MODELING, IDENTIFICATION AND CONTROL, 1989, VOL. 10, NO. 4, 203–211
doi:10.4173/mic.1989.4.2

Simultaneous computation within a sequential process simulation tool

G. ENDRESTØL†, T. SIRA†, M. ØSTENSTAD†,
T. MALIK‡, M. MEEG‡ and J. THRANE‡

Keywords: *Dynamic process simulation, sequential modular simulator, simultaneous network solution.*

The paper describes an equation solver superstructure developed for a sequential modular dynamic process simulation system as part of a Eureka project with Norwegian and British participation. The purpose of the development was combining some of the advantages of equation based and purely sequential systems, enabling implicit treatment of key variables independent of module boundaries, and use of numerical integration techniques suitable for each individual type of variable. For training simulator applications the main advantages are gains in speed due to increased stability limits on time steps and improved consistency of simulation results. The system is split into an off-line analysis phase and an on-line equation solver. The off-line processing consists of automatic determination of the topological structure of the system connectivity from standard process description files and derivation of an optimized sparse matrix solution procedure for the resulting set of equations. The on-line routine collects equation coefficients from involved modules, solves the combined sets of structured equations, and stores the results appropriately. This method minimizes the processing cost during the actual simulation. The solver has been applied in the Veslefrikk training simulator project.

1. Introduction

Process simulators are historically divided into the main categories, sequential modular (subroutine based), and simultaneous (equation based) (Hillestad and Hertzberg 1986). Sequential computation is traditionally used in most commercial products. It has the advantage that stems from the inherent flexibility of data processing within subroutines and the associated freedom in selection of integration methods suitable for different types of variables. In addition no specific process dependent program compilation is usually necessary, since the system can easily be made fully data driven. Process specification and modification may also be done using advanced, user friendly, tools. The main drawbacks of such methods are associated with the explicit couplings between modules. Process simulations almost always contain both fast and slow components, and a realistic computation based on explicit numerical methods require time steps much shorter than those needed for reasons of accuracy. In a dynamical, real time, simulation this may be disastrous, since the necessary shortening of the time step may lead to unacceptably long computation time per simulated time unit. The alternative of modifying the equations for stiff components is equally unpalatable, since it changes the dynamic response times for the simulation.

Received 1 August 1989.

† Institutt for energiteknikk, P.O. Box 40, N-2007 Kjeller, Norway.
‡ Norcontrol Simulation A/S, P.O. Box 1024, N-1391 Horten, Norway.

This paper was presented at SIMS '89 (Scandinavian Simulation Society), 31st Annual Meeting, Bergen, Norway, May 31–June 2, 1989.

Equation based systems are strong regarding selection of numerical integration methods, since these may usually be chosen independently of the models used for the simulations. They also normally lead to more consistent solutions, e.g. because exact mass conservation may be achieved. On the other hand, for large simulations it may be wasteful treating all variables by such methods. Very often also the same numerical approximations will be made for all types of variables. This frequently leads to unnecessarily large equation systems that may require significant computational resources for real time solution. The way system connections are treated within these systems also often necessitates separate compilation of large amounts of code for each simulator. Many engineers actually involved in writing solution algorithms for process modules also find the format used for process specification within equation based systems less intuitive than the corresponding format for subroutine based systems.

Some attempts have been made to improve the performance of sequential simulators by modifying the input data to models (Brosilow *et al.* 1985, Hillestad and Hertzberg 1986). Many of these methods are based on extrapolation of or interpolation between known variable values. They require a specialized design of the system executive for the simulator. The programs described in the present paper were designed to enhance a sequential process simulation system by including the possibility of implicit couplings across model boundaries. In this way all variables controlled by the model itself may be integrated internally, while only stiff external couplings are treated by the superstructure. The solver was aimed principally towards the needs of a real time dynamic process simulation, particularly for training purposes, where output is needed frequently in order to update operator station data. For such purposes often relatively crude unit process models are implemented. This means that the time step will be chosen so short that stability rather than accuracy restricts the calculations. In that case the pressure driven system flows are normally critical to stability. The presently described software takes care of exactly such types of intermodel variables. For this reason the product has been termed a Network Solver. The program system was conceived, designed and implemented at Institutt for energiteknikk as part of a more extensive upgrading of an existing simulator.

