Hybrid and digital computation results in multicomponent distillation simulation

P. F. Stojak*

Churchill College, Cambridge CB3 0DS

Many papers have been published during the last five years on the relative advantage of hybrid computation over digital computation, particularly for simulation of counter-current industrial processes. However, few authors have attempted a quantitative analysis of the times of either technique, nor even a comparison. This paper discusses computation results for a large set of first-order differential equations representing the mass balance of a commercial crude oil distillation unit. The equations are solved on the hybrid computer in the Cambridge Control Engineering Group and on three separate digital computers for comparison purposes. A realistic comparison cannot be based on run-time considerations alone although this is the prime factor; accordingly, development 'costs' are discussed in addition to some techniques available for improving both hybrid and digital solution times. The major conclusion is that, for the particular equation set considered, conventional hybrid computer simulation is marginally faster than digital computer simulation.

(Received August 1972)

1. Introduction

Particularly during the last five years, there has been a good deal of controversy concerning the relevance of hybrid computation in the solution of a wide range of problems. It is now widely recognised however that hybrid computation is a special computing technique suited only to certain classes of problems. For these problems, even though run-time may be considerably reduced in comparison to that achieved by digital computer techniques, there is usually a relatively high preparation cost. Hence the ultimate decision to use a hybrid computer must be based both on the development costs and the total monetary saving expected from hybrid computer production runs.

It is fair to say that, whereas technological advances have been made more or less equally in hybrid and digital computation, the singular lack of success in developing methods to reduce the programming burden currently imposed on hybrid users has had a detrimental effect on hybrid applications. However some progress has been made in automatic patching, static and dynamic diagnostics, and in the development of hybrid compilers (Bekey and Karplus, 1968; Havranek, 1972). So while the future of hybrid computation depends largely on these latter developments, it is most important that more hybrid solutions are quantitatively compared to the equivalent digital solution.

This paper quantitatively compares hybrid and digital solution times for a problem involving the iterative solution of a large set of first-order differential equations. Not only is the discussion pertinent to a wide range of industrial distillation processes, but also to problems involving the solution of parabolic partial differential equations and some two-point boundary value problems. Development costs are noted so that the reader will have a better idea of the total cost involved. Some further improvements in both hybrid and digital schemes are also noted.

2. Development of the equations

The nonlinear equations may be developed with reference to a lumped-parameter model shown schematically in **Fig. 1.** Wilkinson and Armstrong (1957) have previously shown that the set of first-order equations which result from the mass balance is equivalent to the partial differential equation commonly discretised to represent packed distillation columns. Rosenbrock and Storey (1966) advise that such equations are best developed with physical considerations in mind; in this case, the actual plates of the column provide an intuitive partitioning of the column. Nomenclature is given in the appendix. The total material balance may be expressed as

$$\frac{d}{dt} U_j^L = L_{j-1} + V_{j+1} - L_j - V_j + S_j \qquad (1)$$

and each component i may be represented by $(1 \leq i \leq c)$

$$\frac{d}{dt} U_j^L x_{j,i} = L_{j-1}x_{j-1,i} + V_{j+1}y_{j+1,i} - L_j x_{j,i} - V_j y_{j,i} + S_j s_{j,i} \qquad (2)$$

The following assumptions are used in this section:

(*a*) vapour holdup is negligible compared to the liquid holdup.
(*b*) theoretical plate description.
(*c*) no liquid entrained in the vapour.
(*d*) all streams are considered to be single phase.

At the steady state, c equations (2) sum to equation (1) and thus there are only a total of c, not $c + 1$, independent equations. The unsteady state mass balance as derived above introduces the holdup term U_j^L. The holdup is a variable for each plate. Further relationships are required to describe the holdup variation; with the additional mathematical relations there are a total of $c + 1$ independent equations for the mass balance on a single plate. The additional expressions required are described fully in Peiser and Grover (1962), and may be represented by the following functional relationship which has been linearised

$$U_j^L = f_j'(L_j, L_{j-1}, V_{j+1}) \qquad (3)$$

However, for the particular crude distillation column considered equation (3) may be reduced to (4).

$$U_j^L = f_j'(L_j, L_{j-1}) \qquad (4)$$

Substituting (4) into (1) and ignoring second order effects (i.e. assume the changes in liquid flow on plate j are predominantly affected by changes in liquid flow on plate $j - 1$ whereas the changes in liquid flow on plate $j - 2$ have negligible effect on L_j), we obtain

$$\frac{d}{dt} L_j = b_j L_{j-1} - c_j L_j + (c_j V_{j+1} - b_j V_j + c_j S_j) \qquad (5)$$

Hence equation (5) and c equations (2) represent $c + 1$ independent equations, where holdup variation is implicit in (5). This nonlinear set of equations describes the mass balance on each plate.

