisms for the construction of distributed models in the context of modelling language and the underlying numerical solution methods for resulting mathematical equations. Since the numerical methods for distributed processing units will be discussed in section 4, only the formalism is considered in this part.

The variation of the conditions of a distributed parameter system may be described as distribution over one or more space dimensions, molecular weight, particle size, etc. The conditions within a model are characterised by variables and equations, some of which will be distributed over given domains. It should, however, be recognised that different variables and equations within the same model may have different degrees of distribution. They are three underlying key elements that are indispensable for declaring distributed parameter systems in a modelling language. Formal mechanisms that enable those concepts to be described in the language should be provided. The other issue is the formal syntax related with the introduction of partial differentiation and integration operators. Automatic solution methods should be involved for a user to solve the problem without any extensive knowledge of advanced numerical mathematics. Despite direct distributed process modelling and automatic solution procedure being of paramount importance, to our best knowledge, only a very limited number of modelling packages are equipped with the concepts mentioned above. One such modelling package permitting direct modelling of distributed processes is the *gPROMS* package [Oh, 1995; Oh and Pantelides, 1996] in which variation is distributed over multi-dimensional fields can be constructed and numerical methods for IPDAEs are automatically employed. The application of the package will be demonstrated in section 5.

2-2. Handling Complexity

Due to the complexity of chemical processes, it is often difficult to handle the whole modelling task simultaneously. The re-usability of any models that are developed and proven at substantial cost is also of significance. Considering these issues, hierarchical mechanisms that enable the user to construct a complex model from simpler components are introduced. The basic principle of these mechanisms is to repeatedly sub-divide the modelling problem of interest until a sufficiently simple level of the model is reached. In the chemical industry, such models usually correspond to elementary equipment items for unit operations (e.g. pump, valve) or parts thereof (e.g. distillation column trays, column overhead system). Because of its complexity, the activity of modelling a chemical processing system usually proceeds in three steps:

a. partitioning of the system into elementary sub-systems
b. analysing each sub-system
c. synthesising the system from the analysed sub-systems by means of connectivity

Re-usability can also be enhanced through the use of inheritance. The concept of inheritance was first popularised by the object-oriented programming language. Through inheritance, a new model may be declared as an extension or restriction of one or more previously declared models. A model that is directly descended from another model contains all the information associated with the parent, plus any new information declared within the model itself. In connection with inheritance hierarchy, models may therefore be developed in a hierarchical manner through a series of intermediate stages of increasing complexity. It is also a powerful tool for avoiding the repetition of common information during model development. Careful development of an inheritance hierarchy will ensure that information common to several models need only be specified once. In addition, if this common information is declared correctly in the first place, the possibility of errors occurring during the repeated specification of the same information is eliminated.

2-3. Extend Functionality

As discussed in section 3.1, there are many applications from dynamic simulation. The model-based approach has gained more and more popularity in process systems engineering since other applications can benefit from a transparent model. Optimisation (either for steady state or dynamic state) is the most well known example for this. Even if it should be possible to interface a dynamic simulator with an external software package for optimisation, it is certainly advantageous to carry out two activities within the same framework. Interfacing heterogeneous software packages (in this case dynamic simulator and optimiser) in terms of data transfer causes the loss of much important information such as of a symbolic Jacobian matrix, occurrence matrix, block decomposition information, etc. This is particularly true when detailed information is necessitated for reliable and efficient optimisation. Within the same framework, dynamic optimisation utilises the model and the results developed during dynamic simulation task. In addition,

objective function and system constraints are coded using description language prepared by the modelling system. Many commercial modelling tools support steady state optimisation within the same framework of simulation. Examples include *Speedup* [Aspen Technology, 1994], *PRO/II* [Simulation Science, 1995], *HYSYS* [Hyprotech, 1995], etc. Dynamic optimisation in this context is regarded as in its early stage. Only a limited number of packages support dynamic optimisation utilising dynamic simulation.

2-4. General Purpose Simulation Packages

DIVA has been developed at the University of Stuttgart based on the concepts for dynamic simulation in the domain of chemical engineering. A block-oriented flowsheet representation is used to describe the topology of a plant. Single process units are interconnected by flow of energy, mass and information. The topological structure and the model equations of all process units, chosen from libraries containing model equations and property correlation, s are combined by the plant model process to set up the equation system of the plant. Numerical algorithms enhanced by sparse matrix techniques simultaneously solve the DAE systems. A knowledge-based user interface for interactive problem definition and controlling the simulation has been implemented.