2. Network solver environment

The solver software was required to work within the structure of a previously developed sequential process simulation package OTISS (Operator Training Industrial Systems Simulator) made by Special Analysis and Simulation Technology Limited, London, UK. The presently relevant principal characteristics of this simulator are as follows.

It is a C-language based system for dynamic simulation with present applications primarily within the petroleum industry. It has been used both for audit simulations and for training simulator applications.

In addition to dynamic simulation of the actual process also the control and logic systems are included, and facilities both for instructor and operator stations are available.

A thermodynamical and physical properties package is available or a tailor made one may alternatively be installed.

The basic building blocks of a simulation are algorithms with specified sets of parameters and variables. The model of a physical unit may be made up of one or several algorithms. Couplings between different units are specified directly by the argument names used.

The structure of the complete system being simulated is defined by specifying a tagged sequence of algorithms with corresponding argument lists containing variable names. The input stream may contain commands giving rise to program loops useful, e.g. for iteration or integrating subsystems with different time steps. A list of parameters for each instance of an algorithm also has to be prepared.

From these sets of input data the system makes a database for variables and parameters. It contains both names and actual current values. This common database is accessed by the algorithms during a simulation.

It was of primary importance that the solver should work without any essential modifications to this structure. It was also desirable to maintain the inherent flexibility of the simulation specification and to include the possibility for implicit iteration on subsystems. To avoid explicit perturbations of model variables in order to produce response data, presently available unit models would however have to be modified and new ones possibly written for units having no previous counterpart.

3. Functional specification and design goals

In dynamical simulations of process plants several types of connections between different system units must be taken into account. In addition to mechanical, electrical, and logical (signal) couplings, one of the most important arises simply from the transportation of fluids between units. These must usually be modeled as pressure driven flows. A realistic simulation of such flows will very often result in extremely stiff numerical problems both because of small volumes in the vessels themselves as well as in the piping used for transportation, and because the fluid medium may be effectively incompressible. In many instances also relatively large tightly connected networks of such units will be simulated. It is usually not acceptable to produce custom made complex models for each simulation, and building these networks from standard model libraries gives rise to strong connections between variables in many separate algorithms. As a sequential simulator OTISS has few easily accessible tools for handling such systems and especially for making efficient solutions to large connected networks. The network solver addresses the type of problems posed by this kind of couplings.

Consequently the intentions of the project were as follows.

To build a superstructure to the collection of separate unit models enabling truly coordinated simultaneous solution to pressures and flowrates within a whole network. The network may contain transport units like pipes, valves, pumps, compressors, etc., and capacitive units like separators and scrubbers.

The system should be able to locate isolated sub-networks within the total process, and be capable of solving the equations for each such group separately. This may be of importance both for the effectiveness of the solver itself, and at the same time enables iteration on non-linearities in single sub-networks if necessary. Usually, however, no iteration will be needed within a timestep.

The structure of the resulting systems of equations is normally quite sparse. The solver should use effective methods to solve such systems to save both CPU time and memory needed during the solution.

The solver should be able to function within the framework of the existing simulator with only small or no modifications at all to the core of the system. Previous applications should be unaffected by the enhancement. Unit models making use of the solver should be simple to interface to the system.

The solver should be fully automatic needing no explicit information by the user concerning the details of the current network or the method of solution. All necessary information should be extracted from standard OTISS input files and general unit model description files.

To achieve these goals the solver was divided into two distinct phases; an off-line network topology analyser and solution method generator, and a real time solver implemented as a normal OTISS algorithm. In this way all time consuming analysis and optimization processing is done off-line, and the only on-line activity is the actual application of the solver to the pre-analysed networks in question.