The ultimate objective of the study is to investigate control schemes operating to maintain the steady state despite various

*Now with Dynamics and Systems Group, Ebasco Services Inc., 2 Rector St., New York

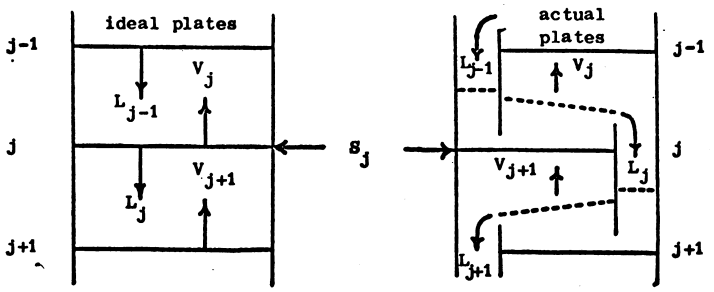

Fig. 1 Plate schematic

disturbances. For this purpose, we may conveniently linearise the mass balances. Although this is not strictly necessary, the hybrid computer implementation is greatly facilitated since a linear set of equations requires fewer non-linear elements.* In addition, the assumption permits faster solution of the equations on a digital computer through application of the transition matrix approach. Each of the variables in the non-linear set (6)

$$\left.\begin{aligned} \frac{d}{dt} L_j &= b_j L_{j-1} - c_j L_j + (c_j V_{j+1} - b_j V_j + c_j S_j) \\ \frac{d}{dt} U_j^L x_{j,i} &= L_{j-1} x_{j-1,i} + V_{j+1} y_{j+1,i} - L_j x_{j,i} - V_j y_{j,i} \\ &\quad + S_j s_{j,i} \end{aligned}\right\} (6)$$

is expanded to two terms of the Taylor series such that

$$z_T = z_{ss} + z$$

The products of perturbational quantities are neglected, and the steady state terms cancel. Substitution of equation (1) into (2) prior to linearisation then reduces the nonlinear set (6) to the linear set of differential equations (7) where the variables are now understood to be perturbation quantities.

$$\left.\begin{aligned} \frac{d}{dt} L_j &= b_j L_{j-1} - c_j L_j + (c_j V_{j+1} - b_j V_j + c_j S_j) \\ \frac{d}{dt} x_{j,i} &= d_j y_{j+1,i} + e_j x_{j-1,i} + f_{j,i} L_{j-1} - g_j y_{j,i} - h_j x_{j,i} \\ &\quad + (k_{j,i} V_{j+1} - 1_{j,i} V_j + m_j s_{j,i} + n_{j,i} S_j) \end{aligned}\right\} (7)$$

Note that the sidestream terms do not occur on every plate although they may represent feeds, refluxes, or withdrawals. The equation set (7) is solved for the following conditions:

$$0 \leq j \leq 20$$
$$1 \leq i \leq 10$$
$$b_0 L_{-1} = e_0 x_{-1,i} = f_{0,i} L_{-1} = d_{20} y_{21,i} = 0$$

The range of the parameter j is determined by the number of theoretical stages required to adequately represent the actual distillation column; the number of components i chosen to describe the crude oil mixture is considered the minimum number which adequately represents the variation of composition throughout the column.

The phase equilibrium between liquid and vapour is assumed to be attained instantaneously. The relationship is described by equation (8).

$$y_{j,i} = K_{j,i}(T_j, P_j)\, x_{j,i} \qquad (8)$$

The K-value, $K_{j,i}$, is assumed to be a function of temperature and pressure but not of composition.

The constraint equation to be satisfied simultaneously at each time step is equation (9).

$$\sum_{i=1}^{c} x_{j,i} = \sum_{i=1}^{c} y_{j,i} + stm_j = 1 \qquad (9)$$

*In general, a linear set of equations does not require non-linear analogue components. However, the hybrid scheme developed in a later section requires the use of multipliers to allow rapid alteration of constants for different components.

As shown in equation (9), the steam exists in the vapour phase only.

In the simulation scheme, the temperature is corrected by forcing the following function to zero.

$$\sum_{i=1}^{c} [K_{j,i}(T_j, P_j)\, x_{j,i}] + stm_j - 1 = 0 \qquad (10)$$

The function has been formed through the combination of equations (8) and (9). Sargent and Murtagh (1969) have shown that if the left hand side of equation (9) is equal to one, so also is equation (8); however, the inclusion of steam in equation (9) means that equation (10) must be used for the temperature correction to account for the effect of steam on temperature.

3. Method of solution

A complete mathematical description of distillation processes includes differential equations of the mass and heat balances, algebraic equations representing phase equilibria, and other equations describing the physical and chemical properties of the distillation plant and crude oil. However, the complete set of equations has a number of degrees of freedom (Gilliland and Reed, 1942; Howard, 1967); that is to say, there are more unknown variables than there are equations. Effectively, this requires that some variables be specified a priori; provision must also be made to correct the initial guess of these variables in the event subsequent calculation indicates inconsistent results.

The heat balance of the column is not germane to the present discussion and is therefore not included in the foregoing development. However, it has been included in the general computation scheme shown in **Fig. 2** to illustrate that it may be used to update the vapour flow rate profile in the tower. Reference to Fig. 2 is invaluable in the following discussion.

