MODELING, IDENTIFICATION AND CONTROL, 1985, VOL. 6, NO. 3, 153–179
doi:10.4173/mic.1985.3.4

Modular integration methods for simulation of large scale dynamic systems†

COLEMAN B. BROSILOW‡, YIN-CHANG LIU‡,
JEFFREY COOK‡ and JOHN KLATT‡

Keywords: *Modular, Simulation, Dynamic, Parallel Processing, Coordination, Integration*

Modular simulation of dynamic systems offers the possibility of computational speed through parallel processing of individual sub-systems and through the use of the best integration algorithms for each sub-system. Such simulation needs co-ordination algorithms to keep the various sub-systems in time synchronization and to compute the interconnection between the sub-systems. A mathematical description of the co-ordination problem leads to the development of several new algorithms. These new algorithms are shown to have desirable convergence and stability properties. In particular a new Newton type algorithm is A-stable in a sense similar to that defined for ordinary integration algorithms.

Numerical tests with several small example problems and with the simulation of the dynamics of an atmospheric crude unit consisting of five interacting columns are used to evaluate the various co-ordination algorithms. The crude unit simulation was carried out using a prototype modular simulator for distillation systems. This simulator is briefly described.

1. Introduction

The increase in the price of energy has brought about conservation measures which have increased the degree of integration in many chemical and petroleum processes. A new dynamic simulation capability is needed to enable engineers to anticipate and compensate for the complex interactions produced through such process integration.

There are two different numerical approaches to simulate the dynamics of an integrated process: (1) the various sub-systems are integrated with a single algorithm (ordinary integration), or, (2) each sub-system has its own algorithm (modular integration).

In the first approach (ordinary integration), all linked sub-systems are considered as one single large system. A single algorithm, explicit or implicit, is used to simulate the dynamics of the whole system. Time is advanced the same amount at each step for each sub-system no matter if it is stiff or not. Typical examples of simulators using variants of ordinary integration are: MIMIC (Sansom and Peterson 1965), CSMP (IBM, 1972), DYNSYS (Barney *et al.* 1975a, 1975b), SPEED UP (Perkins *et al.* 1982) and ASCEND (Westerberg 1980).

In modular integration, each sub-system is integrated independently with independent error control. Explicit and implicit integration algorithms are used to inte-

Received 15 June 1985.

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

‡ Case Western Reserve University, Cleveland, Ohio 44106, U.S.A.

grate non-stiff and stiff sub-systems separately. An example of a simulator using modular integration is MODCOMP. (EXXON 1976).

In general, modular integration may possess the following advantages over ordinary integration:

(*i*) The simulation can be more efficient because:
  (*a*) each sub-system uses an integration algorithm which is best suited to that sub-system.
  (*b*) each dynamic simulator has its own error control, and
  (*c*) all dynamic simulators can operate in parallel.

(*ii*) The software can be completely modular and therefore easier to maintain.

Most chemical and petroleum processes are examples of systems with stiff and non-stiff components. The modular approach to integration permits the use of explicit integration algorithms for the non-stiff sub-systems and implicit integration algorithms for stiff sub-systems. Independent error control in the individual dynamic simulators insures that the proper step size is taken in each sub-system. Thus, the efficiency of the overall simulation is not adversely influenced by the step size in any single sub-division.

Because of the separate integration algorithms for the individual sub-systems, debugging of the computer program for modular simulation can be reasonably simple. Each simulator can be tested independently to locate any possible programming errors. In contrast, the location of errors in highly integrated computer software can be very difficult and time consuming. In addition, a modular simulation can be expanded with little or no disturbance to existing programs.

The principal obstacles to running two or more dynamic simulators simultaneously are communications between simulators and synchronization of the simulation. When a large simulation is composed of several simulators, the individual simulators are not independent of each other. The outputs of one simulator may be input to another in the same way that the outputs of one chemical process unit may be inputs to another. the simulations must proceed at roughly the same pace with none falling behind or racing ahead of the others.

Miller (1978) has developed an interface to link dynamic modules to MIMIC or CSMP. The interface allows the dynamic modules to use different integration algorithms and take different step sizes. Controllers are usually simulated by MIMIC/CSMP and the other process units are simulated by the user generated dynamic modules. No pre-programmed dynamic modules are supplied with Miller's interface.

Miller's interface keeps the simulation in time synchronization by making the modules compute forward until they pass the point MIMIC/CSMP is expected to reach on the step it is about to take. Linear interpolation or extrapolation is used to obtain the inputs needed by MIMIC/CSMP or the modules. If the error in the interpolation or extrapolation is greater than the error tolerance, the module can back up and recompute. Unfortunately, MIMIC/CSMP can not back up when there is an error in the interpolation or extrapolation, so there is no error control between MIMIC/CSMP and the modules. Another problem with Miller's interface is that the MIMIC/CSMP step size controls the overall step size in the simulation when it should be controlled by the error in the interpolation or extrapolation among all the modules.

An alternate method of constructing a modular simulator is with a co-ordinator.

The co-ordinator allows each module to have a different integration algorithm and to take a different step size from the other modules. The co-ordinator keeps the dynamic modules in time synchronization and supplies inputs to the dynamic modules which come from other simulators. Time synchronization is accomplished by having the co-ordinator set a time horizon over which each dynamic module computes. The accuracy of the simulation is maintained by adjusting the size of the time horizon.

Cook (1980) and Cook and Brosilow (1980) have developed a dynamic simulator for distillation systems using the concept of a co-ordinator. The simulator consists of a Fortran software package with dynamic modules to simulate distillation columns, reboilers, condensers, and control systems. The co-ordination algorithm employed by Cook uses either constant or linear extrapolation method to calculate the inputs to each sub-system. Cook's simulator is described briefly in § 5 as an example of the use of extrapolation algorithms. This simulator has been used by Klatt (1983) and Klatt and Brosilow (1984) to simulate the dynamics of a crude unit and this application is described briefly in § 6.

Even though co-ordinators using extrapolation methods have been successfully used to solve significant problems, they can run into trouble. The time horizon used by the co-ordinator can become very small when the dynamic modules are strongly interconnected (Liu 1983; Liu and Brosilow 1983 and Gomm 1981). A small time horizon defeats the purpose of modularization and leads to excessive computational effort. To overcome this problem a Newton type co-ordination algorithm has been developed. The Newton algorithm is described in § 3.

Before describing the various co-ordination algorithms, the co-ordination problem is stated in the next section.

2. The co-ordination problem

The co-ordination problem is formulated for a general dynamic system with a total of n interacting sub-systems.

A sub-system (kth) can be described by eqn. (2.1):

$$x_k(t) = f_k(x_k(t), u_k(t), v_k(t), t) \tag{2.1}$$

where $x_k(t)$ is the state of sub-system k, f_k is a vector of non-linear functions describing the dynamic behavior of the sub-system, and $(u_k^T(t), v_k^T(t))^T$ is a vector of inputs. The inputs are divided into two parts: $v_k(t)$ represents the external input to sub-system k from the surroundings and $u_k(t)$ is the vector of inputs to sub-system k from other sub-systems.

The output $q_k(t)$ from sub-system k can be a vector function of $x_k(t)$ and $(u_k^T(t), v_k^T(t))^T$.

$$q_k(t) = q_k(x_k(t), u_k(t), v_k(t), t) \tag{2.2}$$

Part or all of $q_k(t)$ can be an input to other sub-systems. Thus q_k interconnects sub-system l to the rest of the system. To simplify notation, we assume that all of q_k is connected to the rest of the system.

Figure shows the relationship of sub-system k to the rest of the system. For any sub-system j which receives its inputs from sub-system k, part (or all of $q_k(t)$ becomes part (or all) of $u_j(t)$.

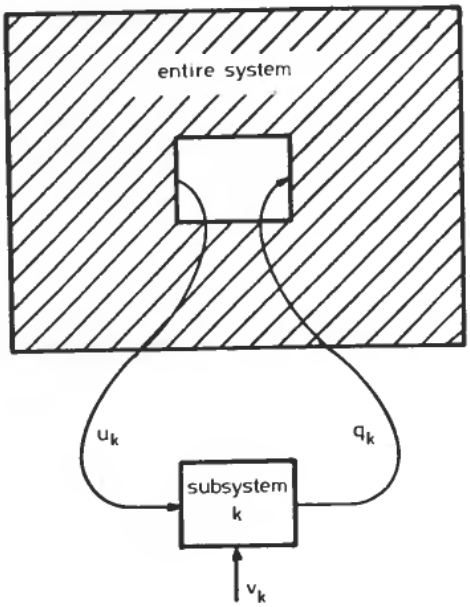

Figure 1. Relationship between sub-system k and the entire system.

The state of the kth sub-system can also be expressed in the form of an integral equation:

$$x_k(t) = x_k(0) + \int_0^t f_k(x_k(\tau), u_k(\tau), v_k(\tau), \tau)\, d\tau \tag{2.3}$$

with $0 \leqslant \tau \leqslant t \leqslant TH$, where TH is the time horizon. To simplify the notation, we use bold letters, e.g., $\boldsymbol{z}$ to denote a function defined over $[0, TH]$, and use $z(t)$ to denote the value of $\boldsymbol{z}$ at time t.

