Korean J. of Chem. Eng., **13**(6), 549-553 (1996)

A DYNAMIC PROCESS SIMULATOR BASED UPON THE CLUSTER-MODULAR APPROACH

Kang Wook Lee, Gibaek Lee, Kang Ju Lee* and En Sup Yoon†

Dept. of Chemical Engineering, Seoul National University, Seoul 151-742, Korea
*AID Corporation, 3rd floor Seokwang Bldg., 1628-29, Seocho-dong, Seocho-ku, Seoul, Korea
(Received 1 December 1994 • accepted 21 September 1996)

Abstract – The objectives of this work are to present the dynamic simulation strategy based on cluster-modular approach and to develop a prototype simulator. In addition, methods for the improvement of computational efficiency and applicability are studied. A process can be decomposed into several clusters which consist of strongly coupled units depending upon the process dynamics or topology. The combined approach of simultaneous and sequential simulation based on the cluster structure is implemented within the developed dynamic process simulator, MOSA (Multi Objective Simulation Architecture). Dynamic simulation for a utility plant is presented as a case study in order to prove the efficiency and flexibility of MOSA.

Key words: Dynamic Simulation, Cluster, Dynamic Simulator, Utility Plant

INTRODUCTION

Dynamic simulation has been accepted as a powerful tool to predict unsteady state behavior of chemical processes and can be used in all phases of process engineering activities such as process design, operation, control and automation. In recent years, the world-wide process industries are now focusing their attention on dynamic simulation to improve the design and operation of plants, process control systems and process safety systems. Much of the work is concentrated on process modeling (structure, representation and tools), numerical techniques (higher-index problems, consistent initialization, etc.) and advanced simulation environment (software integration, user interface, etc.) [Marquardt, 1991].

Up to now, various approaches for solving the equations of the dynamic model of processes have been discussed and implemented. There are two distinctly different methods for dynamic simulation: equation-based and module-based approach. The integration strategy of a large set of mixed differential and algebraic equations is the most essential issue in dynamic process simulation. In fact, there are many possible computational strategies to solve the equations in a dynamic model. The advantages of the module-based approach are;

- ease of understanding and implementation,
- the maximum size of computational block is determined not by the total unit but by the largest unit,
- improvement of computational efficiency by parallelism of process.

The disadvantages of this approach are;

- poor convergence and efficiency when a process has strong interactions between unit processes,
- the need of synchronization or coordination when different integration methods are adopted to different modules.

†To whom all correspondences should be addressed.

The advantages of equation-based approach are;

- flexibility in problem definition,
- convergence not affected by recycle stream.

The disadvantages of this approach are;

- the need of large computer memory,
- the need of structural problem formulation,
- the need of advanced numerical methods.

It is desirable to combine the advantages of above two approaches. In this paper, a cluster-modular approach which has a simultaneous integration capability is presented and the general purpose dynamic process simulator, MOSA, is described.

CLUSTER-MODULAR APPROACH

The module-based approach can be classified into two types of simulation methods according to the role of a numerical integrator: simultaneous and independent modular method (some authors refer to these as "coupled mode" and "uncoupled mode"). In the former, a common integrator is used to integrate differential equations of all modules simultaneously, whereas in the latter, each module has its own integrator. Therefore, the former has the advantage of very accurate integration to take relatively large step size. However, it is unfavorable for the memory problem because it should handle all modules at the same time. Though the latter takes small step size because it has the drawback in accuracy, it is able to be applied to the large-scale processes.

The suggested method is in the middle of two methods. Its basic form of modules is identical with independent modular method and it solves the equations simultaneously as simultaneous modular method. In independent modular method, each module is the independent units for evaluation, but the proposed method uses "cluster", the new evaluation unit that is the combination of one or more modules. The process is shown in sequential arrangement of these clusters. As this method

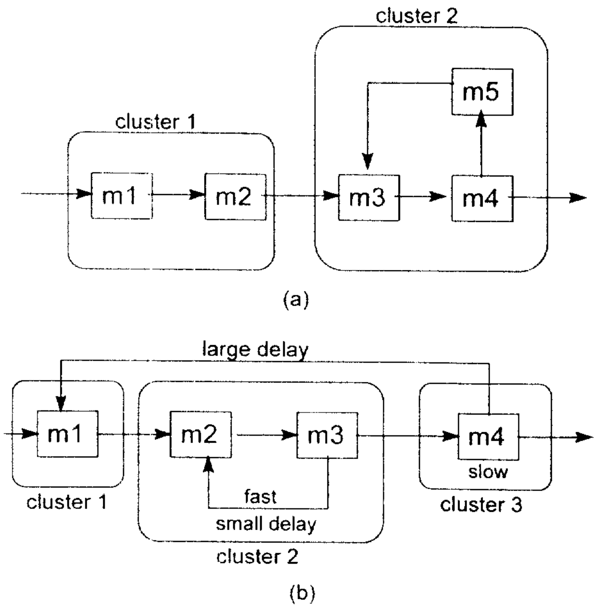

Fig. 1. Example of clustering of process.
(a) topology based, (b) process dynamic based