Amundson and Pontinen (1958) first discussed the automatic

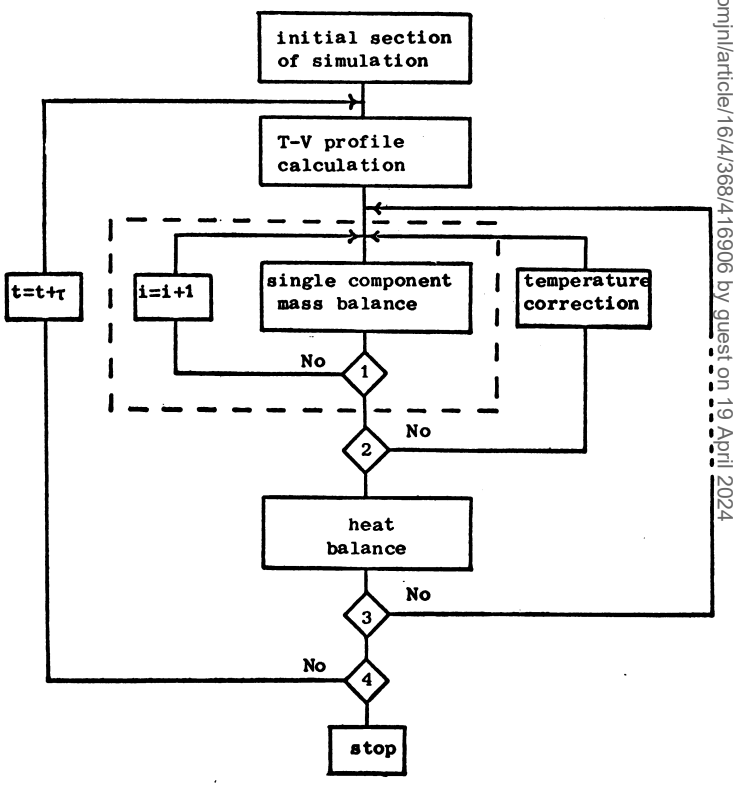

Fig. 2 Computation scheme

KEY

1 $i = c$?

2 $\sum_{i=1}^{c} x_{j,i} \simeq 1$?

3 $V_{jn} = V_{jn-1}$?

4 $t = t_f$?

solution procedure applied on digital computers for distillation problems at steady state in 1958. Derivatives are not normally specified as conditions at the top or bottom of the column and hence we have the classical two-point boundary value problem. In the simulation, the problem is solved in the following manner.

The temperature and vapour flow rate profile throughout the column are estimated from the history of the solution. The mass balance, equation set (7), may then be solved component by component throughout the column (or plate by plate for all components). These equations are solved for a certain time step, τ. Once the mass balance is solved, one must ensure for every plate that equation (9) is simultaneously satisfied. If equation (9) is not satisfied, the temperature profile is assumed incorrect. The temperatures are then corrected using equation (10). Note that, following Amundson and Pontinen (1958), the compositions are normalised prior to the temperature correction to prevent divergence. The mass balances must then be computed iteratively until equation (9) is satisfied. Then the heat balance is used to update the vapour flow rate profile in the column. The solution may proceed to the next time step only after two successive computations have shown that both temperature and vapour flow rates throughout the column have converged. Convergence of these variables is achieved when the two successive iterated values differ by a value smaller than a specified tolerance. After convergence, the state variables are stored on magnetic tape and the time is incremented. The calculation procedure is then repeated for the next time step; the solution terminates when the time reaches the final specified time.

One may now readily observe from Fig. 2 that the solution of the mass balances, equation set (7), is central to the computation. Solution of (7) is required a large number of times since new mass balances must be calculated whenever equation (9) is not satisfied. This occurs often since the correct solution of (7) and (9) depends on a correct temperature profile, which is difficult to estimate. The hybrid computation scheme is a high speed repetitive operation solving equation set (7), signified by the dashed outline in Fig. 2.

Basically, two approaches to the solution have been proposed for digital computers. Generally, plate-to-plate calculations have prevailed; however, the hybrid computer solution involves a component-to-component solution following Amundson and Pontinen (1958) and also Sargent and Murtagh (1969). Details of solutions are discussed in the following sections.

4. Description of hybrid computer scheme

Ruszkay and Mitchell (1966) propose a hybrid scheme which involves sequential plate-to-plate iteration to solve the inherent two-point boundary value problem. The equations representing mass and heat balances are nonlinear; all components (in this case, four) are solved in parallel on the analogue computer and the vapour mole concentrations are forced to sum to unity simultaneously as the solution proceeds. Inter-plate results are stored in the digital computer. Once the boundary conditions have been satisfied, the solution proceeds to the next time step. Ruszkay and Mitchell (1966) report an eight millisecond operate period.

The proposed hybrid scheme differs structurally from that of Ruszkay and Mitchell. This particular scheme has not previously been reported and is based on the following arguments. It is ridiculous to suggest that the entire crude distillation plant may be programmed on an analogue computer; some form of time-sharing is required to solve the large set of equations. Some aspects of time-sharing are discussed by Miura and Iwata (1965). Examination of the total set of equations representing the crude oil distillation unit reveals (*a*) chemical properties cannot easily be represented on the analogue computer due to nonlinearity, dimensionality, and since some are functions of two or more variables, (*b*) a plate-to-plate iteration scheme of the mass balances (7) essentially reduces to simple integration on the analogue computer, and (*c*) time delays may be readily implemented on the analogue computer whereas a plate-to-plate calculation including time delays on the digital computer is more tedious and time-consuming. Note that the differential equation (7) representing the mass balance of each component *i* may be solved independently and sequentially providing temperature variations over the solution period are not excessive.