The state $x_k(t)$ in eqn. (2.3) depends only on $(\boldsymbol{u}_k^T, \boldsymbol{v}_k^T)^T$ and so can be expressed as

$$x_k(t) = x_k(\boldsymbol{u}_k, \boldsymbol{v}_k, t) \tag{2.4}$$

The output of sub-system k is then:

$$q_k(t) = q_k(x_k(\boldsymbol{u}_k, \boldsymbol{v}_k, t), u_k(t), v_k(t), t) \tag{2.5}$$

Equation (2.5) describes the input–output functional mapping of sub-system k. For an input function $(\boldsymbol{u}_k^T, \boldsymbol{v}_k^T)^T$, sub-system k creates an output function $\boldsymbol{q}_k$ over the entire time horizon.

For the n-module system, there are n vector equations in the form of eqn. (2.5). Collecting all of the inputs and outputs, we define the system interconnections as:

$$q(t) \equiv (q_1(t), q_2(t), \ldots, q_n(t))^T \tag{2.6}$$

$$u(t) \equiv (u_1(t), u_2(t), \ldots, u_n(t))^T \tag{2.7}$$

Since by definition, the sub-system outputs q_k change names to become the sub-system inputs u_j, it is convenient to define a single interconnection variable w:

$$w(t) \equiv q(t) \text{ or } u(t) \tag{2.8}$$

We can also collect the states and external inputs of the sub-systems as:

$$x(t) \equiv (x_1^T, (t), x_2^T(t), \ldots, x_n^T(t))^T \tag{2.9}$$

$$v(t) \equiv (v_1^T, (t), v_2^T(t), \ldots, v_n^T(t))^T \tag{2.10}$$

Then, by putting together the input–output mappings of all sub-systems, we have a single vector equation to describe the interconnections of the entire system:

$$w(t) = \mathrm{q}(x(w, v, t), w(t), v(t), t) \tag{2.11}$$

$$\text{where} \quad q = (q_1, q_2, \ldots, q_n)^T \tag{2.12}$$

Finding the solution of eqn. (2.11) is the co-ordination problem of modular integration. Co-ordination algorithms are designed to solve eqn. (2.11) for the interconnection variable $w(t)$ over the entire time horizon. Equation (2.11) is in the form of an integral-algebraic equation. The major difference between eqn. (2.11) and regular integral-algebraic equations is that the function q in eqn. (2.11) is not given explicitly. Only the input–output mappings of the sub-systems are available. The co-ordinator only sees the interconnection variables among the sub-systems. The states of the sub-systems are kept within the sub-systems and are not accessible to the co-ordinator. The size of the interconnection variable w is usually much smaller than the size of the state variable x.

3. Co-ordination algorithms

Four co-ordination algorithms are presented to solve the co-ordination problem given in eqn. (2.11). To avoid complications, the modules are treated as continuous systems which accept continuous inputs and provide continuous outputs. The effects of using numerical algorithms within the modules which provide only a sequence of points rather than continuous functions is treated by Liu (1983). A fairly involved analysis shows that errors introduced by the integration algorithms in the modules are not, in general, amplified by the co-ordinator.

To initiate a simulation, the user provides the following data: initial time horizon (TH), initial values of the interconnection function ($w(0)$), and an error tolerance ($E_{\max}$). The user-supplied time horizon will be used as the time horizon for the initial calculation. This time horizon may or may not be adjusted later depending on the nature of the algorithms. The initial guess of the interconnection variables enables the sub-systems to start calculations in parallel. Usually, the interconnection variables will be assumed constant at their initial values over the initial time horizon. If the initial guess is not good enough, the algorithms make the necessary adjustments automatically. The user is also requested to set an error tolerance which is used to control the computation accuracy.

There are three phases in each algorithm after initialization: (1) estimation of the error, (2) adjustment of the time horizon or the interconnection variables, and (3) computation of the interconnection variables for the next time horizon.

The first phase estimates the error in the interconnection variables and is the same for all four algorithms. Sub-systems take the interconnection variables provided by the co-ordinator as inputs and produce outputs to form a new interconnection variable. The co-ordinator then compares the assumed and the calculated interconnection variables to obtain an error estimate. This error estimate is used as the basis of adjustment in phase 2.

The following norm is used to determine the difference between the assumed and the calculated interconnection variables.

(*Definition 1*) The norm of a vector of functions f over an interval $[0, T]$ is the maximum vector norm of $f(t)$ over $[0, T]$ with $0 \leqslant t \leqslant T$, i.e.

$$\|f\| = \max_{t} \|f(t)\| \text{ for all } t \in [0, T]$$

$$\text{with} \quad f(t) \equiv (f_1^T(t), f_2^T(t), \ldots, f_n^T(t))^T$$

If $w^{(0)}$ and $w^{(1)}$ are two vectors of functions of the assumed and the calculated interconnection variables, a relative error estimate can be calculated as

$$ERR \equiv \frac{\|w^{(1)} - w^{(0)}\|}{\|w^{(1)}\|} \tag{3.1}$$

The second phase adjusts either the interconnection variables or the size of the time horizon. If the error estimate calculated in Phase 1 is not acceptable, the co-ordinator will automatically perform the required adjustment. For Algorithm 1 given below, the co-ordinator recomputes the interconnection variables for the present time horizon taking as input the most recent calculation. For the other algorithms, the co-ordinator reduces the size of the present time horizon in order to reduce the error estimate. The formulae for time horizon adjustments are given with the algorithms.

The third phase of the algorithms is to compute the interconnection variables for the next time horizon. The co-ordinator uses either a polynomial extrapolation method or a Newton type algorithm to calculate the interconnection variables from past data.

In general, the value of the interconnection variable $w(t)$ over any time horizon can be expanded about $w(0)$ with $0 \leqslant t \leqslant TH$. By Taylor's theorem:

$$w(t) = w(0) + tw'(0) + \cdots + \frac{t^p}{p!} w^{[p]}(0) + \frac{t^{p+1}}{(p+1)!} w^{[p+1]}(\xi) \tag{3.2}$$

where $w^{[p]}$ is the pth order derivative of w, and ξ lies between 0 and t. If a pth order extrapolation method is used, $w(t)$ will be approximated by $w_e(t)$ with

$$w_e(t) = w(0) + tw'(0) + \cdots + \frac{t^p}{p!} w^{[p]}(0) \tag{3.3}$$

where the values of $w'(0)$, $w''(0)$, ..., $w^{[p]}(0)$ are calculated from the values of w near the end of previous time horizon or from several previous time horizons. Each element of $w(t)$ for the present time horizon is extrapolated from its past values and is independent of the other elements of the interconnection variable. Equation (3.3) is also used to calculate an initial interconnection variable in Algorithm 1.

The Newton type algorithm calculates new interconnection variables by solving a set of linear integral equations. These integral equations describe the local response of the sub-systems to changes in their inputs. The kernel of the integral is a step response function which is obtained by perturbing the initial guess for the interconnection variables by a small step function. That is, the sub-system outputs are computed first for a nominal interconnection function, which is usually obtained by constant extrapolation, and then by a function which differs by a constant from the nominal. The difference between the sub-system responses for the two different

inputs is the local step response of that sub-system. Collected together the sub-system step responses form the kernel of the integral equations which are solved by the co-ordinator to obtain the new interconnection variables. For an n-vector interconnection variable, an $n \times n$ matrix of step responses $S(t)$ will be calculated. The ijth element ($i \neq j$) of $S(t)$ is the step response of the ith element of the interconnection variable w to the jth element of w. If element w_i is not related to element w_j through a sub-system, $S_{ij}(t)$ is zero. Once the matrix of the step responses is determined, the co-ordinator solves the following integral equation to calculate a new interconnection variable $w^{(1)}$.

$$w^{(1)}(t) = G(x(w^{(0)}, v), w^{(0)}(t), v(t), t) + \int_0^t S(t-\tau)\, d(w^{(1)}(\tau)); \qquad 0 \leqslant t \leqslant TH \tag{3.4}$$

3.1. *Algorithm 1. Direct substitution*

Phase 1. Estimation of error

(1-*a*) Computation of outputs

Sub-systems use the interconnection function $w^{(0)}$ over $[0, TH]$ provided by the co-ordinator as inputs and produce outputs to form a new interconnection function $w^{(1)}$.

(1-*b*) Error estimate

The co-ordinator compares the two interconnection functions, $w^{(0)}$ and $w^{(1)}$, to calculate an error estimate ERR.

$$ERR = \frac{\|w^{(1)} - w^{(0)}\|}{\|w^{(1)}\|}$$

Phase 2. Adjustment

If $ERR > E_{\max}$, replace $w^{(0)}$ with $w^{(1)}$ and go to Phase 1, otherwise increment simulation time by TH and continue.

Phase 3. Computation of the new interconnection variables

The co-ordinator calculates a new interconnection function $w^{(0)}$ by polynomial extrapolation and returns to Phase 1.

3.2 *Algorithm 2. Extrapolation methods*

Phase 1. Estimation of error

Same as Algorithm 1

Phase 2. Adjustment

If $ERR > E_{\max}$, reduce TH and go to Phase (1-*b*), otherwise, increase TH if necessary, increment simulation time by TH and continue. The formula for time horizon adjustment is:

$$(TH)_{\text{new}} = (TH)_{\text{old}} \cdot \left[\frac{E_{\max}}{ERR}\right]^{1/(p+1)}, \text{ for a general system,}$$

and is

$$(TH)_{\text{new}} = (TH)_{\text{old}} \cdot \left[\frac{E_{\max}}{ERR}\right]^{1/(p+2)}, \text{ for a purely dynamic system,}$$

where p is the order of accuracy of the extrapolation method. Derivation of this formula can be found in Lui (1983).

Phase 3. Computation of the new interconnection variables
The co-ordinator calculates an interconnection function $w^{(0)}$ over $[0, TH]$ by polynomial extrapolation and goes to Phase 1.