Speedup package allows specification of steady state simulation, steady state optimisation and dynamic simulation in a unified language specially designed for process engineering applications. Through the high level declarative language, mathematical models are described in terms of sets of variables and the ordinary differential and algebraic equations that relate them. Models of more complex unit operations (such as distillation columns), called "macros", may be formed from instances of the basic models. Finally, a model of the entire plant may be formed by combining instances of both models and macros into a flowsheet. Solution of the underlying mathematical problem is achieved through the use of powerful techniques of symbolic and numerical computation. The symbolic information demonstrates significant improvements in robustness and efficiency over algorithms relying purely on numerical information. *Speedup* offers an interactive environment for process flowsheeting, in which the user can easily create, store, retrieve and modify one or more problems, all of which are stored simultaneously in a specially designed database file. It also handles a user's requests for help and diagnostic information, the display, storage and retrieval of simulation results, and other accounting and house keeping tasks.

OMOLA builds on the hierarchical sub-model decomposition with the introduction of an object oriented modelling framework. An important feature is the ability to use inheritance in the declaration of both model types and complex connection mechanisms. The design of *OMOLA* pays particular attention to the issue of model parameterisation. The use of parameters extends the model type concept by enabling a model type to describe the behaviour of a wide range of similar, albeit not identical, components. The values assigned to the parameters of an individual model instance then customise it to its application. *OMOLA* also introduces the representation of model behaviour as a number of different mathematical realisations (such as DAEs, transfer functions, and state space descriptions), rather than a single realisation [Barton, 1992]. The application of the concepts embodied in the language has been demonstrated through the development of the continuous model of a complete chemical process [Nilsson, 1989].

ASCEND is a language for the declaration of continuous mathematical models, particularly models of chemical processes. The framework of software is deeply influenced by the object-oriented paradigm. A complex model type is constructed from primitive types using a range of language operators such as IS_REFINED_TO, ARE_LIKE and ARE_THE_SAME. The model eventually developed should represent a well-posed mathematical problem that can then be submitted for solution to a suitable numerical method. Both hierarchical sub-model decomposition and model inheritance are supported by language operators.

MODEL.LA [Stephanopoulos et al., 1990a,b] is presented as a language suitable for the description of models to be used for the entire range of process engineering activities. The language is fully object-oriented and hierarchical sub-model decomposition is represented in five levels of abstraction: plant, plant-section, augmented unit, unit and sub-unit. The language has been integrated with the *DESIGN-KIT* package [Stephanopoulos et al., 1987], an object-oriented environment for computer-aided process engineering. An important feature of *MODEL.LA* that distinguishes it from the other languages is the manner in which models of individual unit operations are declared. All the other languages require a basic unit operation model to be declared in terms of mathematical relationships between system variables. In contrast, *MODEL.LA* only requires a declaration of the relationships between system variables (such as mass or energy balances) and a set of assumptions concerning physical and chemical phenomena. The model executive can then automatically generate the correct mathematical relationships from the information. This approach has many advantages, including rigorous model documentation and consistency checking, and greater support for the inexperienced modeller, but may ultimately be restricted by the scope of the knowledge base from which equations are automatically generated. In addition, *MODEL.LA* introduces a framework for multifaceted modelling. This recognises the need to consider a process model at several different levels of abstraction during the evolution of a design. A multifaceted model consists of an arbitrary number of facets that exchange and share information concerning the physical object under consideration. The facet used for a particular activity is determined by the level abstraction required [Barton, 1992].

SOLUTION METHODS

A numerical solution method is the single most important element of a modelling package. The numerical solution code employed by a modelling package should be able to deal with highly non-linear IPDAEs accompanied by discrete events. This section discusses some functionality and desirable aspects of a computer code to support the solution of such systems.

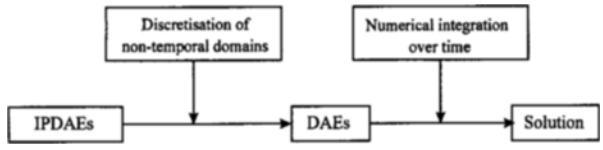

Fig. 2. Two phase solution procedure in the method of lines.

1. Numerical Methods for IPDAEs & DAEs

The solution of IPDAE systems is generally a difficult problem. Changing a parameter or one of the boundary conditions may lead to completely different behaviour from that originally anticipated [Ames, 1992]. Furthermore, although some numerical methods can accurately solve a given IPDAE system, other numerical methods may be totally unable to do so. One of the mostly widely used methods is the method of lines [Schiesser, 1991]. A brief idea of the method is to discretise a given IPDAE system with respect to spatial coordinates, which leads to DAEs. The family of the method of lines comprises collectively a number of finite difference, finite element [Zienkiewicz, 1983] and weighted residual methods [Finlayson, 1980] in which piece-wise local or global approximation functions in the space dimensions are used to convert evolutionary IPDAE problems into initial value DAE problems.

The advantage of this procedure is that sophisticated computer programs that permit fast and accurate integration of large sets of DAEs over time can be employed. In particular, integration codes based on backward differentiation formulae (e.g. DASSL [Petzold, 1982] and DASOLV [Jarvis and Pantelides, 1992]) can solve stiff as well as non-stiff systems of equations. They utilise sophisticated algorithms for automatic step-size adjustment and integration order selection to maintain a user-specified error tolerance. This whole procedure, using the method of lines followed by the application of DAE solver, is referred to as the two phases solution method and is depicted in Fig. 2.