simultaneously solves each module in the cluster that is coupled, it does not need to consider the complicated step to get the tear stream that is needed to get a sequential structure of unit modules.

Generally, a whole process can be partitioned into several blocks according to the process topology or dynamics. Fig. 1 shows the examples of the partitioning. This study uses the strategy that the system makes the clusters automatically from the process topology and the user changes the cluster structure to increase the performance of calculation.

1. Simulation Model

The model equations representing any unit of chemical process can be classified into three groups: first-order ordinary differential equations (ODEs) representing the balance equations [Eq. (1)], a set of internal algebraic equations [Eq. (2), (3)] and a set of coupled algebraic equations [Eq. (4)]. The internal algebraic equations include only internal algebraic variables, x, which are defined only in the module, whereas the coupled algebraic equations include external algebraic variables, z, which are defined in the other module.

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

$$x_i = g_i(y, x_i, x_e, d, t) \tag{2}$$

$$x_e = g_e(y, d, t) \tag{3}$$

$$x_c = g_c(x, y, z, d, t) \tag{4}$$

Here, y is ODE state variable, x and z are algebraic variables, d is design parameter and t is time. If ODE mode [Gani et al., 1990; Gani et al., 1992] is used to solve the above equations, all ODEs of a cluster are solved simultaneously by a common integrator and then each module is in charge of solving algebraic equations in the sequential modular fashion.

2. Three-phase Calling Procedure

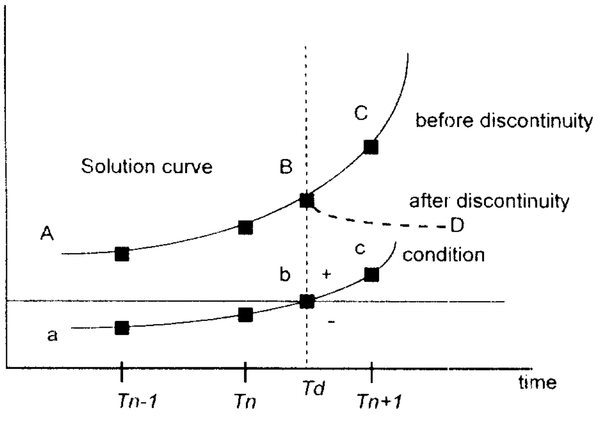

Fig. 2. Discontinuity location.

An inexact coupling may be introduced when Eq. (4) is solved, since the external algebraic variable, z, may not be updated. To avoid this problem, following three-phase module calling procedure is suggested.

Phase 1 – Solving the internal algebraic equations: To solve internal algebraic equations, each module is called by an integrator through interface. After this stage, the values of all internal variables in each module are known. It is expected that many of external variables, z, may be calculated.

Phase 2 – Solving the coupled algebraic equations: To solve coupled algebraic equations, each module is called by integrator.

Phase 3 – Evaluation of right hand side of ODEs: Through phase 1, 2, all variables are available to evaluate the r.h.s. of Eq. (1).

Although above three-phase calling procedure copes with many types of practical simulation problems, it is difficult to have exact coupling when Eq. (4) takes the form of $x_c = g_c(z_c, \ldots)$. This often occurs when the modern complex control scheme is introduced. The only weakness of the modular approach presented here is sequential treatment of this true coupling algebraic equations. A direct substitution is implemented in the developed simulator to overcome this disadvantage. In fact, it was reported that this iteration scheme is stable and has no special difficulties in dynamic simulation [Ponton, 1982]. Therefore general 3-phase calling procedure is used: $1\text{-}2^M\text{-}3$ calling procedure (the phase 2 calculation is forced to do M times where M is small integer, usually, 2 or 3).

MOSA SYSTEM

1. An Overview of MOSA

MOSA is a module-based dynamic process simulator which has simultaneous integration capability and employs highly flexible structure for various application areas including discrete process simulation such as batch process.