The component-by-component scheme presented follows previous digital computer studies by Amundson and Pontinen (1958) and Sargent and Murtagh (1969). Time delays are readily implemented on the analogue computer. The number of components (in the crude oil mixture) in the simulation is not limited by the hardware available; instead, the available hardware limits the number of ideal plates which may be accommodated in the simulation. The latter limitation might be relaxed if the dynamics of real plates were better understood, whereas the number of components which adequately describe crude oil mixtures is likely to be numerous. The time step is basically limited by the variation of nonlinear elements in the hybrid simulation and not by inter-plate dynamics as in the paper by Ruszkay and Mitchell (1966).

Both hybrid computer schemes should be advantageous compared to digital integration methods, not only because of the faster solution, but also since analogue integration does not

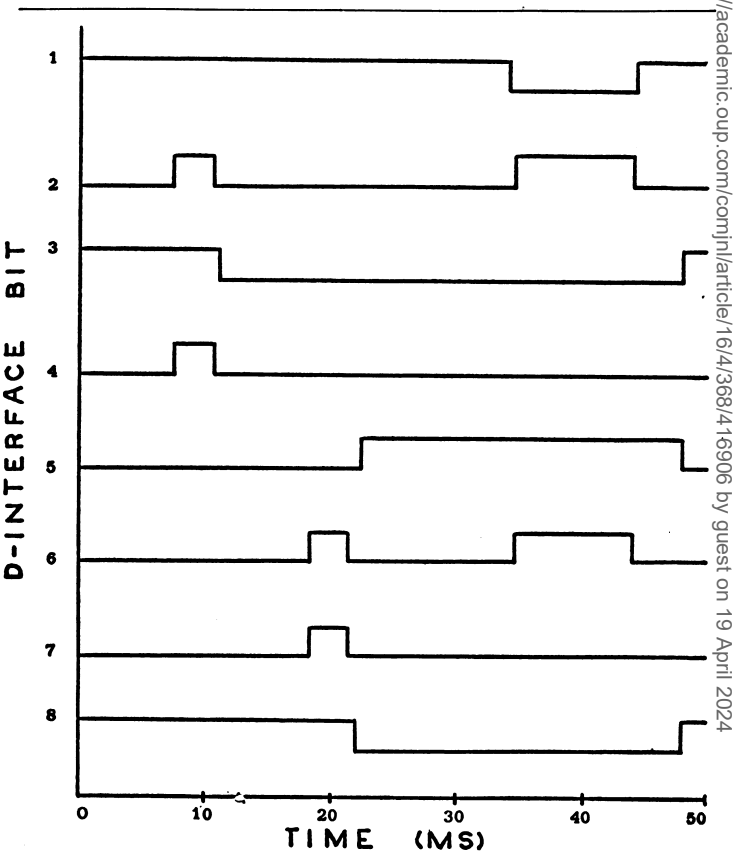

Fig. 3 Hybrid control signal sequence
D1—mode control
D2—track store units 1(A)
D3—relays (A)
D4—track store units 2(A)
D5—D/A switches (A and B)
D6—track store units 1(B)
D7—track store units 2(B)
D8—relays (B)
A—231R V A-machine
B—231R V B-machine
logic level 1—IC mode, track