Finally, initialisation and re-initialisation of a given system triggered by starting simulation activity and discrete events facilitates the solution method for non-linear algebraic equations (NAEs). Despite their simple appearance, the solution procedure for NAEs is probably one of the most difficult issues. This is particularly true at the very first initialisation step of time integration, for good estimates for the values of variables are not always available. Unfortunately, current technology of the solution method for NAEs does not guarantee the convergence of a given problem from very poor initial guesses. Employing a quasi-Newton type algorithm and a priori re-arrangement of the NAEs into block triangular form [Keeping, 1995], in many cases, improves reliability of the method.

2. Detecting and Handling Discontinuity

Except for some special cases, dynamic simulation of industrial processes is hardly continuous. In the chemical industry, all unit operations are involved with digital control or external actions imposed to it; consequently, the behaviour of the unit operations shows combined discrete and continuous characteristics. As already discussed, discontinuous behaviour of a process model arises from intrinsic characteristics of the process or external actions imposed on the physical plant. It is therefore imperative to detect and handle such discontinuities.

Regardless of the reason for discontinuity, it can be categorised into explicit and implicit discontinuity. In the former, the exact time of occurrence is known a priori. Suppose an operator wants to turn a feed pump on 500 seconds after startup; the exact location of this explicit event is given. However, the exact time for a phase transition from liquid to vapour in a distillation column is totally up to the state of a system, e.g. variation of temperature and pressure. One efficient way to seek for an exact location of this kind of problem is that at the end of every successful time integration, all conditions which cause the discontinuity are checked. When the condition is changed from previous integration step (for instance, the Reynolds number is changed from 1800 to 2500, then mathematical equations for fluid flow are also changed from laminar to turbulent), the integration step is reduced until a given tolerance is satisfied. This permits finding an exact location of discontinuity. The way of reducing the integration step determines the efficiency of the method.

The next issue is how to deal with discontinuity. Nowadays, the majority of numerical codes for time integration employ multi-step methods based on backward differentiation formulae [Gear, 1971]. This method involves some data calculated from previous integration and utilises them to obtain the present solution. However, at the point of discontinuity, all these data cannot be used for there is no clear relationship between the present and the past. A remedy for this problem is the re-initialisation of a given system at the location of the discontinuity. It is normal practice to assume values of system variables are continuous across the boundary of each integration step except those causing the discontinuity. However, discontinuity introduces new conditions and eliminates old conditions that are no more available for the system. Re-initialisation is now carried out with respect to a new system containing new conditions. This is then followed by time integration. From the nature of time integration and re-initialisation, frequent discrete events cause a very inefficient and inaccurate time integration (since we cannot use a multi-step method), and in the worst case, the numerical integration code multi-step methods cannot be used.

3. Symbolic and Structural Information

A modelling package for dynamic simulation employing a model-based approach usually handle tens of thousands of mathematical equations which combine stiff and non-stiff equations as well as highly non-linear equations. In spite of the recent progress in numerical mathematics, we experience many problems in tackling such a large system during a solution procedure. As identified by Pantelides and Barton [1993], the source of such problems includes the inherently bad-posed, or those that cannot be solved using currently available techniques. Problems, which belong to the first category, are structural singularity and numerical singularity, whereas high index belongs to the latter. The examples for the latter are the high-index problem and poor initial guesses. Some problems such as poor initial guesses and high-index problems are concerned

with numerical technology, but bad-posed and singularity problems can be detected before numerical methods are employed.

Structural information is a valuable means to detect the solvability of a given system. When the size of the matrix to represent the mathematical equations or its Jacobian matrix is n and the rank of the matrix is less than n, the matrix is referred to as a singular matrix. From its characteristics, a singularity can be categorised into numerical, structural and local singularity. Whereas numerical and local singularities should be tackled by numerical techniques, a structural singularity can be treated differently. If a matrix is indeed structurally singular, only a sparsity pattern of the Jacobian matrix of a given system needs to be considered. A structural singularity in large matrices can be detected very efficiently using graph-theoretical algorithms for output assignment, in particular one proposed by Duff [1980]. This methodology is also utilised to formulate triangular block decomposition to enhance numerical solution procedures and detect high-index problems.

Whereas structural information is concerned with the solvability of a given system, symbolic techniques mainly provide valuable information for solving a given problem efficiently during the numerical treatment phase. In dealing with mathematical equations such as NAEs, DAE and PDAEs, the Jacobian matrix of a given system should be solved. Especially, when time integration of a large system is involved, fast convergence of the Jacobian matrix is essential. The conventional practice of calculating a Jacobian matrix is to approximate it in terms of finite difference methods, and an approximate solution is then obtained. Because this calculation demands a great deal of computational time, a Jacobian matrix is not usually calculated until divergence is faced. When the problem does not converge, lately calculated approximate Jacobian is to be updated. This is error-prone and at the same time very time consuming. In this respect, exploiting symbolic treatment of the Jacobian matrix is very beneficial. Symbolic differentiation of a given system, which leads to a Jacobian matrix of the system, is executed and the result is utilised whenever required. During numerical treatment, for instance quasi-Newton methods, the exact Jacobian gained from symbolic differentiation is calculated. This usually guarantees better convergence as well as fast calculation.