The main components of MOSA are shown in Fig. 2. The characteristics and functions of each component are as following:

(1) Input language system

The input language system interprets the keyword-oriented input file and then generates simulation problem by writing main drive routine, event routines and data. Also, this pre-processor analyzes the flowsheet structure and generates an initial clustering structure by performing partitioning and precedence ordering. User can modify this initial clustering structure by adding his specific knowledge of the process.

(2) Executive

The executive monitors and controls the overall simulation task such as one step integration of overall problem, time and state event handling, interactive simulation, printing and display. Especially at run time, user can interrupt the simulation by pressing any key on the keyboard and communicate with the system.

(3) Supervisory integration routine

The supervisory integration routine is in charge of one-step integration of the whole problem.

(4) Equation solving package

The equation solving package includes stiff integrator, matrix solver, etc.

(5) Model library

The model library is a set of FORTRAN 77 subroutines which model unit operation blocks or computational blocks.

(6) Physical properties package

The physical properties package has a physical property database of hydrocarbon components and subroutines for calculation of enthalpy, density, K-value through equation of states such as Soave-Redlich-Kwong, Peng-Robinson, etc.

Current version of MOSA is implemented using FORTRAN 77 on PC.

2. Sparse Jacobian Handling

As the size of integration block increases, the Jacobian matrix becomes more sparse. Sparse matrix technique enables us to save storage and computing time. The sparsity pattern can be constructed analytically or by numerical differentiation. In this work, we obtain the Jacobian sparsity pattern by using sensitivity analysis based on the numerical perturbation. Using this pattern, the variables are grouped to reduce the computational burden of Jacobian evaluation by finite difference method.

3. Discontinuities Handling

Various discontinuities can be introduced in dynamic simulation. Typical reasons for discontinuities in process model are phase change, control variables change by digital control system or a switch from a certain model equation to more appropriate one during a transient. Furthermore a process may have inherent discontinuities due to discontinuous process operation. These are typically encountered in batch processing and in continuous processes during start-up or shut-down operations.

In this work, MOSA finds discontinuity time using interpolation of discontinuity equation which is saved as function. As the sign of the equation is changed at the discontinuity point, MOSA changes the equation to Lagrange polynomial function using the past value of the equation and finds this point by Newton method. This method is very efficient as it solves only one polynomial equation. As the discontinuity does not occur during integration, the stable integration is possible. When the discontinuity point is detected, MOSA performs a new integration including the equation that causes the discontinuity regardless of the previous integration process.

Fig. 3 explains this method. When the integration is performed to T_{n+1}, the sign of the a-b-c curve which represents the

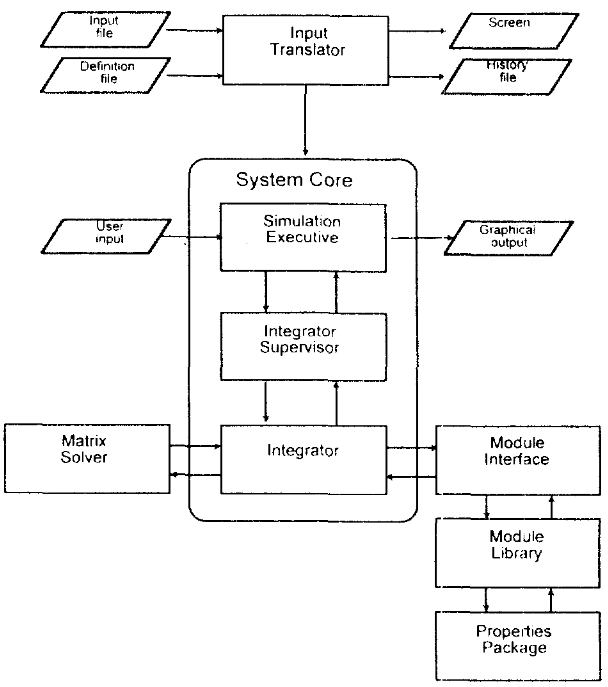

Fig. 3. The architecture of MOSA.

discontinuity condition is changed and the solution is C. As the a-b-c curve is changed to the polynomial equation and this equation is solved by Newton method, the discontinuity point of the condition curve, b, is found and the discontinuity point, T_d is determined. As the discontinuity point is known, the solution curve is interpolated to get the solution of this point, B. When the solution at T_d is obtained, the integrator starts to solve the new problem with the initial value of this solution. After the discontinuity is introduced, the shape of the solution curve is changed to A-B-D.

