Data structures for a network design system

M. Etherton

The Computer Laboratory, University of Cambridge, Corn Exchange Street, Cambridge

This paper briefly describes RAINBOW—a suite of programs designed to assist in the specification and analysis of network problems (electrical, logical and other). It goes on to describe in some detail the internal storage structures used for representing pictorial and network data and ends by outlining the action of two simple electric network analysis programs implemented by the author.

(Received January 1971)

1. Introduction

RAINBOW is a self-contained, computer-based system designed to assist an engineer to draw, edit, amalgamate, file and perform numerical computations on diagrams of networks, in particular of electric circuits. Network data is modelled internally by two kinds of ring structure, a package is provided to permit the creation, modification and deletion of rings.

The RAINBOW system is thought to possess the advantages of providing a very fast interaction at the graphics console and a flexible form of data representation particularly well suited to a wide range of applications programs. A more comprehensive description of the entire system is given in Cheyney et al. 1971.

2. PIXIE—a network drawing program

The reading and editing program operates in a PDP7 computer controlling a Model 33 Teletype and a 340 cathode ray tube display unit with a light pen. The program, PIXIE, is designed primarily for drawing electric networks and is described in detail elsewhere (Wiseman, Lemke, and Hiles, 1969); briefly, it operates as follows:

By simple manipulations of the light pen, drawings of electric circuits may be produced, comprising circuit elements connected by straight line segments. The connecting wires (nodes) are produced by sweeping the pen across the screen, the circuit elements (branches) by pointing at conveniently placed light buttons. The representation of the drawing in the PDP7 store is, in the first instance, a sequence of display commands for the 340 display controller. The designer may indicate when a section of the drawing is considered to be finished — whereupon the display file generated for the section is processed to produce an equivalent representation of the drawing in the form of a ring structure (see section 5). This stores the geometrical information about the diagram in a form more readily handled in subsequent processing. Such a data structure we may call a PIXIE structure. When the PIXIE structure representation of a drawing has been formed, textual data (component values and names, etc.) can be associated with the elements of the picture by means of commands delivered at the Teletype. Certain desirable editing operations may then be carried out — such as deleting, tracking (moving around) and labelling portions of the drawing. In addition 'subcircuits' may be defined for later inclusion in other, distinct circuit drawings.

The PDP7 computer has a data link to TITAN, a larger installation which has both an elaborate, disc-based filing capability and an on-line system by which it may be manipulated. A command is provided at the PDP7 teletype to switch PIXIE into an 'await-transfer' condition. The data transfer request is completed by the user at a TITAN on-line console, and TITAN may be used to file the PIXIE data structure. Subsequent processing of the diagram may then be carried out in TITAN.

3. PIXIE support programs

PIXIE is limited in two major respects. One is that the small store size of the PDP7 severely limits the size of drawing that may be produced. The other is that a PIXIE data structure is not a very suitable form in which to present information to most conceivable applications programs.

To help overcome these difficulties, a number of support programs have been provided; they all run in TITAN.

- **CONN** reduces a PIXIE data structure to a 'CONN-structure' (see section 5)
- **CONNMAP** prints a catalogue of the network connectivity.
- **CONNS** permits CONN structures to be generated from a textual description of a network, rather than a PIXIE structure.
- **EDIT** permits data held in PIXIE object blocks to be modified independently of PIXIE.
- **REPLACE** replaces user-defined objects by their 'equivalent circuits'.
- **JOINUP** allows two or more CONN-structures to be amalgamated—by joining them at similarly labelled nodes.

\[ \text{CONNMAP, REPLACE and JOINUP} \] operate only on CONN structures.

4. Applications programs

At present the following programs are complete or in an advanced state of development. With the exception of PLOT and GRAPH, they all operate on CONN structures.

- **GRAPH** constructs a PIXIE structure of a graph from tabular information.
- **PLOT** uses a Calcomp plotter to produce a hard-copy drawing from a PIXIE structure, with optional scaling and translation of axes.
- **LADAN** analyses simple electric filters or ladder networks over a range of frequencies.
- **LINNET** performs an A.C. analysis of more general linear electric networks; the networks may include nullators and norators.
- **CANOTRAN** performs a D.C. and transient analysis of electric networks containing non-linear elements.
- **LOGSIM** simulates the operation of a logic circuit.