ILLUSTRATIVE EXAMPLE

This article identifies the benefits of employing a process modelling tool for dynamic simulation of complex chemical processes. It permits

a. direct modelling of a complex physical plant including distributed processes,
b. modelling external actions including the change of operating conditions,
c. automatic solution methods for resulting mathematical descriptions.

In attempting to illustrate the benefits mentioned above, we introduce a process modelling tool, called *gPROMS* which is mainly for the modelling and dynamic simulation of combined lump/distributed parameter processes accompanied by either continuous or combined discrete/continuous events. As already discussed, sophisticated numerical methods for DAEs and IPDAEs are provided and automatically invoked whenever necessary.

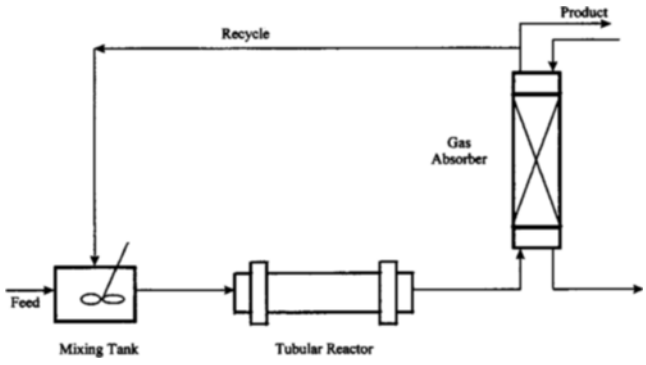

Fig. 3. Flowsheet of a mixer, tubular reactor and gas absorber.

The target process, shown in Fig. 3, comprises a well-stirred mixing tank, a tubular reactor and a gas absorption column which leads to a combined lump and distributed process [Heydweiller et al., 1977]. The mathematical description of the well-stirred mixing tank is DAEs, whereas mathematical equations of the tubular reactor and the gas absorption column are parabolic and hyperbolic PDEs, respectively. As a result, the system is described in terms of a mixed set of PDAEs. In conjunction with numerical solution procedures, modelling of such a process using a programming language is by no means a trivial task. The mathematical form of the resulting equations becomes different according to underlying approximation methods. In order to perform this task successfully, deep knowledge of numerical mathematics is essential. A high level declarative modelling language which allows direct modelling of lumped and distributed processes is provided and, in this case, a user's only responsibility is to describe the mathematical model using the language. The numerical method chosen by a user is automatically involved with the solution procedures.

In order to demonstrate the capability of handling combined continuous/discrete events, operating conditions is also changed: i.e., at 40 dimensionless time after startup, the concentration at the inlet of a mixer is increased by 20 %. This causes discontinuity to the process. Overall, simulation activity is associated with time integration of the combined lumped and distributed parameter process accompanied by combined discrete and continuous events.

The reactor carries out the gas-phase reaction

$$A + B \rightarrow 2C$$

The reactor product enters the bottom of the countercurrent absorption column where C is partially absorbed in the liquid phase. The remaining gas is recycled to the mixer where it is combined with fresh feed. The dynamic response of this system is determined by simultaneously solving the ordinary differential equations describing the mixer, the parabolic differential equations for the tubular reactor and the hyperbolic

differential equations for the gas absorption column.

These equations are shown below in dimensionless form. Because an isotherm process is assumed, only mass balance equations are considered here.

mixing tank:

$$\frac{d\phi_i^m}{d\tau} = K_8[\phi_i^f + K_9\psi_i^e - (1+K_9)\phi_i^m] \qquad i=A, B, C$$

where $\phi_i^m(\tau)$ is the dimensionless concentration of component i in the mixing tank, ϕ_i^f the dimensionless concentration of component i in the feed stream and ψ_i^e the dimensionless concentration i of component i in the recycle.

tubular reactor:

Mass balance:

$$\frac{\partial\phi_i}{\partial\tau} = K_1\frac{\partial^2\phi_i}{\partial\lambda^2} - \frac{\partial\phi_i}{\partial\lambda} + \nu_i K_2\phi_A\phi_B \qquad \forall\lambda\in(0, 1), i=A, B, C$$

Boundary conditions:

$$\frac{1}{Pe}\frac{\partial\phi_i}{\partial\lambda} = \phi_i - \phi_i^m \qquad @\lambda=0, i=A, B, C$$

$$\frac{\partial\phi_i}{\partial\lambda} = 0 \qquad @\lambda=1, i=A, B, C$$

where $\lambda\in[0, 1]$ is the dimensionless axial position and τ the dimensionless time, while $\phi_i(\lambda, \tau)$ represents the dimensionless concentration of component i in the reactor and ϕ_i^m the corresponding quantity in the reactor feed. The stoichiometric coefficients used in the mass balance are $\nu_A = -1$; $\nu_B = -1$; $\nu_C = +2$

gas absorber:

Mass balance:

$$\frac{\partial\varphi_A}{\partial\tau} = -K_3\frac{\partial\varphi_A}{\partial\zeta} \qquad \forall\zeta\in(0, 1]$$

$$\frac{\partial\varphi_B}{\partial\tau} = -K_3\frac{\partial\varphi_B}{\partial\zeta} \qquad \forall\zeta\in(0, 1]$$

$$\frac{\partial\varphi_C}{\partial\tau} = -K_3\frac{\partial\varphi_C}{\partial\zeta} - K_4(\varphi_C - K_5\theta_C) \qquad \forall\zeta\in(0, 1]$$

$$\frac{\partial\varphi_C}{\partial\tau} = -K_6\frac{\partial\theta_C}{\partial\zeta} - \frac{K_4}{K_7}(\varphi_C - K_5\theta_C) \qquad \forall\zeta\in[0, 1)$$

Boundary conditions:

$$\varphi_i = \phi_i^e \qquad @\zeta=0, i=A, B, C$$

$$\theta_C = 0 \qquad @\zeta=1, i=A, B, C$$

where $\zeta\in[0, 1]$ is the dimensionless axial position of gas absorber, $\varphi_i(\zeta, \tau)$ and $\theta_i(\zeta, \tau)$ the dimensionless concentrations in the gas and liquid phases, respectively and ϕ_i^e the dimensionless concentrations at the reactor exit.

Figs. 4-6 demonstrate the concentration of component A in the tubular reactor, gas absorber and mixing tank, respectively.

Transient behaviour of component A is shown in Fig. 4. A 20% increase in the concentration of component A in the feed occurs at 40 dimensionless time. The concentration of component A rapidly changed near this point. As time integration continues, the concentration of component A in the tubular reactor and gas absorber also changes (see Figs. 5-6).

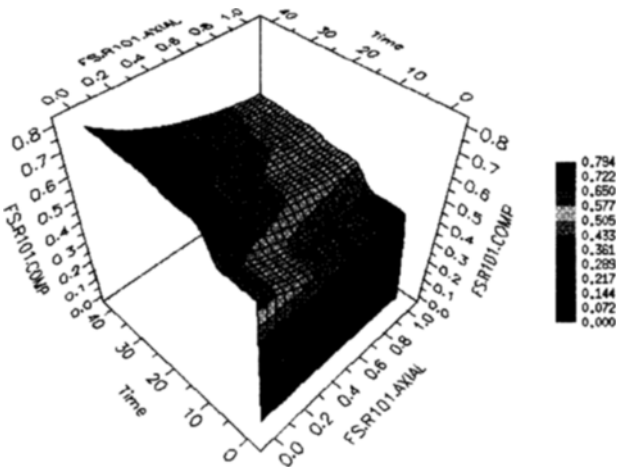

Fig. 4. Concentration profile of component A in reactor.

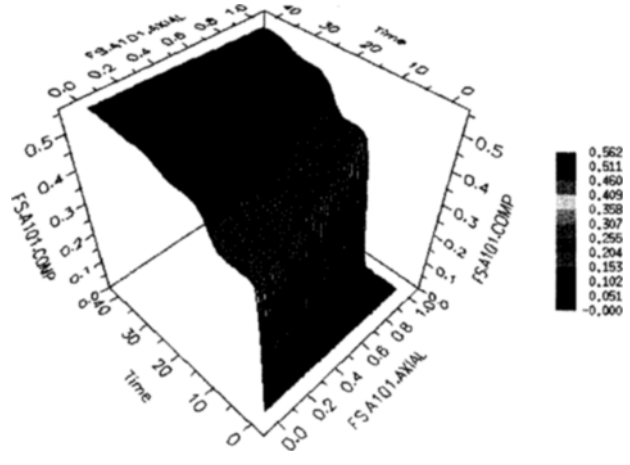

Fig. 5. Concentration profile of component A in gas absorber.

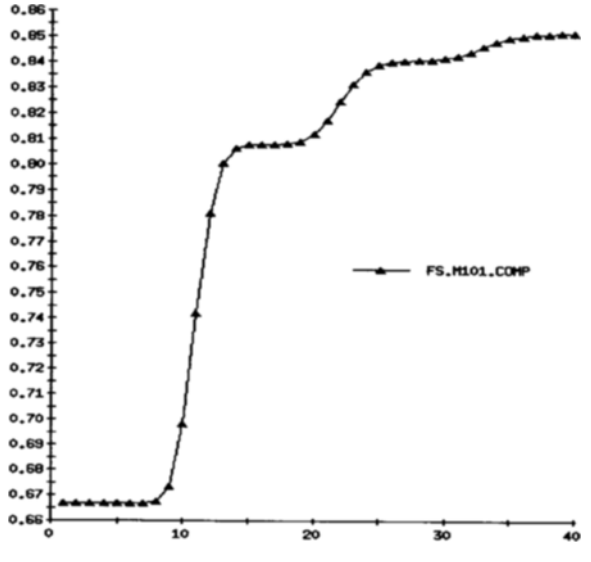

Fig. 6. Concentration profile of component A in mixing tank.