4. Logical network structure and solution method

The topology of a network of pressure driven flows is normally that some type of generalized piping connects vessels containing processing equipment. Some nodes may also be of a type that simply joins pipes without performing any activity besides mixing input streams and distributing them to output streams. Several pipes may connect to one vessel, and the vessels may have internal structures giving rise to several different pressures and possibly effectively some internal flows.

The network solver has been designed to be able to integrate this kind of network. All types of pipe models are assumed to produce flow rates as functions of end pressures, and the basic external couplings for vessel pressures are supposed to be flows through connected pipes. Pipe units may use incompressible (one flow per pipe) or compressible (two flows per pipe) models, and to take care of vessels with a more complex internal structure some pressures may also depend directly on other network pressures. One unit model may contain several pressures and/or several pipes. Completely internal flows treated by the network solver are also possible. Process boundaries are taken care of by special types of pressure and flow units.

The network equations employed are in principle determined by each unit model. All coefficients in equations between (potentially) connected variables are model generated. The solver collects and combines these data into the set of equations for the whole network. For training simulator purposes, however, the natural choice is a once-iterated implicit Euler formalism. There are several reasons for this. A few of the more important are:

Training simulators normally use such short time steps that the required accuracy is obtained by a simple Euler method. The crucial concern in this connection is stability rather than accuracy. It is of vital importance that the simulator does not break down or use excessive time to obtain the end result for any time step. In this respect robustness is equally or more important than accuracy.

In a medium size or large simulator discontinuities in derivatives or even function values will occur constantly. This is a result both of the physical processes taking place and the numerical algorithms implemented for solving the process equations.

It is therefore inappropriate to use higher order backward methods for these systems, and Runge–Kutta type of methods (Gear 1971) require intermediate function values that are not easily obtained in this kind of environment. Frequent restarts are also usually unacceptable due to limitations in computing power.

Iteration of the Euler method to obtain the true implicit solution does not normally improve the numerical accuracy, and it is possible to formulate the equations in such a way that stability is achieved by using only one iteration per time step.

Modeling of pipe flows is very important for the proper functioning of the network solver. For small flowrates it exhibits a highly non-linear flowrate versus pressure drop characteristic. A normal type of Newton iteration may not even be stable for all situations, and it often converges extremely slowly. To get stable and correct results using a once-iterated implicit Euler method therefore requires a different linearization method. The combination of suitable equation formulations and efficient solution methods is the key to the success of any implicit solver in the real time simulator business. For the system to be actually taken into use, both by modelers and by system builders, a simple and time saving user interface is also of primary importance.

We have found that a secant method through zero flow gives numerically satisfactory results. Usually it will also be assumed that compressible fluid models are used for all vessels. Even for liquids the normal compressibility, perhaps combined with actual pipe or tank volume versus pressure characteristics, is fully sufficient for creating a numerically well-behaved system of equations. The possibility for very small compressibilities (potentially used only for numerical reasons) has been incorporated in the formulation of the equations used by the solver. The end result is then normally an unconditionally stable and diagonally dominant system of linear equations to be solved at each time step. No special attention has to be paid to closed valves or similar occurrences. If compressibility is neglected, isolated sub-networks may otherwise result in a singular coefficient matrix. The present method implies that a regular Gaussian elimination method may always be applied for solving the equations.

The solver itself has been based on direct solution of the system of pressure equations using a sparse matrix Gaussian elimination method (Duff *et al.* 1986). For most systems an iterative solution method (Hageman and Young 1981) will not be superior for real time applications because the coefficient matrix usually may change dramatically from one time step to the next. The problem size is normally also quite manageable for direct solvers. The solver needs to collect equation coefficients from the OTISS database, set up and solve the appropriate network equations, and store the required results. To incorporate the necessary degree of flexibility into this kind of system, a table driven solver seems like a suitable choice for developmental basis. This is true all the more because tables may be stored much more compactly than equation coefficients, and because most of the relevant processes lead to highly sparse systems where considerable fill in occurs only in the very last stages of the elimination process.