LADAN and LINNET are described more fully in sections 6 and 7 respectively.

5. RAINBOW data structures

As already suggested, display file is not a convenient representation of a screen drawing either for picture editing or, still less,
for interpretation of the drawing as an electric circuit, logic diagram, pipe network, etc. Accordingly the display file generated by the dynamic drawing action is converted into PIXIE ring structure form. Compilers exist for performing this conversion in both directions (Wiseman, 1968).

In turn, the PIXIE-structure for a drawing carries a large amount of geometrical information concerning the relative dispositions of the nodes and branches of a network which is not relevant to, say, circuit analysis. Further, the logical hierarchy appropriate to components of a drawing, and reflected in the PIXIE structure, is not meaningful when the drawing is to be regarded as a network.

For this reason, before further analysis, PIXIE data structures are all preprocessed by a program, CONN, which generates a new ring structure containing basically information concerning the connectivity of the component nodes and branches. The input to CONN is thus a PIXIE structure, its output we may call a 'CONN-structure'.

These two types of higher level data structure, the PIXIE-structure and the CONN-structure, are both special instances of ring structures.

5.1. Ring structures—a summary

A ring structure is a way of using computer storage to represent objects between which relationships exist (Sutherland, 1963; Evans and Katzenelson, 1967; Gray, 1967; Lang and Gray, 1968; Wiseman and Hiles, 1968). The objects represented may be real 'solid' objects, having their counterpart in the sensible world outside us, or abstract entities—consisting of purely mental abstractions from the physical world.

An object is represented by a block of data, called an object-block, which carries the object's description. Relationships between objects are denoted in a ring structure as in the more familiar 'list' by pointers attached to the object-blocks. The name 'ring structure' stems from the fact that sets of object-blocks are linked, through the pointers, into closed rings—thus permitting any object-block to be reached from any other involved in the same relation. (One interpretation of a ring is as a list whose last member carries a pointer to the beginning of the list.) An object-block may carry any number of ring pointers, and hence may be simultaneously involved in several relationships with distinct sets of other object-blocks. It is convenient to select exactly one object on each ring to be the 'ringstart'—the analogue of the first member of a list. The general layout of an object-block is shown in Fig. 1.

Ring processing systems have been described elsewhere (Gray, 1967; Lang et al., 1968; Wiseman et al., 1968). The Ring Structure Processor used by RAINBOW programs is called RSP (Wiseman et al., 1968) and is a simplified form of ASP (Lang et al., 1968)—without its built-in complexity of form for representing hierarchies.

5.2. The PIXIE structure

PIXIE structures are described in detail in (Wiseman et al., 1969). Briefly, a PIXIE ring structure is particularly concerned with representing the component parts of a certain limited kind of drawing as objects in a ring structure. The relation which it is most concerned to represent is that of 'belonging to'.

Two types of picture component are selected as being of special interest viz. 'Point' and 'Line segment'. However, other entities prove themselves to have value. Within a picture there are often common occurrences of a certain group of picture components. It is desirable to represent this group by a new entity—Subpicture—and a particular occurrence of the subpicture by an 'Instance'.

Accordingly the elements of a PIXIE structure are of these four types viz. 'Point', 'Line segment', 'Subpicture', and 'Instance'.

Fig. 1. Format of RSP object-block.

In particular, each circuit element or group of touching line segments (node) is represented by a Subpicture object-block. In summary, a Subpicture can represent any of the following:

1. A Simple Circuit Element—In this case the Subpicture may carry Point elements which define 'attachment points' or 'terminals' for the component.
2. A Node—Here the Subpicture will carry Line element(s) to define the constituent line segments of the node.
3. A Subcircuit—A collection of Instances of further Subpictures.

It is now perhaps appropriate for us to examine the PIXIE structure for a particular drawing. To fix our ideas we use an electric circuit for illustration.