CONCLUDING REMARKS AND FUTURE DIRECTIONS

This paper reviews the current state of the art of modelling and dynamic simulation tools for complex chemical and biochemical processes. Process modelling of chemical processes includes both modelling a physical plant, which depicts intrinsic behaviours, and modelling operation, which imposes on the physical plant. Such modelling issues for a physical plant as well as operations are discussed. The software structure and basic elements of a dynamic simulation package are examined. It typically consists of a user interface, translator, solution methods, built-in mathematical model libraries and a result analysis system. Each element is examined in moderate depth. This is then followed by a consideration of the recent progress of a modelling tool. Main progress lies in distributed process modelling, handling complexities and extending functionality including steady state/dynamic optimisation. Numerical methods to deal with a complicated process are discussed in detail. Since there is no universal numerical method to tackle a wide spectrum of problems arising in chemical engineering, various numerical methods based on sound mathematical principles should be considered. One example, which illustrates the typical characteristics of chemical processes, is chosen and dynamic simulation is carried out using the *gPROMS* modelling package. This demonstrates the benefits of the utilisation of the modelling package to deal with complicated chemical processes.

Despite the fact that the recent progress of modelling tools allows a user to deal with a considerable number of complicated modelling and simulation problems, there are many outstanding problems to be resolved. Consequently, the final section concentrates on three areas which, we believe, are of a more strategic and fundamental nature.

1. Unified Framework

The possibility of performing various activities within the unified framework of a modelling tool is one of the main attractions of an equation-oriented approach. In the equation-oriented approach, the behaviour of a physical plant and its external actions are expressed by mathematical descriptions. Once a well-posed mathematical model is developed, it is applicable for various applications by merely adding some conditions or modifying a part of equations according to the type of applications.

One important application is the unified work of steady state and dynamic simulation. Mathematically speaking, steady state is regarded as one trivial case of a dynamic state. To be more specific, a dynamic state of a given system becomes a steady state of the system when all time derivatives of the dynamic model are eliminated. However, traditional practice performs steady state and dynamic simulation separately, facilitating different simulation software packages. Since there is no obvious relation between them in this practice, efforts and time made for steady state simulation are wasted and the same amount (or even more) of activity should be given for dynamic simulation. In the unified framework, the same mathematical model is used to describe both steady and dynamic state behaviour of a given system. During the course of a steady state simulation, the initial conditions of all time derivatives become zero and numerical integration of the system is not carried out. It is then followed by dynamic simulation with appropriate initial conditions of the same mathematical model within the unified framework of a modelling package.

Optimisation for both the steady state and dynamic state is an important ingredient of process design. General formulation of an optimisation problem consists of an objective function, equality and non-equality constraints [Fletcher, 1991; Edgar and Himmelblau, 1989]. The mathematical equations validated during the previous simulation phase serve as one of the equality constraints. The determination of the objective function and other constraints is subject to the purpose of optimisation and characteristics of a given process. When constraints and objective functions do not include time derivatives, it is defined as steady state optimisation; otherwise, dynamic optimisation [Biegler, 1984]. Optimisation gains more popularity in the design and optimal control area due to strong international competition and tighter environmental regulations.

An operator training system is another area that can benefit from the unified framework of an equation-based approach. An operator training system utilises the mathematical model and dynamic simulation facility in order to demonstrate dynamic responses to an operator as required. The details of a mathematical description of a target process are usually omitted for an operator training system. This is due to the fact that the operator training system aims to educate operators, and details of dynamic response are not a major concern. An operator training system inevitably incorporates a sophisticated user interface, which is very similar to actual working environments (e.g. monitoring system in digital control system).

Another possible application includes structural optimisation, which is described in terms of MINLP problems, parameter estimation, safety analysis. Fig. 7 depicts an equation-based approach in process systems engineering and its applications.

2. Well-posedness of PDAEs

Process modelling tools afford the user considerable flexibility both in model construction, and in the specification of degrees of freedom and initial conditions. The possibility therefore exists for the definition of problems which are either badly posed (in the sense that they do not posses a well-defined mathematical solution), or impossible or difficult to solve using the current state of solution techniques.