Presently the solver pre-eliminates all network flows and sets up equations for pressures only. This often cuts the size of the global problem more than in half, and the elimination can be done in such a way that maximum numerical accuracy is maintained throughout. It also leads to systems that can be analysed by standard methods for sparse matrices. The local Marcowitz criterion has been chosen as the primary elimination order selection method, supplemented with a set of different tie-breaking strategies. Their primary purposes are to maximize the efficiency of the table driven solver and to ensure stability of the resulting schemes against arbitrary reordering of the input data not producing any topological changes to the actual network.

Re-use of the same coefficient matrix for all sub-networks and re-use of coefficient positions during the solution of each sub-network guarantee minimum memory requirements for the solver. To obtain sufficient accuracy of the result, the whole solution is performed in double precision arithmetic. This ensures that no small flows are lost during the process due to round off errors. The present solver will not give incorrect results even for very high conductancy pipes. Loss of accuracy might otherwise be a problem for implicit solvers not using numerically based pivoting strategies. This type of pivoting has been decided against both for reasons of memory capacity management and real time computational requirements.

5. Off-line data processing

The off-line processing performs the main tasks for the network solver system. All network analysis and solution optimization is done at this stage. This is done to minimize the real time processing costs and because the calculations do not take any excessively long time as judged by a user of the system. This processing is split into two parts: (1) Collection of the required data from OTISS input files and determination of the network structure, (2) Determination of the actual solution strategy and production of input data for the on-line solver routine.

To be able to determine the structure of the network and obtain the necessary variable name and coupling information in each case, the first program, the network topology analyser, requires the following generic unit model information:

The generic model name including necessary data like the total number of algorithms involved.

For each pressure or flow component within the model the actual component type. Several types have been defined for potential use. They differ in their formally 'external' couplings. Some models may have a flexible structure where the actual configuration is determined partly by parameters for each actual instance.

Location information to where names of all required variables and values for all needed parameters may be found.

This data is common to all actual usage of the models. It is stored in a model description file which must be updated every time a generic model is included or modified, but is otherwise independent of the actual simulator implementation.

Concrete applications are made from reading standard OTISS driver files containing all the needed specifications. Implicit in this data is also the structure of the whole network.

From this data the program makes certain that a consistent and complete network has been specified and writes the necessary information about all instances of all pressure and pipe units and their couplings to a network topology file read by the second off-line program, the network analyser. The program will produce messages if any errors are detected during the processing. In this way the user will also be more certain that a correct structure has been set up for the simulation.

The network topology file may pass directly to the network analyser, but the user has some possibilities for modifying the file. In the original form all couplings between pressures and flows are taken to be implicit. Occasionally the user may want to modify this property. It is possible to specify that any flow coupled to any vessel pressure shall be integrated by explicit numerical methods. This means that either the value of the flowrate itself or the pressure at the opposite end shall be taken from the previous iteration (time step). This may be a means of simplifying a complicated network by introducing decouplings at 'safe' positions or a way of generating sub-networks from desired components, e.g. for special non-linear iterations. Normally, however, no such explicitness specifications will be used.

The network analyser requires only the network topology file as a basis for its computations. From this it performs the following main tasks:

It determines the structure of the global system of equations for all network pressure variables.

It splits the network into a sequence of independent sub-networks. Subnetwork dependencies are taken into account.

For each sub-network it tries to find the optimum solution method and makes tabular data that determines the detailed solution procedure. It also keeps track of the state of the coefficient matrix to ensure maximum efficiency in the utilization of memory capacity.

It prints all necessary tabular processing information to files read by the real time solver and makes user oriented files specifying the contents of all sub-networks found.

The person(s) making the actual application may not know anything about the details of the network solver to be able to include the solver in their calculations and prepare its usage during the simulations. Only the names of the off-line programs and of the real time algorithm is needed.