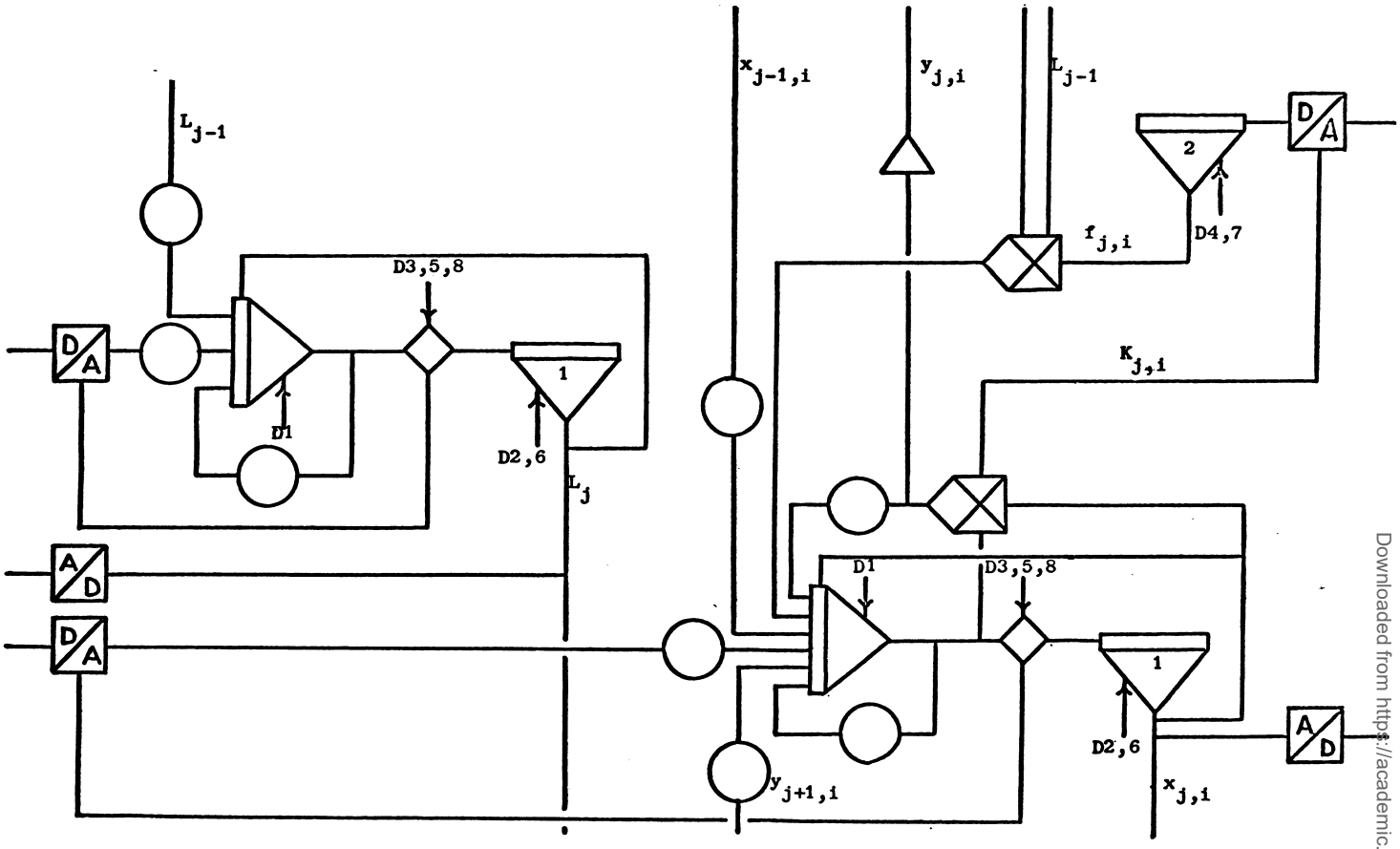

Fig. 4 Analogue computer diagram for single plate

introduce parasitic solutions nor other numerical difficulties (Mah, Michaelson and Sargent, 1962).

Fig. 3 is the logic control sequence diagram for digital computer control of the analogue computer and **Fig. 4** is the analogue computer circuit diagram for each plate (except the top and bottom plates) in the component-by-component hybrid scheme. The same analogue elements may be used for each component in the crude oil mixture; only the digital-to-analogue conversions (DAC) change. These interface values which vary with the crude oil component are the terms in brackets in equation set (7). Among the other terms, only the coefficient $f_{j,i}$ is a function of the crude oil components, although combination of the latter equation in (7) and equation (8) introduces the K-value which is different for each component. The mass balance (7) may be successfully solved using the analogue configuration of Fig. 3. The total equipment required consists of 42 integrators, approximately 160 potentiometers, 63 track/store units, 41 multipliers, 63 DAC channels, 42 ADC channels, and 42 relays (i.e. 95 per cent of that available on two Pace 231RV analogue computers).

The patching for a single plate may appear over-complicated. However, due to the excessive interface requirements all 64 DAC channels would soon have been utilised. The patching (Fig. 4) essentially 'doubles up' on each DAC channel and hence there is the need for relays and track/store units as shown. The operation on each plate becomes obvious when the control logic sequence of Fig. 3 is understood. While the electronic modal switch D1 is in the IC position, the relays are energised via D3 and D8 to transfer the DAC voltages representing initial conditions to the main track/store units (labelled 1 in Fig. 4). These units initially track and then store the IC until D1 switches the integrator to operate mode. In operate, the main track/store units track the state variable and store the final value at the end of the operate period for ADC readout to the digital computer. Throughout the OP period, the coefficient track/store units (labelled 2 in Fig. 4) have held coefficients initially tracked during the IC period. The interface DAC channel track/store units store forcing functions and K-values during the operate period. The length of the operate period is controlled by interrupt from an analogue timing device.

The digital computer uses the final state variables of the previous time step as the initial condition of the subsequent time step once convergence has been achieved as described in Section 3. The solution is similar to the digital method of Mah *et al.* (1962) in that some variables (e.g. K-values, $K_{j,i}$) are held constant over each iteration although this does not preclude alteration of these variables between iterations. Thus the solution relies on a slowly varying temperature profile over each iteration.

The foregoing discussion is not intended to be a detailed resumé of the hybrid computer scheme, but gives an indication of the general procedure as the solution proceeds. For a more informative discussion, the reader is referred to Stojak (1973).

5. Description of digital computer programs

A variety of simulation languages is available, and a large number of integration methods are suitable for the set of equations (7). Choice of the most suitable integration method for a particular set of equations is often formidable. The three following programs were used or developed to give the reader an idea of solution times of a variety of methods on various machines. For a more detailed discussion of numerical methods, the reader is referred to Mah *et al.* (1962), White (1971), Gear (1971), and Rosenbrock and Storey (1966). The three methods chosen are:

(*a*) various integration methods available in CSMP IBM, (1967).

(*b*) Merson variable step-length integration (Christiansen, (1970).

(*c*) integration method based on the transition matrix approach (Gollop, 1971).