As indicated by the formula for time horizon adjustment, a pth order extrapolation method is of $(p+1)$th order accuracy for a general modular system. When the system is purely dynamic (i.e., $q_w = 0$), the order of accuracy improves by one.

3.3. *Algorithm 3, The Newton type method*

Phase 1. Estimation of error
Same as Algorithm 1

Phase 2. Adjustment
If $ERR > E_{\max}$, reduce TH and go to (1-b), otherwise, increase TH if necessary, and move on to the next time horizon and continue. The formula for time horizon adjustment is:

$$(TH)_{\text{new}} = (TH)_{\text{old}} \cdot \left(\frac{E_{\max}}{ERR}\right)^{1/2} \quad \text{for a general system,}$$

and is

$$(TH)_{\text{new}} = (TH)_{\text{old}} \cdot \left(\frac{E_{\max}}{ERR}\right)^{1/3} \quad \text{for a purely dynamic system.}$$

Phase 3. Computation of the new interconnection variables

(3-a) Co-ordinator calculates an initial interconnection function $w^{(0)}$ with $w^{(0)}(t) = w^{(0)}$ for $0 \leqslant t \leqslant TH$.

(3-b) Sub-systems take $w^{(0)}$ as input to produce an output $q(x(w^{(0)}, v), w^{(0)}, v, t)$.

(3-c) The co-ordinator makes a constant perturbation on $w^{(0)}$ and calculates a matrix of step responses $S(t)$ for $0 \leqslant T \leqslant TH$.

(3-d) The co-ordinator solves the following equation for a new interconnection function $w^{(1)}$, and returns to phase 1.

$$w^{(1)}(t) = g(x(w^{(0)}, v), w^{(0)}(t), v(t), t) + \int_0^t S(t-\tau)\, d(w^{(1)}(\tau)) \qquad 0 \leqslant t \leqslant TH$$

The Newton method is of second order accuracy for a general modular system as indicated by the time horizon adjustment formula. When the system is purely dynamic ($q_w = 0$), the method becomes third order accurate.

The above Algorithm is actually only an approximation to an exact linearization since the step response function given above and in eqn. (3.4) should actually be evaluated as $S(t, \tau)$ rather than $S(t-\tau)$ as shown. That is, an exact linearization generally results in a time varying system rather than a constant coefficient system as is implied by $S(t-\tau)$. However, the computational effort required to compute a step response for all τ between zero and the time horizon, TH, is prohibitive. There-

fore we approximate $S(t, \tau)$ as: $S(t, \tau) \cong S(t - \tau, 0)$. That is, we use only only the initial step response. This approximation does not change the order accuracy of the method. Further, Algorithm 3 gives the exact solution for any time horizon when the original system is linear and constant coefficient.

3.4. A combined algorithm

The first step of Algorithm 3 (Newton method) is actually a step of the constant extrapolation method. Since the results of constant extrapolation may be acceptable, it seems reasonable to check the error at the end of step (3-*a*) before linearization. If the error estimate of constant extrapolation is small enough, we can proceed to the next time horizon directly without linearization. Only when the results of constant extrapolation are not acceptable, is the linearization needed. This may save significant computational effort. Thus, a possibly more efficient version of Algorithm 3 can be constructed by introducing an error check step after step (3-*b*).

3.5. Convergence and stability

A detailed discussion of convergence and numerical stability for the various coordination algorithms can be found in Liu (1983), and Liu and Brosilow (1983). The following summarizes the result of the aforementioned references.

Iteration by substitution (Algorithm 1) converges to the solution of 2.1 and 2.2 if (1) f and g satisfy Lipschitz conditions with respect to both x and w, (2) the continuous system is stable and, (3) the Lipschitz constant with respect to the interconnection variables for the purely algebraic portion of system (i.e. eqn. 2.2) is less than one. The foregoing conditions are quite mild. Indeed, if there are no purely algebraic relationships between the interconnection variables, then the algorithm converges for any problem which can be shown to have a stable unique solution. In addition to the foregoing, one can prove that for linear systems obeying the same conditions as given above, iteration by substitutions is numerically stable for any finite time horizon.

Algorithm 2 (extrapolation) is convergent for non-linear modular systems if the extrapolation method is a first or higher order polynomial. Constant extrapolation is a zeroth order polynomial and so may not be convergent. That is, the results of a constant extrapolation need not approach the true solution even when the time horizon approaches zero. The numerical stability of extrapolation methods when applied to linear modular systems requires that the magnitude of the roots of a characteristic equation be less than one. This characteristic equation is similar to the characteristic equation which results when multistep methods are applied to the numerical integration of linear differential equations and requires the evaluation of the determinant of a sum of matrices. It is therefore too complex to yield insight into any but the simplest of problems.

Finally, the Newton type algorithm is convergent for non-linear modular systems and is stable for any time horizon when the algorithm is applied to linear sub-systems. Indeed, the Newton algorithm is exact when the sub-systems are described by linear constant coefficient differential equations. It is not even necessary for the linear sub-systems to be stable.

4. Small scale numerical experiments

The various co-ordination algorithms are tested on simple problems which are selected so as to demonstrate the strengths and weaknesses of these algorithms. Problem (*i*) below is a linear, constant coefficient system which is highly oscillatory. The eigenvalues of the system are $-8{\cdot}8889$ and $-0{\cdot}05555 \pm 0{\cdot}33078i$.A The overall system is therefore stable as is each sub-system. Problem (*ii*) is a non-linear system which has a simple solution (i.e. $x(t) = w(t) = e^{-t}$, $y(t) = e^{t}$) and yet has both algebraic and dynamic coupling. Problem (*iii*) poses a severe test of the convergence properties of an algorithm, since the sub-systems are non-linear and interact so as to yield a limit cycle. Problem (*iv*) was developed by F. T. Krogh (Gear 1971, p. 218) to test ordinary integration algorithms which treat stiff differential equations. The stiffness of the non-linear problem can be adjusted by adjusting the parameters β_i, and the number of sub-systems N. For convenience we have chosen $N = 4$.

In order to reduce computational complexity the following procedure was followed for each example:

(*a*) Each sub-system uses a fixed-step Runge–Kutta algorithm for integration. The step size taken by each sub-system has been made small enough so that the results of the sub-system integrations can be considered perfectly accurate.

(*b*) The same step size is taken by each sub-system so that no interpolation is needed in the determining step responses, and in solving the integral-algebraic equations.

(*c*) Numerical quadrature is carried out using the trapezoidal rule on the same mesh as that of the sub-systems.

Problem (*i*)

sub-system 1: $\dot{x}_1 = -2x_1 - 5{\cdot}56x_2 + x_3, \qquad x_1(0) = 1$
sub-system 2: $\dot{x}_2 = -3x_2 + 5x_3 \qquad\qquad x_2(0) = 1$
sub-system 3: $\dot{x}_3 = -4x_3 + x_1 + 4{\cdot}78x_2, \qquad x_3(0) = 1$

Problem (*ii*)

sub-system 1: $\dot{w} = -1/y,\ x = w^2 y, \qquad x(0) = w(0) = 1$
sub-system 2: $\dot{y} = 1/x, \qquad\qquad y(0) = 1$

Problem (*iii*)

sub-system 1: $\dot{x} = -(1 + e^{w})x - 0{\cdot}5(e^{w} - 1), \qquad x(0) = -0{\cdot}1111889$
sub-system 2: $\dot{y} = xe^{w} - 8{\cdot}9y^2 - (2 + 2{\cdot}225y) + 0{\cdot}5(e^{w} - 1),$
$y(0) = 0{\cdot}0323358$
$w \equiv 25y/(2 + y)$

Problem (*iv*)

Overall system: $\dot{y} = UZ - UBW,\ y_i(0) - -1,\ 1 \leqslant i \leqslant N$

where $U_{ij} = \begin{cases} 1/N - 1 & \text{if } i = j \\ 1/N & \text{if } i \neq j \end{cases}$

$w = Uy$
$Z_i = w_i^2$
$B = \text{diag}\,(\beta_i)$

when $N = 4$,

sub-system j is given by

$$y_j = \frac{1}{4}\left[\left(\sum_{i=1}^{N} \beta_i\right) y_j + \sum_{\substack{k=1 \\ k \neq 1}}^{N} \left(\left(\sum_{i=1}^{N} \beta_i\right) - 2\beta_j - 2\beta_k\right) y_k\right]$$

$$+ \frac{1}{8}\left[-\left(\left(\sum_{i=1}^{N} y_i\right) - 2y_j\right)^2 + \sum_{\substack{k=1 \\ k \neq 1}}^{N} \left(\left(\sum_{i=1}^{N} y_i\right) - 2y_k\right)^2\right] \qquad j = 1 \cdots 4$$

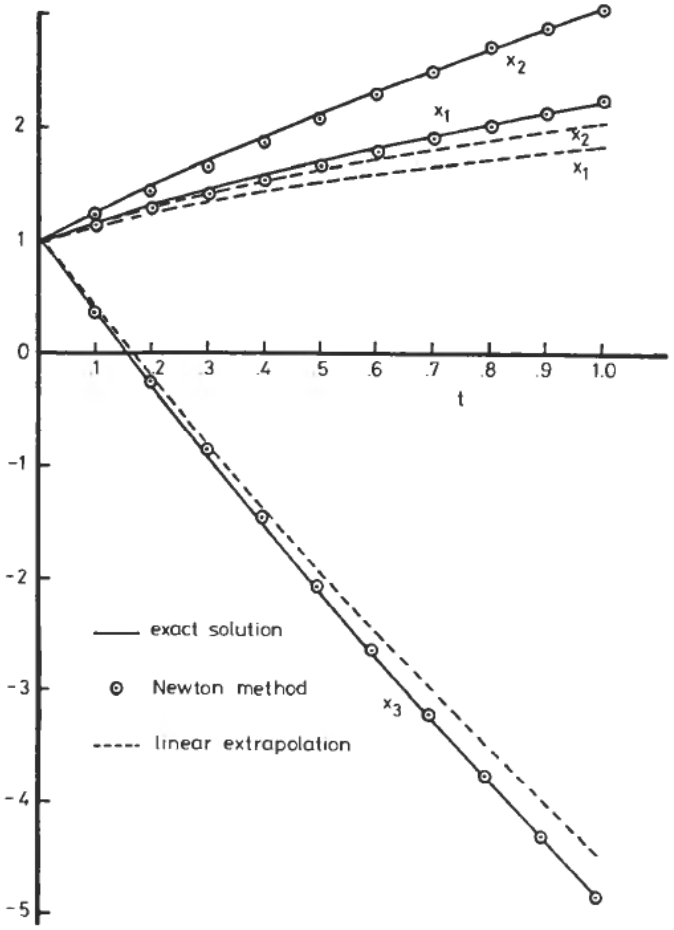

Figure 2. Results of problem (*i*), $TH = 1{\cdot}0$.