Consider Fig. 2 which, in terms simply of line segments, has many components. However if without deeper analysis we introduce 'V', 'W', 'X', and 'Y' as Subpictures, the drawing becomes quite manageable; it possesses only 11 components. However PIXIE chooses to represent Figure 2 by the 32 object-blocks shown in Fig. 3a.

In Fig. 3a the lettered squares represent ring structure object-blocks; represents a ringstart and a ring pointer. The horizontal lines carry the pointers attached to the object-block from which the line issues, and the vertical lines indicate the objects connected together by the rings. Thus the blocks on the horizontal lines enable one to count the relationships in which a particular object is involved while the blocks on the vertical lines enable us to identify all the objects connected in a specific relationship.

In PIXIE data structures generally, the additional information about the picture conveyed by the distinction between ringstarts and ringpointers is not always of great value, since other conventions largely determine the possible relationships between the four basic objects. However, RSP is particularly sensitive to the distinction between ringstarts and ringpointers—and hence to increase efficiency this distinction is often used to replicate information which could be obtained, less conveniently, by examining the identities of the objects.

The interpretation of Fig. 3a, now, is as follows:

S is the Subpicture which represents Fig. 2 as conceived in the abstract.
I is the Instance which represents a particular occurrence of Fig. 2, for example, Fig. 2 located at a particular place on the printed page.

11 to 15 are instances of five Subpictures S1 to S5 representing the five nodes of Fig. 2. N1, N2A, N2B, N3, N4, N5A, N5B

![Diagram](https://example.com/diagram.png)
are Lines representing the seven line segments which make up the nodes. SR, SC and SL are the 'prototype' subpictures which represent the standard circuit components "-\(\mathcal{R}\)-", "-\(\mathcal{C}\)-" and "\(\mathcal{L}\)" respectively.

PR1, PR2, PC1, PC2, PL1, PL2 are point elements representing 'attachment points' for the built-in elements (Evans, et al., 1967).

16, I7 are the two distinct Instances of "\(\mathcal{W}\)"; I8 that of "-\(\mathcal{C}\)" and I9 that of "\(\mathcal{L}\)" as they are relatively situated in Fig. 2.

Hence 'reading' our PIXIE structure from the top downwards, we interpret it to mean:

'Here is an Instance of a Subpicture comprising 11 components, namely seven occurrences of simple line segments and four occurrences of three simple Subpictures, namely a resistor, "\(\mathcal{W}\)" (twice), a capacitor, "-\(\mathcal{C}\)", and an inductor "\(\mathcal{L}\)."

Apart from their exact relative geometrical dispositions, this precisely describes the content of Figure 2. In PIXIE we carry the relative geometrical disposition of the components as additional data, within the Instance object-blocks I and II to I9.

Notice that circuit components (resistors, capacitors, etc.) are produced by PIXIE to a standard specification, so that only a single prototype (SR, SC, SL) of each type of circuit element need be represented in the PIXIE structure, although several Instances of each may occur. However, nodes or even simple line segments, since they are drawn with greater flexibility, may be of arbitrary length and orientation. Hence it is not practicable to have a single prototype Subpicture for all nodes or line segments, and each has its own distinct Instance and Subpicture.

We may note, in passing, the following 'Theorem':

'The Subpicture for an Instance is always at the ringstarts of their common ring.'

and its corollaries:

1. 'Ringpointers (as opposed to ringstarts) in a Subpicture never relate the Subpicture to an Instance of itself.'
2. 'Subpicture ringpointers which relate it to Instances are the route to the components of the Subpicture.'

We call such a Subpicture with more than one component, a 'complex' Subpicture. Thus in Fig. 3a, S, S2, S5 are complex Subpictures, but S1, S3, S4, SC, SR, SL are not.

5.3. The CONN-structure

A connectivity ring structure, or CONN structure, is designed to show the connectivity of components within a diagram represented by a PIXIE structure. To this end, it distinguishes 'components' i.e. simple subpictures which are not lines (the resistors and capacitor of Fig. 2) from groups of touching lines, which are regarded as 'connectors'. Using the terminology of Graph theory we refer to the components as branches and the connectors as nodes.