Table 1 Hybrid and digital solution times

METHOD	COMPUTER	STEP LENGTH (HOURS)	RELATIVE ERROR	SOLUTION TIME (MILLISECONDS)
Hybrid	Pace 231R-V linked to ICL/ Elliott 4130	0·01	0·05%	10
RKS	IBM 370	0·003	0·1	21
ADAMS	IBM 370	0·0001	—	50
MILNE	IBM 370	0·001	0·1	68
SIMP	IBM 370	0·00005	—	75
TRAPZ	IBM 370	0·00005	—	162
LINSTEP	ICL/Elliott 4130	0·01	—	1,360
MERSON	ICL/Elliott 4130	0·004	0·1	3,047

Definitions
RKS Fourth-order Runge-Kutta integration
ADAMS Second-order Adams integration
MILNE Fifth-order predictor corrector Milne integration
SIMP Simpsons method
TRAPZ Trapezoidal integration
LINSTEP Integration based on transition matrix approach
MERSON Merson integration
Note:
1. Rectangular integration was not accurate for a complete range of fixed step lengths.
2. For variable step length methods, the Table 1 step length refers to the best initial value for step length.
3. Relative error for the hybrid computer refers to the accuracy obtained from the analogue computers (100 volt machines).

These methods were chosen for their ease of use and availability. Prior to discussion of these methods, it is worth noting that the hybrid dynamic solution was verified originally using the Kutta–Merson integration method in the Automatic Programming and Scaling of Equations program (NCC, 1971).

Christiansen (1970) discusses the Merson integration method which compares favourably with classical fourth-order Runge–Kutta techniques (White, 1967). This version is particularly worthwhile since it incorporates a variable step-length algorithm. The algorithm published by Christiansen is used directly in an ALGOL program on the ICL Elliott 4130 digital computer.

Since the equation set (7) and (8) is linear, a method based on the transition matrix solution should yield faster solution times than more general integration methods. Procedure 'STEP' (Gollop, 1971), available as a library routine in ALGOL on the ICL Elliott 4130 system in the Control Engineering Group*, is used in an ALGOL program for timing purposes. This method is particularly powerful for linear sets of equations since the convergence criteria is independent of step-size.

The CSMP simulation language (IBM, 1967) is a general purpose program which has many facilities. In particular, a number of integration routines are available most of which are fixed step-length methods. Thus since the IBM 370 digital computer was used†, one may compare various integration techniques on a modern digital computer. Details of the digital integration methods may be found in the IBM user's manual (1967).

All digital computer solution times reported in Section 6 are based on the accuracy of solution of the hybrid computer. That is to say, for each method used, the choice of time step (and initial time step and relative solution error for variable step-length methods) is such that the accuracy of the digital solution is commensurate with the analogue/hybrid computer solution accuracy. For fixed step-length methods, this is accomplished simply by a series of computations with different step lengths to find the step length which produces the desired accuracy. In the case of variable step-length methods, both the relative error and initial step length affect the accuracy, and solution time. Again in these cases, a number of computations are done to find the right combination of initial step length and relative error to yield the desired accuracy.

*University of Cambridge, England.
†Computing Laboratory, Cambridge, England.

6. Results and discussion

Table 1 summarises the computation results of the hybrid scheme and the various digital integration methods. As shown, the hybrid computer scheme demonstrates the fastest solution time of all the schemes considered. However, the solution times exhibited by the IBM 370 are only marginally slower; in fact, the variable step-length Runge–Kutta method proves to be the fastest integration method of those tested in the CSMP package. It is well known (Mah, Michaelson and Sargent, 1962; Gear, 1971; Gollop, 1971) that there exist superior methods of digital integration particularly in terms of computation time. Note that the IBM 370 digital computer demonstrates much superior CPU times compared with the ICL Elliott 4130 computer although the Runge–Kutta and Merson techniques are roughly comparable (White, 1967) in terms of solution time on similar computers.‡ Table 1 provides a comparison of various solution methods on various computers for the set of equations (7) and (8). The survey is not comprehensive enough to be a definitive result: however, the trends are nonetheless revealing.

Three major results are evident from Table 1. Primarily, digital integration on a digital computer comparable to the IBM 370 may be nearly identical in solution time to hybrid computation. This is true for the equation set (7) and (8) where only a few years ago digital computation could not compete with the hybrid solution. Secondly, combined with an improved knowledge of the limitations of digital integration methods, it is fair to say that in general improvements in digital computers have substantially reduced the hitherto advantages of hybrid computation. The singular lack of success in developing hybrid compilers, methods for automatic patching, and static and dynamic diagnostics has partially contributed to the now marginal effect of hybrid computers. Thirdly, the advantages in terms of solution time of various methods are evident from Table 1.

Some further points are worth consideration. Table 1 demonstrates the computation advantage of an integration method

‡This enables an approximate solution time correlation between these digital computers.

based on the transition matrix approach (STEP) in comparison with the Merson algorithm reported by Christiansen (1970). Applying this approach on the IBM 370 would then mean a solution time of the order of seven milliseconds. It is certainly possible that further improvements over Runge–Kutta integration for the nonlinear set (6) could be implemented following Mah *et al.* (1962). In particular, there are special methods available for solving stiff equations, of which distillation equations are a prime example. The reader is referred to Gear (1971) for the relevant discussion.

In addition, the significance of the computation times of the variable step-length integration methods must not be overlooked. In the first place, a variable step-length hybrid solution poses a lot more difficulties than the equivalent digital methods. Hence, the success of a fixed step-length hybrid technique may well be determined by the maximum permissible real time step length, which is determined by the frequency limitations of the analogue equipment and the method of solution. On the latter point, consider the difficulty of using the Mah *et al.* (1962) approach if one could not consider the flow coefficients constant over some step length. It should be noted that it is certainly possible to improve the solution time of the hybrid scheme. However, the lower computation limit even for modern hybrid computers cannot be accurately reduced much below a one millisecond operate period. Thus it is fair to conclude that hybrid computation for the problem here considered has a speed advantage over digital computation between 5:1 to 2:1 at best.