4.1. Numerical results for Problem (i)

Figure 2 compares the trajectories generated by the three sub-systems over a single time horizon using linear extrapolation and the linearization methods. A co-ordinator which uses constant extrapolation gives such poor results that they are not included in Fig. 2. Table 1 compares the same algorithms when the time horizon is five. The results in this table confirm the statement made previously, that the Newton method solves a linear, constant coefficient system exactly. The small discrepancies from the true solution are due entirely to discretization errors within the sub-systems.

	$x_1(t)$				$x_2(t)$				$x_3(t)$			
t	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
1·0	−4·851	−4·831	−4·4607	−1·836	3·107	3·095	2·044	1·634	2·243	2·235	1·185	1·468
2·0	−9·538	−9·488	−9·198	−2·220	4·701	4·675	2·667	1·665	3·075	3·058	2·335	1·447
3·0	−12·727	−12·646	−13·837	−2·272	5·632	5·595	3·269	1·667	3·496	3·472	2·841	1·445
4·0	−14·238	−14·139	−18·462	−2·279	5·871	5·829	3·870	1·667	3·503	3·478	3·349	1·445
5·0	−14·088	−13·991	−23·086	−2·280	5·465	5·428	4·471	1·667	3·140	3·119	3·856	1·445

(1) exact solution, (2) Newton method, (3) linear extrapolation, (4) constant extrapolation.

Table 1. Results of problem (*i*) with time horizon = 5·0

	$y(t)$			
t	(1)	(2)	(3)	(4)
0·2	1·221	1·219	1·211	1·220
0·377	1·458	1·452	1·436	1·462
0·530	1·698	1·689	1·663	1·715
0·671	1·956	1·948	1·904	1·977
1·027	2·794	2·786	2·672	2·805
1·309	3·703	3·698	3·480	3·705
1·701	5·477	5·461	4·951	5·430
1·895	6·653	6·653	5·845	6·559
2·09	8·082	8·100	6·957	7·877

(1) exact solution, (2) Newton method, (3) constant extrapolation, (4) linear extrapolation.

Table 2. Results of problem (*ii*).

4.2. *Numerical results for Problem* (*ii*)

Results obtained by applying the various algorithms are given in Table 2. To simplify the content of Table 2, only results for $y(t)$ are included. A total of 9 time horizons are needed to reach $t = 2{\cdot}0$. To reach the same time, the linear extrapolation method needs 34 steps, and constant extrapolation needs 196 steps. Results from the linearization method and both extrapolation methods are also plotted in Fig. 3 for comparison. With the same error tolerance, the results from the constant extrapolation method are much less accurate than the other two methods. This may be due to the accumulation of local integration errors.

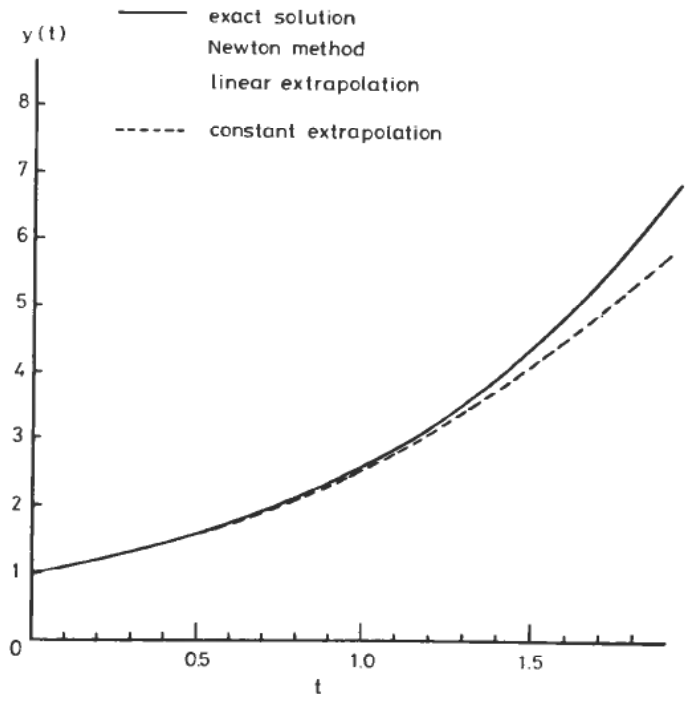

Figure 3. Results of problem (*ii*).

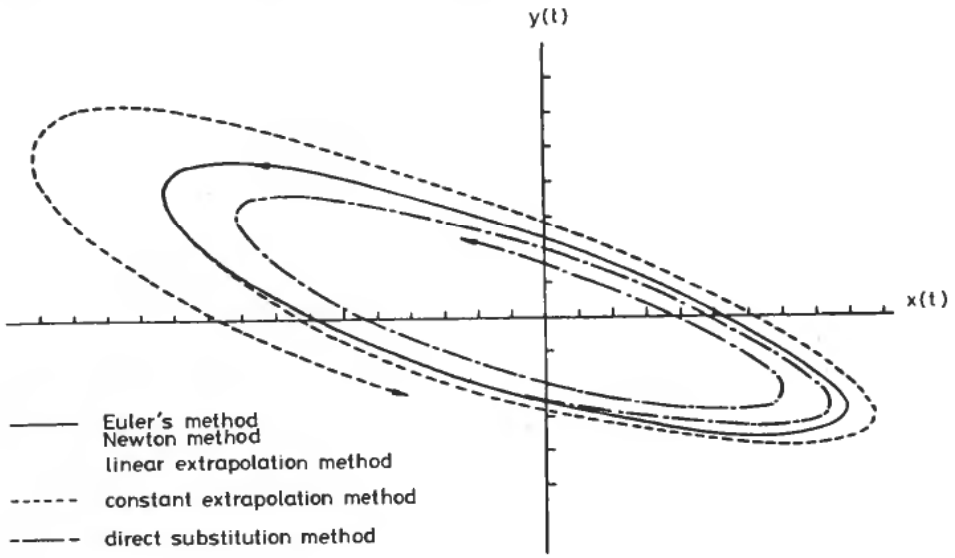

Figure 4. Results of problem (*iii*).

4.3. Numerical results for the limit cycle, Problem (*iii*)

An analytical solution of Problem (*iii*) is not available. A solution obtained by Euler's explicit method with a small step size ($h = 0{\cdot}0001$) is used as reference (see Fig. 4). All three co-ordination algorithms are tested with this problem. The results of the Newton method and to linear extrapolation method are virtually identical to the reference solution obtained with Euler's method. However, it took the linear extrapolation method 58 time horizons to integrate from $t = 0$ to $t = 6{\cdot}0$ with an error tolerance of 0·003. The Newton method took 19 time horizons to integrate over the same time period with the same error tolerance.

Again, using the same error tolerance, the constant extrapolation method diverges from the true solution while taking 252 time horizons to reach $t = 6{\cdot}0$. The results of the iteration method with a fixed time horizon of 0·3 is also shown in Fig. 4 for comparison. The time horizon of 0·3 is chosen to be approximately the average time horizon used by the Newton method. Three to four iterations were needed to 'converge' for each time horizon. The iteration method also diverges from the true solution and decays towards the origin. The probable reason for the divergence is that an insufficient number of iterations were used at one or more time horizons.

In addition to yielding accurate results, the Newton method and linear extrapolation method use relatively large time horizons. Using Euler's explicit method to solve the integrated problem with a fixed mesh requires about 1500 mesh points. Since the linearization method and linear extrapolation method require only 19 and 58 co-ordinator mesh points, there is a lot of integration going on in the sub-systems between time horizons.