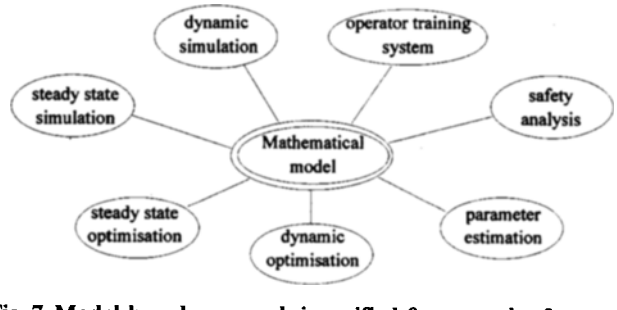

Fig. 7. Model based approach in unified framework of a modelling tool.

The detection and diagnosis of badly posed problems is mainly concerned with solvability, structural singularity, local singularity, high index and consistent initial conditions of a given system. Such a task exploits a sparsity pattern of the symbolic Jacobian or its variants of a given system. Partly due to sheer system size and partly to high non-linearity, it is already a difficult task to detect and diagnose well-posedness for systems of nonlinear algebraic equations and mixed ordinary differential and algebraic equations.

For IPDAE systems, this task is much more complex. One of the important numerical methods currently used is the method of lines, which discretises the variation over non-temporal domains and converts a given PDAEs into temporal DAEs. One of its main limitations is that they make no attempt at estimating and controlling the error incurred from such discretisation. Instead, they rely entirely on the user to select an appropriate method and grid. This sometimes introduces a significant discretisation error that is an order of magnitude bigger than that from time integration. Given the lack of a necessary mathematical framework, we probably have to rely mainly on analysis performed at the level of the DAE system resulting from discretisation, checking, for instance, the well-posedness of the system and its initial condition, and the index of the DAE system.

3. Result Storage

The powerful modelling capabilities of a modelling package for dynamic simulation together with ever increasing computational speeds imply that the software is used to build and study increasingly complex problems involving variations over relatively large numbers of dimensions. From this new situation, it is clear that much more needs to be achieved in reducing further the volume of results being stored without reducing the amount of information.

The collection and storing of results as a fixed frequency over long time horizons may be particularly wasteful. We note that the DAE integration packages used for solving the discretised IPDAE system utilise sophisticated step-length adjustment algorithms for traversing a given time horizon in the least number of steps. Moreover, it is possible to reconstruct the entire solution trajectory from the values of the variables at these steps together with the order of integration at each step. It may therefore be advantageous to store results only at the steps used internally by the integrator. This will have the desirable side-effect of shifting some of the computational overhead (that associated with interpolating the results of the integration to produce the values of the variables at the fixed reporting intervals) from the integrator to the results storing/visualisation system. The use of data compression techniques [see, for example, Hale and Sellars, 1981] is another possible means of reducing the volume of stored results.

NOMENCLATURE

K : model constants
A, B, C : components

Greek Letters

ϕ : dimensionless concentration in reactor and mixer
ψ : dimensionless concentration in gas phase of absorber
ζ : dimensionless axial coordinates
τ : dimensionless time
ν : stoichiometric coefficient in reaction
θ : dimensionless concentration in liquid phase of absorber

Subscript

i : component i

Superscripts

e : exit
f : feed
m : mixer

REFERENCES

Ames, W., "Numerical Methods for Partial Differential Equations", Academic Press, New York, 1992.

Anderson, M., "Omola-An Object Oriented Language for Model Representation", PhD Thesis, Lund Institute of Technology, 1990.

Aspen Technology, Inc. Speedup User Manual. Cambridge, Massachusetts, 1994.

Baton, P. I., "The Modelling and Simulation of Combined Discrete/Continuous Process", PhD Thesis, 1992.

Barton, P. I. and Pantelides, C. C., "The Modelling of Combined Discrete/Continuous Processes", *AIChE J.*, **40**, 966 (1994).

Biegler, L. T., "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation", *Computers Chem. Engng.*, **8**(3), 243 (1984).

Cho, S. I., Park, S. Y., Oh, M. and Moon, I., "Operation and Modularisation of Solvers in Operator Training System", *Theories and Applications of Chem. Eng.*, **2**(1), 117 (1996).

Edgar, T. F. and Himmelblau, D. M., "Optimization of Chemical Processes", McGraw-Hill, 1989.

Finlayson, B. A., "Nonlinear Analysis in Chemical Engineering", McGraw-Hill, New York, 1980.

Fisher, C. N. and LeBlanc, R. J., "Crafting a Compiler", Benjamin/Cummings, 1988.

Fletcher, R., "Practical Methods of Optimization", John Wiley & Sons, 1991.

Froment, G. F. and Bischoff., K. B., "Chemical Reactor Analysis and Design", John Wiley and Sons, New York, 1990.

Gear, C. W., "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, New Jersey, 1971.

Hale, J. C. and Sellars, H. L., "Historical Data Recording for Process Computers", *Chem. Engng Prog.*, November : 38 (1981).

Heydweiller, J. C., Sincovec, R. F. and Fan, L. T., "Dynamic Simulation of Chemical Processes Described by Distributed and Lumped Parameter Models", *Computers chem. Engng*, **1**, 125 (1977).