Finally, the problem of the cost of hybrid schemes must be considered. Whereas approximately two man-weeks were required to successfully implement the CSMP package, approximately three man-months were necessary to successfully complete the hybrid scheme. The ultimate decision to use hybrid computation in preference to digital computation must be based on these pre-production development costs and the expected savings from production runs. Particularly because of the rather laborious and lengthy development time, the overall efficiency of hybrid computation is considerably reduced.

7. Conclusions

These results herald an important turning point particularly for the simulation of large systems of equations. Several years ago, analogue component time-sharing was introduced to overcome the excessive amount of hardware necessary to simulate large systems on a hybrid computer. The hybrid scheme presented here uses approximately 95 per cent of the analogue equipment available on the two Pace 231 RV computers. This equipment is used in a time-sharing mode; nonetheless, the solution time is only marginally better than that of a modern digital computer. Although a hybrid implementation of digital differential analysers may improve solution times substantially, one must conclude that in the class of problem involving the iterative solution of a large set of differential equations hybrid solution time marginally improves the solution time on modern digital computers. Conventional hybrid computers will not be able to compete with further improvements in digital computers unless development costs can be drastically reduced and reliability improved.

Acknowledgement

The author appreciates the hybrid computer facilities made available by the Control Engineering Group of Cambridge University and the data provided by Mobil Services Company Limited. Advice on the hybrid computing scheme from Mobil Services Company Limited and suggestions in the preparation of this paper from Dr. G. W. T. White of the Control Group are gratefully acknowledged. In particular, the constant help and advice given by Mr. B. N. Wootton, without which the hybrid implementation would not have been possible, is most appreciated.

Appendix

Nomenclature

U	molal holdup
L	liquid molal flow rate
V	vapour molal flow rate
S	sidestream molal flow rate
x	liquid mole fraction
y	vapour mole fraction
s	sidestream mole fraction
b, c	coefficients in equation (5)
$d, e, f, g, h, k, l, m, n$	coefficients in equation (7)
K	equilibrium constant (K-value)
T	temperature
P	pressure
stm	steam concentration
τ	time step
f'	functional relationship of equations (3) and (4)
z	general variable

Subscripts

j	denotes plate
i	denotes component
T	total nonlinear variable
ss	steady state
n	refers to the n-th iteration
f	final time of the simulation

Superscripts

L	liquid phase
c	total number of crude oil components

References

AMUNDSON, N. R., and PONTINEN, A. J. (1958). Multicomponent Distillation Calculations on a Large Digital Computer, *Industrial and Engineering Chemistry*, Vol. 50, No. 5.

Automatic Programming and Scaling of Equations, User's manual, The National Computing Centre Limited, Quay Street, Manchester M3 3HU.

BEKEY, G. A., and KARPLUS, W. J. (1968). *Hybrid Computation*, John Wiley and Sons, Inc.

CHRISTIANSEN, J. (1970). Numerical Solution of Ordinary Simultaneous Equations of the 1st Order Using a Method for Automatic Step Change, *Numerical Mathematics*, Vol. 14.

Continuous System Modelling Program, User's manual, IBM Application Program, Program Number 360A-CX-16X, IBM Corporation, 1967.

GEAR, C. W. (1971). *Numerical Initial Value Problems In Ordinary Differential Equations*, Prentice-Hall, Inc.

GILLILAND, E. R., and REED, C. E. (1942). Degrees of Freedom in Multicomponent Absorption and Rectification Columns, *Industrial and Engineering Chemistry*, Vol. 34, No. 5.

GOLLOP, P. J. (1971). *An Algorithmic Approach to State-Variable Estimation*, Ph.D. Thesis, University of Cambridge, Pp. 117-119.

HAVRANEK, W. A. (1972). New Methods of Hybrid Computation and their Application in Chemical Engineering, ICE, *Analog/Hybrid Computation Symposium*, 7 January, 1972.

HOWARD, G. M. (1967). Degrees of Freedom for Unsteady-State Distillation Processes, *Industrial and Engineering Chemistry Fundamentals*, Vol. 6, No. 1.
MAH, R. S. H., MICHAELSON, S., and SARGENT, R. W. H. (1962). Dynamic behaviour of multi-component multi-stage systems. Numerical methods for the solution, *Chemical Engineering Science*, Vol. 17.
MIURA, T., and IWATA, J. (1965). *Study on time-shared analogue computation technique*, Hitachi Central Research Laboratory, Tokyo, Japan.
PEISER, A. M., and GROVER, S. S. (1962). Dynamic Simulation of a Distillation Tower, *Chemical Engineering Progress*, Vol. 58, No. 9, September 1962.
ROSENBROCK, H. H., and STOREY, C. (1966). *Computational Techniques for Chemical Engineers*, Pergamon Press.
RUSZKAY, R., and MITCHELL, E. E. L. (1966). Hybrid Simulation of a Reacting Distillation Column, *Proc. SJCC*.
SARGENT, R. W. H., and MURTAGH, B. A. (1969). The Design of Plate Distillation Columns For Multicomponent Mixtures, *Trans. ICE*, Vol. 47.
STOJAK, P. F. (1973). *Simulation of Multicomponent Distillation Dynamics*, Ph.D. Thesis, Control Engineering Group, University of Cambridge, April 1973.
WHITE, G. W. T. (1967). Digital Simulation Languages for the Solution of Process Control Problems, Proc. IBM Scientific Computing Symposium, Digital Simulation of Continuous Systems, New York, 1967.
WHITE, G. W. T. (1971). Methods of Modelling and Simulating Dynamic Systems, *Measurement and Control*, Vol. 4.
WILKINSON, W. L., and ARMSTRONG, W. D. (1957). An approximate method of predicting composition response of a fractionating column, *Chemical Engineering Science*, Vol. 7.