4.4. Numerical results for the Krogh problem, Problem (*iv*)

The parameters β_2, β_3, and β_4 were fixed at 8, -10, and 0·001 and $N = 4$. The parameter β_1 was adjusted from 10 to 1000 to study the effect of stiffness of the overall system on the simulation. When $\beta_1 = 1000$, the stiffness ratio of the overall system is about 10^6, while the time constants for the sub-systems are only about

t	$y_1(t)$ exact	$y_1(t)$ Newton method	$y_1(t)$** linear extrapolation
0·050	−1·280	−1·280	−1·281
0·066	−1·388	−1·387	−1·387
0·074	−1·445	−1·445	−1·443
0·087	−1·545	−1·544	−1·541
0·119	−1·821	−1·821	−1·821
0·168	−2·329	−2·328	−2·326
0·236	−3·133	−3·132	−3·127
0·336	−4·198	−4·199	−4·198
0·434	−4·831	−4·832	−4·833
0·532	−5·119	−5·120	−5·118
0·632	−5·229	−5·229	−5·226
0·732	−5·260	−5·261	−5·258
0·832	−5·262	−5·263	−5·261
0·932	−5·255	−5·256	−5·254
1·032	−5·244	−5·246	−5·244

* the maximum time horizon allowed is 0·1.
** interpolations are used to calculate $y_1(t)$ if necessary.

Table 3. Results for Krogh problem with $\beta_1 = 10$ and 0·5% relative error tolerance*

250. Thus, simulating the Krogh problem using a modular approach should be advantageous compared to ordinary integration.

To simulate the sub-systems, we use Euler's method with a step size of 0·001. Table 3 gives the results of Newton method for $\beta_1 = 10$ and a relative error tolerance of 0·5%. To reduce the content of this table, only the results of y_1 are reported for t up to 1·0. For a relative error tolerance of 0·5%, the constant extrapolation method needed a time horizon smaller than 0·0001 to start the simulation and was terminated.

Table 4 compares the Newton method to the linear extrapolation method for stiffness ratios from 10^4 to 10^6 with either a 0·5% or a 1·0% relative error tolerance. Based on these results it appears that the number of time horizons used by the Newton algorithm to achieve a fixed error tolerance increases approximately as the 1/2 power of the stiffness ratio. On the other hand, the linear extrapolation method fails as the stiffness ratio gets very large.

			Number of steps to $t = 1·0$	
β_1	Relative error	Stiffness ratio	Newton's method	Linear extrapolation
10	0·5%	10^4	15	58
10	1·0%	10^4	14	43
100	0·5%	10^5	44	fail*
100	1·0%	10^5	33	90
1000	0·5%	10^6	138	fail*

* Initial time horizon $< 10^{-4}$.

Table 4. Results for Krogh problem from $t = 0$ to $t = 1·0$. $N = 4$, $\beta_2 = 8$, $\beta_3 = -10$. $\beta_4 = 0·001$.

5. A prototype modular simulator for distillation systems

The simulation package consists of four types of units; distillation columns, reboilers, condensers, and control systems. They are identified by the names DIST, RED, CON, and EXTRA respectively. In order to describe the inputs and outputs to a unit all the streams in the system are numbered. A stream can be either a material flow, heat flow, or a temperature. A material flow consists of a flow rate, enthalpy, and composition. Each unit in the simulation is also numbered. The unit numbers are assigned in the order in which the units are input to the program.

The program begins by reading the input data. If an initial state for the simulation has not been specified then the program will guess an initial state and drive the system to a steady state. The simulation then begins from the user's specified initial state or the steady state and ends when the time equals the user's specified finishing time.

The present capacity of the program is

15 components
20 units
10 units of a single type
10 input streams to a unit
10 output streams from a unit
40 streams
40 stages to a distillation unit.

5.1. *Co-ordinator*

The co-ordinator used in the software package is limited to constant and linear extrapolation. The co-ordinator itself selects between linear and constant extrapolation based on which method gives the largest predicted time horizon to meet the specified error tolerance. The procedure used is similar to that used by Gear (1971) to control the order and step size of multi-step integration algorithms. The error (ERR) is estimated from the values of the interconnection variables at the ends of the time horizon. That is

$$\text{Constant extrapolation: } ERR_{(n+1)} = \|W(t_{n+1}) - W(t_n)\| \tag{5.1}$$

$$\text{Linear extrapolation: } ERR_{(n+1)} = \|W(t_{n+1}) - (W(t_n) + \hat{W}'(t_n)(TH)_n)\| \tag{5.2}$$

$$\hat{W}'(t_n) = \frac{W(t_n) - W(t_{n-1})}{(TH)_{n-1}} \tag{5.3}$$

$$(TH)_n = \text{Time horizon from } t_n \text{ to } t_{n+1}$$

Equations 5.1 and 5.3 are used to determine whether or not to accept the computed values, $W(t_{n+1})$, of the interconnection variables at the end of the current time horizon. If the error estimate is four times greater than the error tolerance then the co-ordinator will make each module recompute over the time horizon with a smaller time horizon and constant extrapolation. (The analysis of Liu (1983) indicates that this is probably not a very good policy and that the co-ordinator should be allowed to choose between constant and linear extrapolation even when it is necessary to reduce the time horizon).

To select between constant and linear extrapolation over the next time horizon the co-ordinator estimates the error, E_{n+1}, over the next time horizon and then

chooses the order of extrapolation which will permit the largest time horizon for the specified error tolerance.

Constant extrapolation

$$\hat{E}_{n+1} = \|\hat{W}'(t_n)(TH)_{n-1}\| \tag{5.4}$$

Linear extrapolation

$$\hat{E}_{n+1} = \tfrac{1}{2}\|(\hat{W}'(t_n) - \hat{W}'(t_{n-1}))(TH_{n-1})^2\| \tag{5.5}$$

The time horizon in either case is calculated as

$$(TH)_n = (TH)_{n-1}\left(\frac{E_{\max}}{\hat{E}_{n+1}}\right)^{1/(p+1)} \qquad p = 0, 1 \tag{5.6}$$

Flow rate and heat duty inputs and outputs use only constant extrapolation. They are not included in the co-ordinator's error estimate. This is done because flow rates and heat duties are often coupled by algebraic equations and constant extrapolation is significantly more stable than linear extrapolation for algebraic equations.

5.2. Modules

The present distillation column module uses Ballard's algorithm (Ballard (1979) and Ballard, Brosilow and Kahn (1978). The algorithm treats columns with spatially and temporally varying phase rates, and varying holdups on a tray. The algorithm automatically adjusts the step size to maintain an error tolerance.

The reboiler and condenser modules presently use a constant holdup version of Ballard's algorithm. The condenser is a partial condenser.

The present control system, which is simulated in the module EXTRA is a proportional plus integral control system which manipulates heat flows based on temperature measurements. The integration algorithm used is the trapezoidal rule.

Special controllers, columns, reboilers and condensers can be simulated by modifying the sub-routine which simulates that particular module.

5.3. User input

User input to the program comes in two forms: the input file (or card deck) and user generated Fortran sub-routines. The input file consists of the following sections:

(1) system description
(2) constants
(3) initial state (optional)

User generated sub-routines are for physical property data, distillation column holdups (optional) and input changes to the simulation (optional). The physical property data is entered through the sub-routines KDATA, LIQENT, and VAPENT which describe equilibrium data, liquid enthalpy, and vapor enthalpy respectively. If a variables holdup to a distillation unit is used then statements must be added to the sub-routine HLDUP. If the user wishes to change the value of an input stream during the simulation then statements must be added to the sub-routine SYSFED.

= continuation mark

System Description

DIST (number of stages FEED List1 PROD list2 list1 T list1)

REB
CON IN list3 LIQ list4 VAP list4 Q list4 T list3

EXTRA IN list3 OUT list2

list 1 = (stream number, stage number) ...
list 2 = (stream number, stage number, L or V or none – L) ...
list 3 = (stream number) ...
list 4 = (stream number)

Constants — CONST

1) NAME = real 2) NAME (index) = real 3) NAME (index) = list 5
index = stream or unit number
list5 = component1, component2, ..., componentNCOMP

1) NAME = DT, ERR, FIN, HINIT, NCOMP, START, PRINT
2) NAME = CONHLD, REBHLD, DISTHLD, EXTPRM1, EXTPRM2, EXTPRM3, PRES, STRM, STRMEN, QSTRM, TEMP
3) NAME = STRMX

Initial State — INIT

REB or CON	EXTRA
X = list5	real number
T = temperature [opt.]	

DIST
liquid vapor, list5, [temperature, opt.]
⋮ ⋮ ⋮ ⋮

Table 5. Summary of input notation.

The input has the following form

system description input
CONST
constant input
INIT
initial state input] OPTIONAL

A summary of the input is in Table 5.

The system description input describes the inputs, outputs and interconnection between units. A unit is described by a name which specifies the type of unit followed by a list containing the input and output stream numbers.

The constant section of the input sets values to streams, holdups, pressures and other constants. The streams and units must first be defined in the system description section.

The initial state input sets the initial state to all the units in the simulation. A simulation can be run without initial state input. If no initial state is specified the program will generate a steady state to begin the simulation.

5.4. Output

The output of the program is in tabular format. A summary of the input is printed followed by the initial or steady state. Either the stream values or the entire state of the simulation can be displayed at the print interval.

6. Simulation of the dynamics of a crude unit

6.1. Crude unit model

An atmospheric refinery crude unit generally refers to a system which consists of a main distillation column, several side-strippers, a fired heater, and a number of pumparounds and condensers. A crude unit is the first major processing unit in a refinery. It serves to separate the crude oil into 4 to 10 different boiling point fractions. Approximately 1% of the crude oil charged to the crude unit is used as energy (Wade *et al.* 1962).

An example of a typical crude unit, supplied by the Standard Oil Company (Ohio), is depicted in Fig. 5. It consists of a main fractionating tower and four side-stripper units. Reflux is provided by four pumparound streams. Preheated feed enters near the bottom of the main column, and steam is added at the bottom of the tower (the residual tank). The lower two side-strippers utilize steam as a separating agent while the heavy and light kerosene strippers have reboilers.