6. Real time solver

The solver routine has been implemented using same simulator interface as any other algorithm in the OTISS system. This guarantees that no simulator modification is needed for its inclusion. It also allocates dynamically all memory needed for storage of the coefficient matrix and the tables of instructions determining the solution procedure for each sub-network. At the initial call to the solver the input files are read and the necessary preparations for the solution process are made. The only special facet about the solver relative to other models is that it must access database variables that are not its own. It has to be able to reach all network pressure and flow variables as well as variables containing coefficients in the appropriate equations. These belong to other models and are not stated on it's argument list. This is achieved by calling standard simulator routines returning pointers to specified variables.

Since the network solver looks like an OTISS algorithm, it must be included in the input files to the simulator. This may be done in two ways. The simplest method is to call the solver once after all relevant model routines with parameters specifying that all sub-networks shall be solved during this single call. For the more sophisticated user calls to single sub-networks may be inserted at appropriate points in the list of algorithms. By using an intelligent ordering of models this may lead to more accurate results since updated values will become available to model routines at an earlier point in time. It also makes it possible to introduce iteration processes containing the solver by including the appropriate set of models into the loops or to make different parts of the simulation run with different time steps.

The solver reads all essential network information from special, automatically generated, files. This has made it possible to make its simulator parameter and variable lists extremely simple. It contains only two parameters, one specifying the network version (if several have been made with different explicitness declarations) and the other the actual subsystem. In addition it has one single (output) status variable.

7. Conclusions

The OTISS process simulation system has benefited from the inclusion of the network solver at several different levels. Some of the product enhancements are:

It has simplified the tasks of the person developing unit models of some important types by presenting a tool that automatically connects key interface variables implicitly to complementary variables in other models. Normally the effort needed to make use of the solver in a model is negligible compared to the benefits reaped by using it. Also very little knowledge about the numerical methods involved are necessary to be able to use the solver.

It has improved the productivity during the generation of a simulation setup by making it possible to pay less attention to numerical stability problems that may occur during a simulation run. No specialized technical knowhow is required to produce the simulation specification information.

It has made it possible to reduce the computation time per simulated time unit by enabling safe usage of longer time steps without danger of numerical instabilities and by employing an efficient solution procedure for the resulting set of equations.

It has improved the quality of the simulation results by ensuring availability of a more consistent set of values for pressures and flowrates at any point in the simulation. This makes it easier to obtain exact mass balance for the process.

The network solver was employed for the first time in the Veslefrikk training simulator project. This is a relatively small scale petroleum production platform process simulator made for Den norske stats oljeselskap a.s. (Statoil) by Norcontrol Simulation A/S. In this case the solver was applied in the most straightforward manner without utilizing any of its more specialized capabilities. No particular problems were encountered during this trial application, and the resulting simulator has successfully passed functional tests beyond its specified operational limits.

Acknowledgments

Funding was given through the Eureka programme, which is aimed at supporting European state of the art development projects with a bi- or multi-national character. In the present case the cooperating companies were Norcontrol Simulation A/S, Norway and Special Analysis and Simulation Technology Limited, UK.

References

Brosilow, C. B., Liu, Y.-C., Cook, J. and Klatt, J. (1985). Modular integration methods for simulation of large scale dynamic systems. *International Seminar on Modern Methods in Dynamical Simulation of Industrial Processes*, The Norwegian Institute of Technology, Trondheim, May 20–22, 1985.

DUFF, I. S., ERISMAN, A. M. and REID, J. K. (1986). *Direct Methods for Sparse Matrices* (Oxford University Press).

GEAR, C. W. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice-Hall, Inc., Englewood Cliffs, New Jersey).

HAGEMAN, L. A. and YOUNG, D. M. (1981). *Applied Iterative Methods* (Academic Press, Inc., New York).

HILLESTAD, M. and HERTZBERG, A. (1986). Dynamic simulation of chemical engineering systems by the sequential modular approach. *Modeling, Identification and Control*, **7**, 107–127.