Book reviews

The Macro Implementation of SNOBOL 4, by Ralph E. Griswold, 1972, 310 pages. (*W. H. Freeman and Company, San Francisco*, £7·00)

SNOBOL 4 is a programming language for manipulating character strings. The original SNOBOL language was developed in 1962 at Bell Telephone Laboratories, Holmdel, purely for internal use. It soon attracted attention outside, and in 1963/4 it was made available to several universities, research laboratories and other organisations. I was one of the first university users, and well remember the excitement of suddenly being able to give an undergraduate course on compiler construction. At that time the folklore said that it took 40 man years of programming effort to implement a compiler, and now that time could be reduced to a few hours. Today, the compiler course based on the use of SNOBOL is a cornerstone of university courses throughout the world. SNOBOL was also quickly adopted for a variety of unusual programming tasks ranging from language translation to musical composition. Bell Telephone Laboratories then allowed the authors of the system to continue its development; and, in the fine tradition of service to the community which has characterised this laboratory, subsequent versions of SNOBOL were made freely available to outside users. It is a great pleasure for me to be able to acknowledge here the friendly help which, like many others, I so often received.

Later versions of SNOBOL were more than a refinement of string manipulation facilties. Two main themes were pursued: first the fundamental basis of string manipulation was investigated. From this investigation the semantic concept of a 'pattern' emerged. Patterns can be constructed and manipulated in a wide variety of ways, and for example, can be returned from procedures. In a recent paper in the *Communications of the ACM*, (Vol. 16, No. 2, February 1973) J. F. Gimpel has pointed out that the power of the SNOBOL pattern analysis facilities is still beyond easy explanation by present day programming language theory. Second, the authors aimed at a translator/interpreter system which would be truly portable between different machines. This was achieved by basing the implementation on macros; by using these macros the system has actually been transferred to machines differing as widely as the CDC 6600 and the Atlas 2; this is surely an achievement which has been matched by few other software systems.

The present book is not a detailed manual for SNOBOL 4 implementors, it is an account of the development of SNOBOL 4, with a description of the fundamental problems, as the authors saw them, followed by a discussion of the philosophical basis for their solution. It is thus a glimpse into the minds of the implementors and is therefore much more informative than any manual could be. Because of this, the book is going to become a prime reference for historians of programming, and could well be a standard text in graduate courses on software engineering. Several chapters actually have exercises in them, but these are somewhat restricted to technical details, and most teachers will, in my opinion, want to follow up the general discussion in each chapter.

Part of the popularity of SNOBOL is the result of the original flavour of the language; problems have been thought through in a new way, and very little has been borrowed. This tone of originality comes through in the book, and readers will find themselves alternately delighted and irritated by the views expressed, for example, the remark

> 'Declarations are largely a concession to implementation . . . . Programmers become used to such requirements and may even think of them as useful programming tools'

is likely to provoke a strong reaction from language designers and implementors. To sum up, the book is obviously required reading for the knowledgeable SNOBOL enthusiast, but will be just as valuable to anyone interested in the broader aspects of language and software design, and will be particularly valuable to advanced students.

J. J. FLORENTIN (London)

Elements of BASIC, 1973; 84 pages. (*National Computing Centre*, £1·80)

This is an introductory book on BASIC which the authors claim is suitable for use in schools or as a self study book. To overcome the problems of machine dependence, loose cards are included in a flap at the back to show variations in a number of manufacturers implementations. On the whole I found this book unsatisfactory. Any text for a programming language should ensure that the programs given as examples are exemplary. Bearing in mind that the examples are likely to be fairly simple the particular point which should be emphasised is program legibility. This should manifest itself through good layout, adequate commentary, and in the case of BASIC, careful layout of the data statements and good run-time documentation. Few of the programs exhibit these features and some of the necessary facilities such as blank lines and comments following an apostrophe are not introduced. One of the programs only works by the skin of its teeth but is totally incomprehensible. The authors also have or impart a few misconceptions about BASIC, particularly concerning string arrays and the MAT statements. Concepts such as the current file and when one may omit quotes around strings and jargon such as real-time are introduced with no explanation and the description of some of the algorithms are rather brief. Finally the book is an odd shape, meaning 12″ wide by 8½″ high which makes it difficult to store in a bookshelf.

On the positive side it seems fairly error free with a clear layout. The flowcharts are good and I applaud the use of teletype output for program listings and results. There is a good selection of examples from the simple to the sophisticated. At £1·80 for a hardback it is fairly cheap.

G. M. BULL (Hatfield)