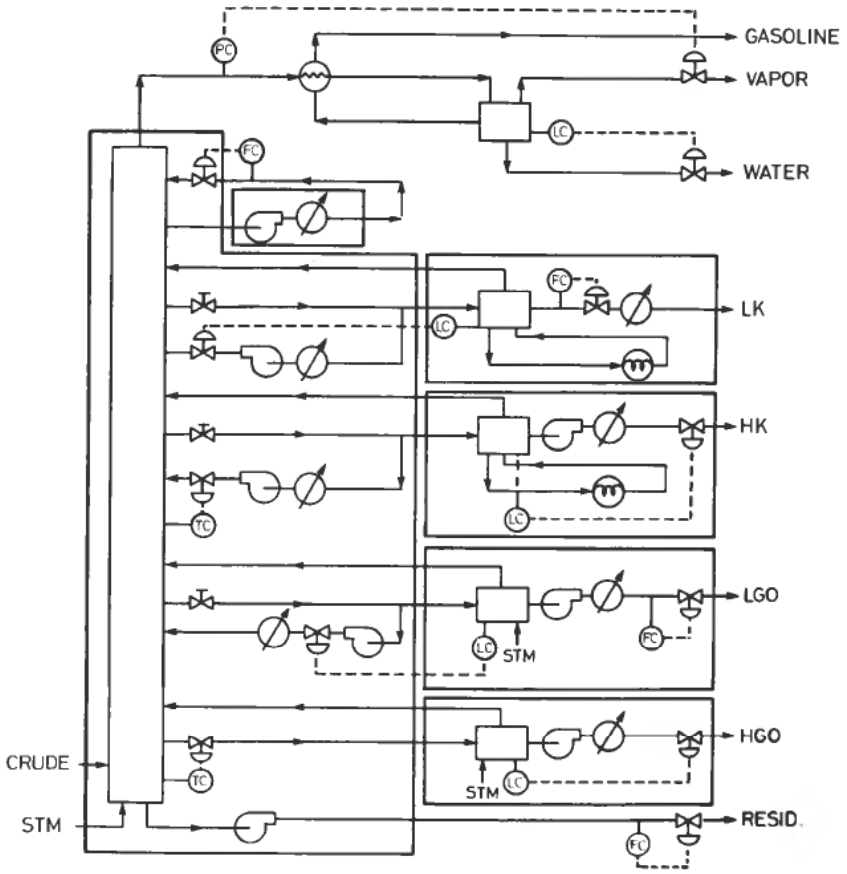

Figure 5. SOHIO atmospheric crude unit.

The physical characteristics of the crude oil and the steady-state operating conditions of the crude unit, including intermediate column temperatures and product specifications, were also furnished by Sohio.

Figure 5 depicts the model of the crude unit as simulated. Each of the blocked areas represents one sub-system. The model consists of five distillation columns, the main column having twenty-five equilibrium stages and each of the four side-strippers have four stages. The top pumparound is modeled simply as a heat exchanger with a controlled amount of heat removal. Four constant side draws leave the main tower to the side-strippers, which have fixed reboiler duties or stripping steam addition. The remaining three pumparound units are modeled as constant heat removals taken directly out of the main column. Appendix I gives further information on the crude unit.

The algorithm used to simulate the distillation columns was developed by Ballard *et al.* (1978). The algorithm uses a second order semi-implicit integration technique with variable step size adjustment. The basic model assumptions are: the liquid on each tray is perfectly mixed and of uniform temperature; there is negligible vapor holdup on the trays; total equilibrium exists on each tray; tray hydraulics are described by the Francis weir equation; and only one liquid and one vapor phase exist on a tray. The pumparound model equations are integrated using the same double-step method employed in the distillation algorithm.

6.2. Initialization

The first step in simulating the crude unit is in initializing the simulation. Starting values for all the state variables in the simulation are needed. Typically, the starting values are chosen to correspond to a known operating condition of the simulated unit. For the atmospheric crude unit, Sohio had supplied specifications for the feed, product streams several column temperatures, and condenser and pumparound heat duties, but not a complete set of steady state operating data. Thus initialization of the crude unit simulation to correspond to the specified data required solving for the steady-state operating conditions from the specifications.

The first approach at initializing the simulation consisted of guessing the columns' compositions equal to the crude oil composition and temperature, setting all the heat duties and side-draw rates to their design conditions, and then running the simulation to steady-state. This approach was not successful. The distillation algorithm developed by Ballard *et al.* (1978) assumes that on every distillation tray, a liquid and vapor phase exist which are in perfect equilibrium with each other. During the start-up of a distillation column, transient conditions may arise in which only a single phase exists. To avoid this condition in the simulation, the crude unit was initialized one column at a time, similar to an actual crude unit start-up.

To initialize the crude unit in a start-up fashion, it is necessary to interact with the simulation in the same way an operator would interact with an actual unit. The modular structure of the simulation proved well suited for this task. Because of the modular design, it is easy to simulate the main column initially and introduce the side-strippers one by one into the simulation. The modular approach has an additional benefit. Upon introduction of an additional column or major change in heat duty or side-draw rate, if an infeasible condition occurs in any of the sub-systems during a time horizon, the inputs to that sub-system can easily be changed. This allows the simulation to proceed past an ill-defined condition instead of halting.

The following pseudo start-up procedure was used to initialize the crude unit.

	CPU (min)
1 Main tower start-up without steam	10·80
2 Addition of main tower stripping steam	5·32
3 Addition of light kerosene cut	3·18
4 Addition of heavy kerosine cut	4·54
5 Addition of light gas oil stripper	5·93
6 Addition of heavy gas oil stripper	7·18
7 Final adjustments	7·60

Table 6. CPU utilization during initialization changes

First, the main fractionating tower was run to steady-state by guessing all its tray compositions equal to the crude feed composition, and then simulating the column at almost complete reflux. Next, the feed flow rate was increased and the pump-around heat duties adjusted closer to the given design conditions. After the main column was brought to steady-state, the side-strippers were introduced one at a time, starting with the lower cuts and working upwards. Finally, all the flow rates and heat duties were adjusted to the given conditions and the crude unit was run to steady-state.

In Table 6, the CPU requirements for initializing the crude unit are given. Approximately 20% of the computer usage was due to the co-ordinator and input/output requirements.

The dynamic simulation initialization was carried out on a DEC VAX-11/780 computer. Typically, after a major operating condition change, approximately 6 min of 11/780 time is required to accurately simulate an hour of the crude unit's dynamic response.

6.3. Comparison of extrapolation methods

A comparison of the accuracy of the two extrapolation methods, as applied to the crude unit, is shown in Fig. 6. Extrapolation errors are depicted for 40 minutes of the crude unit's response to a -10% change in the top pumparound's heat duty. The time horizon is held at a constant 1 minute for the entire simulation, for both methods. Immediately after the change in the pumparound's duty, constant extrapolation of the interconnections results in a smaller error. Throughout the rest of the simulated response, linear extrapolation is more accurate.

During initialization of the crude unit, the co-ordinator generally chose to use constant extrapolation. The criteria for the determination of the extrapolation method is given by the following equations:

If $THl > THc$, use linear extrapolation.

$$Err = \sum \left| \frac{Ui - Wi}{Ui} \right| \tag{6.1}$$

where: Wi — actual interconnections
Ui — extrapolated interconnections

$$THc = THc \cdot (Tol/Errc) \tag{6.2}$$

$$THl = THl \cdot (Tol/Errl)^{1/2} \tag{6.3}$$

THc = Time horizon for constant extrapolation
THl = Time horizon for linear extrapolation

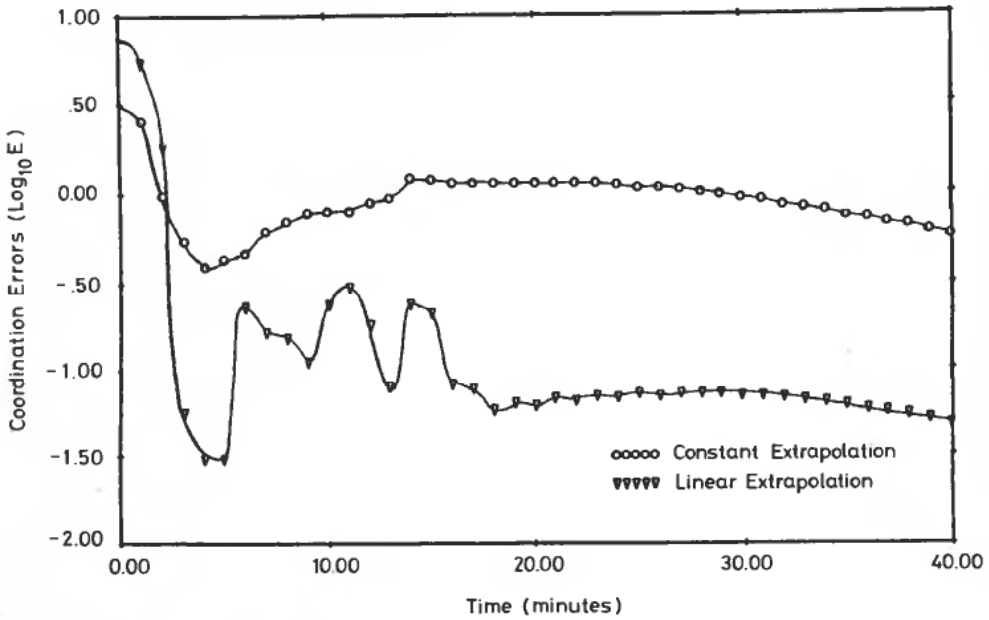

Figure 6. Linear vs. constant extrapolation for a dynamic response of the crude unit.