4. Event Processing

Many discrete events are included in applications of dynamic simulation; for example, batch process, start-up, shut-down operation, etc. For these discrete events, the concept of event and state is used. An event is something that happens at a point in time, such as equipment trip, operation change and the change of the system state to a new state. A state diagram can be represented by a graph whose nodes are states and directed arcs are the transitions labeled by event names. An example of a state diagram is shown in Fig. 4, and event section of input file, in Fig. 5. Events have condition statement which represents the time of the event and action statement which represents the action at this time. The type of condition statement is either implicit or explicit. Some authors refer to these as state event and time event [Barton and Pantelides, 1994]. For an explicit event, the occurrence time is known. Thus, the integration is performed to exact point of discontinuity and initial value problem is now solved from this point. On the contrary, for an implicit event, the time of the occurrence of the event (discontinuity) is not known. This time is obtained by the method explained previously. Next step is identical to explicit event handling.

And the type of action is either point or duration. If the con-

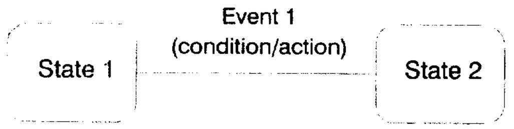

Fig. 4. State and event diagram.

```
SECTION EVENT
  Boiler_Trip IN CLUSTER 1
  TYPE = EXP , POINT
  COND = ( TIME - 60. )
  ACTION { NB = 4.0 }
```

Fig. 5. Event section.

dition is satisfied, point action is taken once at that moment. Duration action is a continuous action taken during the time when the condition is satisfied.

PERFORMANCE TEST

To evaluate the performance of the cluster-modular approach, a simple example is presented as shown in Fig. 6. For simplicity, all modules are assumed to be a perfect mixing tank without heat involved. Even distribution of flows from the same unit is also assumed. The mixing tank has NC ODEs (component material balance equations) and 3+NOUT AEs (total holdup, total outlet flow rate, level of tank and NOUT outlet flow rates), where NC is number of component and NOUT is number of outlet streams of a unit module. After partitioning, we have 6 clusters, (1), (2 3 4 5), (6 7 8), (9 10), (11), (12). Various runs are compared according to the size of cluster, and Jacobian evaluation method in Table 1.

In case I, full type Jacobian and no grouping method is used whereas in case II, sparse type and grouping method is used. In both cases, three runs are tested according to number of clusters (12 indicates all sequential and 1 does all simultaneous integration). In case I, 6-cluster simulation shows the best result. Though 1-cluster method takes relatively large step size and requires small number of steps, it requires more function evaluations due to having large size of Jacobian matrix. In case II, however, 1-cluster is the best when sparse technique is used.

CASE STUDY

A real process was simulated to show the ability of MOSA. This is the utility boiler process of a refinery plant, which consists of 5 boilers including control system. The process model comes from Lee's study [Lee, 1993a] and consists of 11 unit blocks and has 19 ordinary differential equations and 118 variables. Fig. 7 shows the block diagram of the utility boiler plant. The key function of a utility boiler plant is to supply high quality steam stably; hence the pressure of header should be controlled tightly. The level of a drum should also be controlled to guarantee safe operation. The objective is the improvement of

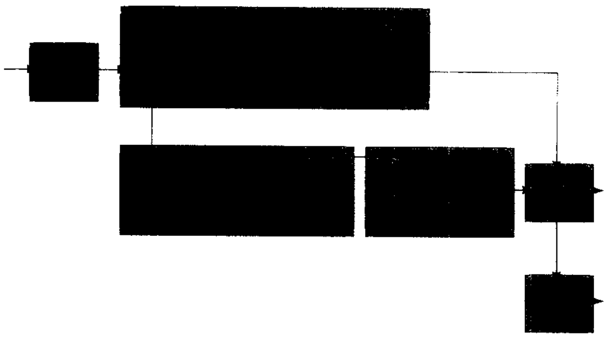

Fig. 6. Block diagram of 12-unit process (gray box indicates one cluster).