Jarvis, R. B. and Pantelides, C. C., "DASOLV-A Differential-Algebrai Equation Solver", Technical report, Centre for Process Systems Engineering, Imperial College, London, 1992.

van Kampen, N. G., "Stochastic Processes in Physics and Chemistry", North-Holland, New York, 1981.

Keeping, B. R., "Efficient Methods for the Solution of Large Systems of Differential-Algebraic Equations", PhD Thesis, University of London, 1995.

Kroner, A., Holl, P., Marquardt, W. and Gilles, E. D., "DIVA-An Open Architecture for Dynamic Simulation", *Computers chem. Engng*, **14**, 1289 (1990).

Grossmann, I. E. and Straub, D. A., "Recent Developments in the Evaluation and Optimization of Flexible Chemical Processes" (Puigjaner, L. and Espuna, A., eds), Computer-Oriented Process Engineering, Elsevier, Amsterdam, pp. 49, 1991.

Hofmann, H., "Future Trends in Chemical Engineering Modelling", *Computers chem. Engng*, **12**, 415 (1988).

Hyprotech, Steady-State and Dynamic Simulation Using HYSYS, 1995.

Marquardt, W., "Dynamic Process Simulation-Recent Progress and Future Challenges. In Chemical Process Control" (Arkun, Y. and Ray, W. H., eds.), Pergamon Press, pp. s329, 1992.

Mattsson, S. E. and Anderson, M., "OMOLA-An Object-Oriented Modelling Language", In M. Jamshidi and C. J. Herget, editors, Recent Advances in Computer Aided Control Systems Engineering, Elsevier, New York, 1992.

Nilssen, B., "Structured Modelling of Chemical Processes-An Object-Oriented Approach", PhD Thesis, Lund Institute of Technology, 1989.

Oh, M., "Modelling and Simulation of Combined Lumped and Distributed Processes", PhD Thesis, University of London, 1995.

Oh, M. and Moon, J. K., "Optimisation of Cycle Time and Feed Pressure on Rapid Pressure Swing Adsorption in Separation of N_2/O_2", *HWAHAK KONGHAK*, **36**(2), 196 (1998).

Oh, M., Jang. E. J. and Moon, J. K., "A Temperature Swing Adsorptive Reactor for the Enhancement of Catalytic Dehydrogenation Reaction", *HWAHAK KONGHAK*, **36**(1), 109 (1998).

Oh, M. and Pantelides, C. C., "A Modelling and Simulation Language for Combined Lumped and Distributed Parameter Systems", *Computers chem. Engng*, **20**, 611 (1996).

Pantelides, C. C. and Oh, M., "Process Modelling Tools and Their Application to Particulate Processes", *Powder Technology*, **87**, 13 (1996).

Park, S. Y., Oh, M. and Moon, I., "The Classification and Characteristics of Discrete Events Arising in Dynamic Simulation of Chemical Processes", *HWAHAK KONGHAK*, **34**(5), 585 (1996).

Petzold, L. R., "A Description of DASSL: A Differential/Algebraic System Solver", In Proceedings IMACS World Congress, Montreal, Canada, August : 65, 1982.

Piela, P. C., Epperly, T. G., Westerberg, K. M. and Westerberg, A. W., "ASCEND: An Object-Oriented Computer Environment for Modelling and Analysis: The Modelling Language", *Computers chem. Engng*, **15**, 53 (1991).

Ramkrishna, D., "The Status of Population Balances", *Rev. in Chem. Engng*, **3**, 49 (1985).

Schiesser, W. E., "The Numerical Method of Lines", Academic Press, New York, 1991.

Simulation Science Inc., *PRO/II* Application Briefs Manual, Simulation Science Inc., 1995.

Stephanopoulos, G., Johnston, J., Kriticos, T., Lakshmanan, R., Mavrovouniotis, M. and Siletti. C., "DESIGN-KIT: An Object Oriented Environment for Process Engineering", *Computers chem. Engng*, **11**, 655 (1987).

Stephanopoulos, G., Henning, G. and Leone, H., "MODEL. LA. A Modelling Language for Process Engineering-I. The Formal Framework", *Computers chem. Engng*, **14**, 813 (1990).

Stephanopoulos, G., Henning, G. and Leone, H., "MODEL. LA. A Modelling Language for Process Engineering-I. Multifaceted Modeling of Processing System", *Computers chem. Engng*, **14**, 847 (1990).

von Watzdorf, R., Naf, U. G., Barton, P. I. and Pantelides, C. C., "Deterministic and Stochastic Simulation of Batch/Semicontinuous Processes", *Computers chem. Engng*, **18S**, 343 (1994).

Wozny, G. and Jeromin, L., "Dynamic Process Simulation in Industry", *Intl. chem. Engng*, **34**, 159 (1994).

Zienkiewicz, O. C. and Morgan, K., "Finite Elements and Approximation", John Wiley & Sons, New York, 1983.