A difficulty with the above criteria is its use in conjunction with a constraint on the maximum size of a time horizon. If the maximum time horizon is chosen such that both extrapolation errors are small (*Errc* and *Errl* $\ll$ *Tol*), constant extrapolation will be preferentially chosen. For example, if $Errc = 0{\cdot}1 \cdot Tol$, then the linear extrapolation error (*Errl*) needs to be ten times smaller than *Errc* for linear extrapolation to be chosen.

Another observation made during several simulations runs is that the error definition given in eqn. (6.1) tends to be rather sluggish in responding to rapid changes in the interconnections. Because the error is defined as the sum of all the interconnection errors, the co-ordinator does not respond to large errors in a few interconnections. In order to spot sharp changes in a single interconnection, without having to wait until the entire system simulation is affected, the co-ordination error definition was redefined as simply the largest single co-ordination error, c.f. eqn. (6.4)

$$Err = \underset{i}{\text{maximum}} \left| \frac{Ui - Wi}{Ui} \right| \tag{6.4}$$

6.4. Atmospheric crude unit simulation

After initializing the crude unit, corresponding to the design conditions supplied, the product qualities did not match up with the specifications for the product streams. Therefore, product specifications were used to calculate a True Boiling Point (TBP) of a new feed.

To test the utility of the simulator as a steady-state simulator, the crude unit was simulated for a change in feed composition corresponding to the calculated TBP curve. The feed was introduced to the main fractionator as a step change. Immediately, because of the increased light ends, the stage above the feed ran dry. The column pumparounds were adjusted to avoid the transient problem and were slowly

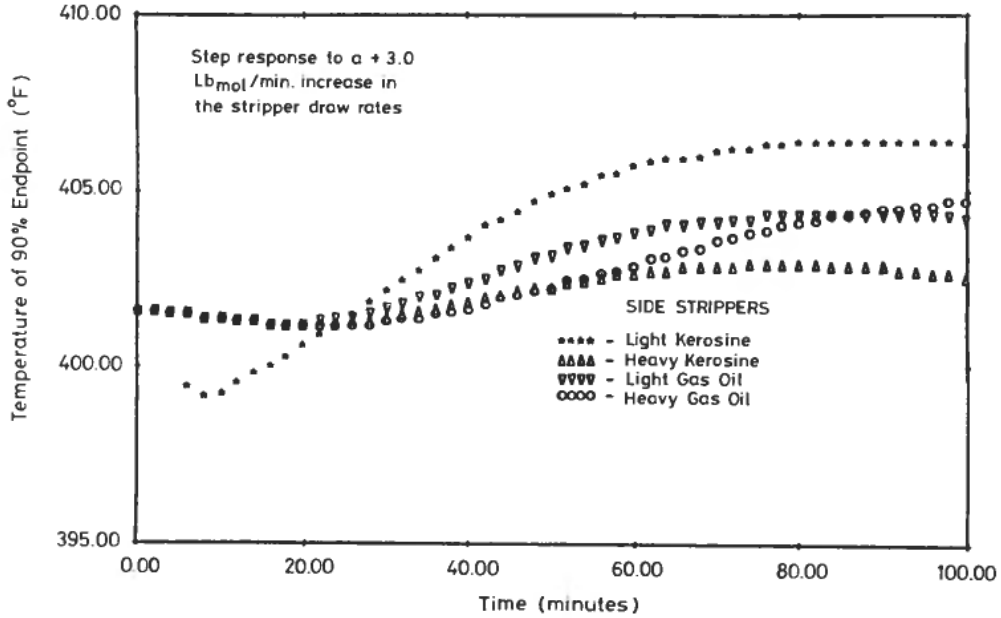

Figure 7(*a*). Light kerosine stripper 90% endpoint.

reset to their steady-state levels. After the run was complete, the new product qualities were found to correspond well with the given design conditions. The total CPU usage for the run was 7·32 min.

A second example of the simulator is shown in Fig. 7. For many multi-variable distillation control schemes, the composition and or temperature response data is needed for changes in side draw rates (DiBiano 1981). The dynamic response of the

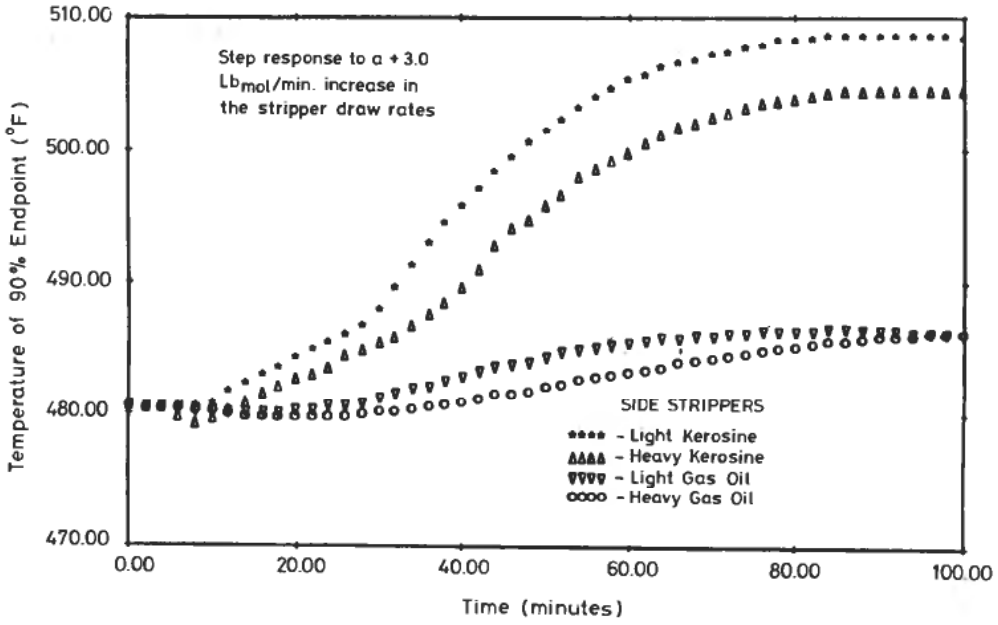

Figure 7(*b*). Heavy kerosine stripper 90% endpoint.

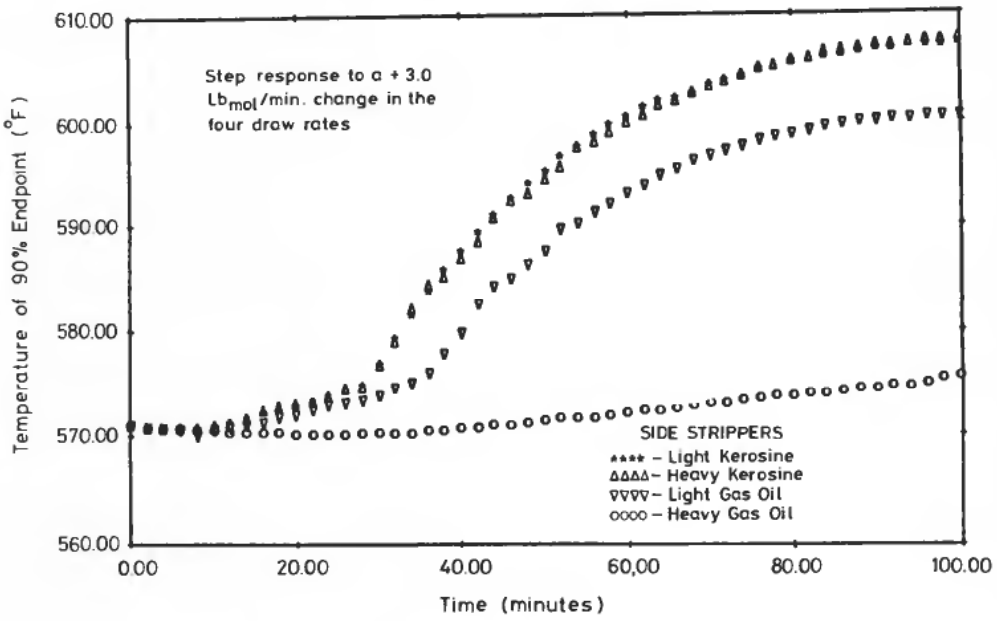

Figure 7(*c*). Light gas oil stripper 90% endpoint.

crude unit to changes in each of the four side-stripper draw rates are shown in Fig. 5. The changes are from the nominal flows of 26·66, 44·25, 76·88, and 33·44 pound moles per minute for the light kerosine, heavy kerosine, light gas oil and heavy gas oil draw rates respectively. Each figure shows the response of the 90% end point of each stripper product to changes in the four draw rates.