Table 1. Performance statistics for Fig. 4 process

Case	I: full, nogroup			II: sparse, group		
No. of cluster	12	6	1	12	6	1
nstep	151	127	117	-	117	108
nfe	164	160	202	-	140	139
nje	1	1	2	-	1	1
cpu (%)	100	78	95	-	69	62

nfe: number of function evaluation
nje: number of Jacobian evaluation
cpu: cpu time (percent ratio compared with case I of 12-cluster)

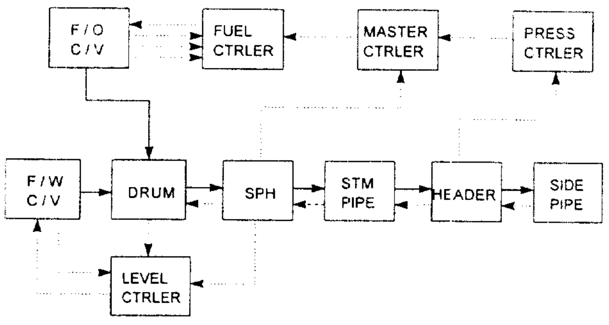

Fig. 7. Block diagram of utility boiler plant.

operability and safety through dynamic simulation of expected faults or situations. Thus, after the inspection of the simulated behavior of the safety device such as pressure safety valve equipped at the boiler, the safety of the boiler operation can be enhanced by the outfit or the adjustment of the device.

The simulated situation is the trip of one of the 5 boilers and the simulation time is 60 minutes. Fig. 8 shows the pressure of high pressure header and the level of boiler drum versus time. In this figure, the pressure and the level reach a new steady-state in about 30 minutes. Although the direct comparison of the simulation results with the real data in this situation was impossible for the difficulty of data acquisition, it was proven by field engineers that the trend of these two important variables was similar to the real situation.

CONCLUSIONS AND FURTHER STUDIES

The general purpose dynamic process simulator, MOSA, which

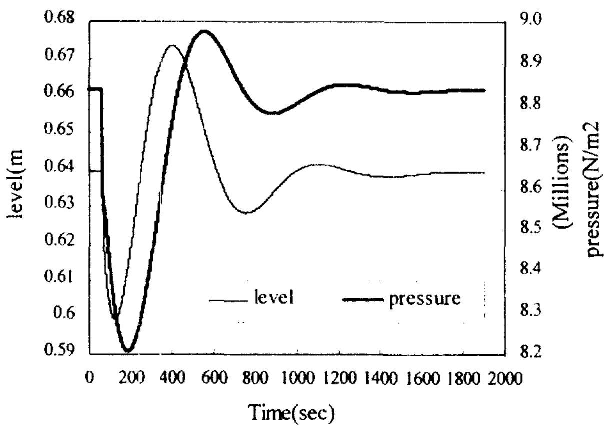

Fig. 8. Pressure and level.

is based on cluster-modular approach is presented. It is shown that the architecture of MOSA is adaptable to various problems and application areas through the case studies of batch and continuous processes such as batch reactor, tank, distillation column [Lee, 1993b], and boiler.

The methods based on sequential modular approach as the one used in this study require the user to provide the reasonable initial conditions. Therefore, if the main objective of the simulation is to study the responses to disturbances at steady state, the method to get the steady state values, such as the interface with external steady state simulators, should be studied.

Current general-purpose dynamic simulators do not solve the high-index problems which occur mainly in the process with equilibrium calculation or dynamic design problem. However, many cases of index problem occur due to the simple fault of modeling or unnecessary assumptions for modeling. Therefore, the structural method for modeling is needed to avoid index problem in modeling process. Also, the expansion of model libraries and applications to real plants is being studied.

ACKNOWLEDGMENTS

This work has been supported by the MOSA consortium, which consists of 20 refinery, engineering and chemical companies, and Automation Research Center at Pohang Institute of Science and Technology.

REFERENCES

Barton, P. I. and Pantelides, C. C., "Modeling of Combined Discrete/Continuous Processes", *AIChE J.*, **40**(6), 966 (1994).

Gani, R., Perregaard, J. and Johansen, H., "Simulation Strategies for Design and Analysis of Complex Chemical Processes", *Trans IChemE*, **68**, 407 (1990).

Gani, R., Sorensen, E. L. and Perregaard, J., "Design and Analysis of Chemical Processes through DYNSIM", *Ind. Eng. Chem. Res.*, **31**, 244 (1992).

Lee, G. B., "Dynamic Modeling and Simulation of Steam Boiler Plant in Chemical Process", M.S. Thesis, Seoul National University (1993a).

Lee, K. J., "A Study on Dynamic Process Simulation with the Flexible Modular Approach", Ph.D. Thesis, Seoul National University (1993b).

Marquardt, W., "Dynamic Process Simulation-Recent Progress and Future Challenges", *Proceedings CPC IV*, 131 (1991).

Ponton, J. W., "Dynamic Process Simulation Using Flowsheet Structure", *Comput. Chem. Engng.*, **7**, 13 (1982).