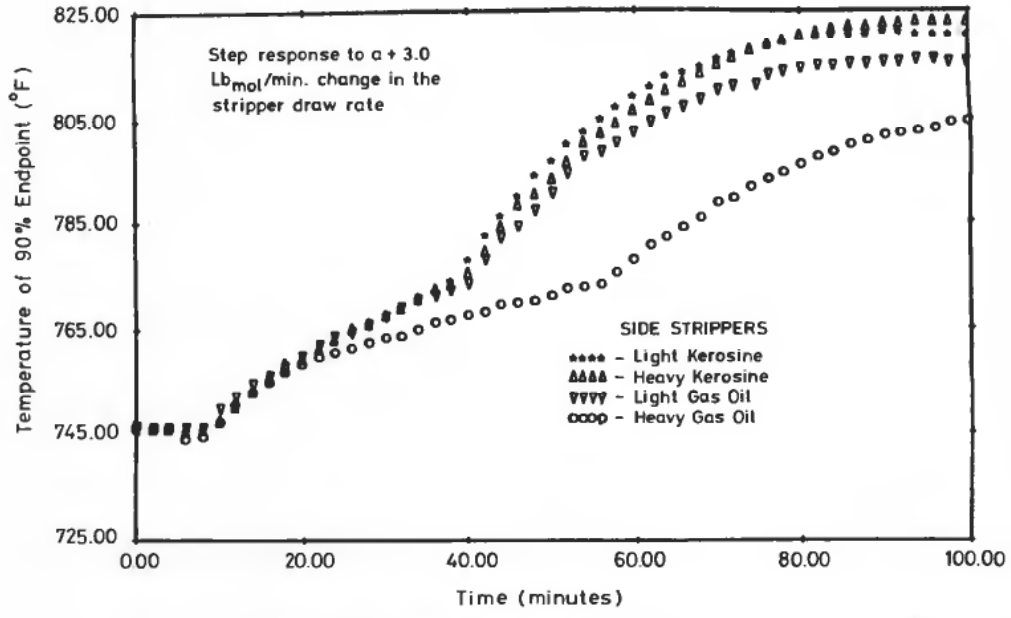

Figure 7(*d*). Heavy gas oil stripper 90% endpoint.

Conclusions

Based on the numerical experiments described in § 4 and 6, it appears that modular integration methods are robust and efficient. These observations are buttressed by the theoretical analysis of the stability and convergence properties of modular integration methods given in § 1 and 2 and in Liu (1983).

A rudimentary completely modular simulator, using constant and linear extrapolation is now available for simulating the dynamics of distillation systems. This simulator has been successfully applied to study the dynamics of a crude unit consisting of five interacting columns. It may be possible to improve the efficiency of the aforementioned simulator by incorporating a Newton type algorithm along with constant and linear extrapolation methods. Also, work is needed to develop the software constructs necessary to enable user friendly simulation on parallel processors.

References

Ballard, D. M. (1979). Dynamic Simulation of Distillation, M.S. Thesis, Department of Chemical Engineering, Case Western Reserve University, Cleveland, Ohio, August.

Ballard, D. M., Brosilow, C. B., and Kahn, C., (1978). Dynamic Simulation of Multicomponent Distillation Columns, presented at the AiChE Meeting, Miami Beach, Florida, November.

Barney, J. R., Ahluwalia, R. S., and Johnson, A. I. (1975a). DYNSY 2.0 User's Manual, University of Western Ontario, London, Ontario, Canada.

Barney, J. R., and Johnson, A. I., (1975b). DYNSYS 2.0 Systems Manual, University of Western Ontario, Canada.

Cook, W. J. (1980). A Modular Dynamic Simulator for Distillation Systems, M.S. Thesis, Case Western Reserve University.

Cook, W. J., and Brosilow, C. (1980). A Modular Dynamic Simulator for Distillation Systems, presented at the 73rd Annual AIChE Meeting, Chicago, November.

DiBiano, R. (1981). Importance of versatile control strategy on crude unit, *Chemical Engineering Processes*,

EXXON (1976). Modular Dynamic Distillation Control Model (MODCOMP), Version 1.0, EXXON Program 36985.

Gear, C. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations*, Section 9.3, (Prentice-Hall, New Jersey).

Gomm, W. (1981). Stability analysis of explicit multirate methods, *Mathematics and Computers in Simulation*, **23**, 34–50.

IBM (1972). Continuous System Modelling Program CSMP 1.3, IBM Program No. H20-0367-3.

Klatt, J. H. (1983). The Application of a Modular, Dynamic Simulator to a Crude Distillation Unit, M.S. Thesis, Case Western Reserve University.

Klatt, J. H., and Brosilow, C. (1984). Simulating the Dynamics of a Crude Unit, presented at the 1984 SCSC Boston, July.

Liu, Y. C. (1983). Development and Analysis of Coordination Algorithms for Interacting Dynamic Systems, Ph.D. Thesis, Case Western Reserve University.

Liu, Y. C., and Brosilow, C. (1983). Modular Integration Methods for Simulation of Large Scale Dynamic Systems, presented at the 75th Annual AIChE Meeting, Washington, D.C., November.

Miller, J. (1978). A Software-Generated Interface for Connecting Independent Continuous System-Simulators, Ph.D. Thesis, Case Western Reserve University.

Perkins, J. D., and Sargent, R. W. H. SPEEDUP: A Computer Program for Steady State and Dynamic Simulation and Design of Chemical Processes. AIChE Symposium Series Vol 78, No 214 (1982).

Sansom, F. J., and Peterson, H. E. (1965). MIMIC A Digital Simulation Program, SESCA International Memo 65-12, Wright–Paterson Air Force Base, Ohio, May.

WADE, H. L., *et al.* (1962). Computer Control of Crude-Vacuum Units for Energy Conservation. 41st meeting of the API Refinery Conference, Los Angeles.
WESTERBERG, A. (1980). ASCENDII An Advanced System for Chemical Engineering Design. Proceedings of the 11th Annual Pittsburgh Conference, Part 2, Systems and Control, ISA.

Appendix

CRUDE DISTILLATION UNIT MODEL

Sub-systems

Main fractionating tower
- 25 stages
- Fixed linear pressure profile
- Feeds enter stages 1, 5, 11, 16, 21, 24
- Output streams off of stages 2, 5, 11, 16, 21, 25
- Heat removal (pumparounds) from stages 6, 12, 17
- Stripping steam enters stage 25

Light and heavy kerosine strippers
- 4 stages
- Fixed constant pressure profile
- Feed enters stage 1
- Output streams off of stages 1, 4
- Heat addition (Reboiler) stage 4

Light and heavy gas oil strippers
- 4 stages
- Fixed constant pressure profile
- Feeds enter stage 1
- Output streams off of stages 1, 4
- Stripping stream enters stage 4

Top pumparound (using condenser sub-routine)
- Fixed heat duty
- Constant holdup

Crude oil composition

31 pseudo components are used to model the crude oil. The equilibrium expression relating y and x is of the form:

$$K_i(T) = (P_{\text{REF}}/P) \cdot \text{Exp}\,(d_{1i}/T + d_{2i} + d_{3i} \cdot T + d_{4i} \cdot T^2)$$

Liquid enthalpies are given by:

$$h_{l_i} = a_{1i} + a_{2i} \cdot T + a_{3i} \cdot T^2 + a_{4i} \cdot T^3; \qquad \text{per component } (i)$$

$$h_{l_j} = \sum_{i=1}^{c} h_{i_j} \cdot x_{ij}; \qquad \text{per stage } (j)$$

The vapor enthalpies are given by:

$$H_{V_i} = b_{1i} + b_{2i} \cdot T + b_{3i} \cdot T^2 + b_{4i} \cdot T^3; \qquad \text{per component } (i)$$

$$H_{V_j} = \sum_{i=1}^{c} H_{V_{ij}} \cdot K_{il} \cdot x_{ij}; \qquad \text{per stage } (j)$$

The liquid's density is given by:

$$\rho_i = \rho_{i60} \cdot (c_{1i} - c_{2i} \cdot T); \qquad \text{per component } (i)$$

$$\rho_j = 1/\sum_{i=1}^{c} (x_{ij}/\rho_{ij}); \qquad \text{per stage } (j)$$

Distillation towers

The material and energy balances describing the unit are:

(1) $$\frac{dM_j}{dt} X_{ji} = L_{j-1} X_{j-1,i} + F_j Z_{j,i} + V_j V_{ji} - L_j X_{j,i} - S_j X_{j,i} - S_j' Y_{j,i} - V_{j-1} Y_{j-1,i}$$

(2) $$\sum_{i=1}^{C} X_{j,i} = \sum_{i=1}^{c} Y_{j,i} = 1$$

(3) $$\frac{d}{dt}(M_j h_j) = L_{j-1} h_{j-1} + F_j h_{F_j} + V_j H_j - L_j h_j - S_j h_j - V_{j-1} H_{j-1} - S_j' H_{j-1} + Q_j$$

(4) $$\frac{dM_j}{dt} = L_{j-1} + F_j + V_j - L_j - S_j - S_j' - V_{j-1}$$

The column trays are taken to be at complete equilibrium.

The holdup on a tray is given by the Francis weir formula:

$$M_j = \rho_j \cdot A_1 \cdot A_2 \cdot ((L_j/\rho_j)^{2/3} + A_3)$$

The tray area A_1 is taken to be double the actual area to account for the actual holdup of the 50 tray column. The equations are solved using the double-step integration method. An error tolerance in the range (0·005–0·002) was found to be effective for the five columns (using the maximum norm definition), with a maximum step size of 1·0 min.

Top pumparound

The material and energy balances describing the unit are:

$$M_\rho \frac{dX_i}{dt} = F_{\text{in}} \cdot Z_i - F_{\text{out}} \cdot Z_i$$

$$F_{\text{in}} = F_{\text{out}}$$

$$0 = F_{\text{in}} \cdot h_{\text{in}} - F_{\text{out}} \cdot H_{\text{out}} + Q$$

Q, M_ρ, F_{in}, F_{out}, are fixed by the user.

Equations solved by use of the double-step integration method. An error tolerance of 0·01 was used for the solution of the compositions (maximum error norm used), with a maximum step size of 2·0 min.