The object-blocks within a CONN-structure represent these two kinds of entity together with one other—the subnetwork.
A subnetwork is any group of branches and nodes which it is convenient to regard as a single object.

Within a PIXIE structure, any complex Subpicture which is not a node defines a subnetwork. Thus, in Fig. 3a, S is a subnetwork whose components are represented by the dependent substructure (N1, N2, etc.). We see, therefore, that the entire network is itself a special case of a subnetwork. ‘Proper’ subnetworks may be created by the user at lower levels by ‘grouping’—a light button is provided to help effect this operation.

Fig. 3b.CONN structure for Fig. 2

Two types of ring are present in a CONN-structure; they represent the relations:
(a) ‘Connected to’—of branches to a common node
(b) ‘Belonging to’—of branches, nodes and subnetworks to a subnetwork.

The way in which this is done is best illustrated by first describing the one-level CONN-structure, wherein no proper subnetworks exist. We may then describe a simple modification which permits multi-level or hierarchical (i.e. with proper subnetworks) CONN-structures to be represented.

5.3.1. The one-level CONN-structure
To illustrate this we may consider Fig. 2. It will first be noticed that, since a group of touching line segments forms but a single node, there are fewer nodes than line segments in the picture. Thus we use N2, N5 to indicate the nodes comprising the line segments N2A, N2B, and N5A, N5B respectively.

The one-level CONN-structure for Fig. 2 has one object-block to represent the entire circuit; we call it the peg P. Attached to this there is an object-block for each node and an object-block for each circuit element (see Fig. 3b).

Two rings start on the peg; one joins all the nodes, the other all the branches. Thus P serves as a convenient entry point into a CONN-structure—in the same way as does I for a PIXIE structure such as Fig. 3a.

By examining Fig. 3b, it is not difficult to guess the conventions adopted in forming the CONN-structure.

Let us forget for the moment the N-ring, on which the nodes are all connected, and the B-ring, which connects all the branches. Then we see that each node element contains exactly one ringstart and each branch element as many ringpointers as it possesses terminals to which nodes may be joined (i.e. two for each element in Fig. 2). The fact, for example, that R1, R2 and R3 are all connected to node N2 is expressed by having N2 as the ringstart of a ring linking the appropriate terminals of R1, R2, R3 together.

This form for a CONN-structure is not the only one conceivable but is a very simple way of representing the topology of a circuit, logic diagram or any drawing whose components fall into one of the two classes ‘node’ or ‘branch’. It has proved a very convenient structure from which to extract information for the purpose of circuit analysis.

It might be noted that the presence of ‘node’ object-blocks is not relevant to the aim of merely representing connectivity. For this it is sufficient to have only object-blocks corresponding to branches and show their connectedness by putting them on a common ring. However the presence of node blocks serves the dual purpose of easing the implementation of applications programs and providing a place to store data associated specifically with nodes (e.g. applied voltages and currents).

5.3.2. The multi-level CONN-structure
The inclusion of proper subnetworks into CONN-structures may be effected by a simple extension of the one-level scheme. In this the peg, P, is taken to represent a subnetwork and lies on, not two, but five rings (see Fig. 4).

There are the node (N) and branch (B) rings already described; they hold the component nodes and branches of the subnetwork represented by P. In addition to these there are three more rings, two of which, the A-ring and the S-ring, start on P.

The A-ring is a ring of ‘attachment nodes’—the CONN analogue of PIXIE attachment points. The attachment nodes are a subset of the N-ring nodes and constitute the permissible points of attachment of subnetworks one to another in the building of more complex networks.

The S-ring is a ring of lower level peg elements i.e. of component subnetworks of the subnetwork P.

The S’-ring is a ring of subnetworks at the same level as P. The ring-start of the S’ ring is however a peg element, P, at one level higher than P; the S-ring for P’ coincides with the S’-ring for P. The S’-ring of the top level subnetwork and the S-rings of lowest level subnetworks are, of course, all null.

Fig. 5 illustrates a possible hierarchy of subnetworks in a multi-level CONN-structure; the component nodes and branches of each subnetwork are not shown.

5.4. Generating a CONN-structure
The production of a CONN-structure from a PIXIE structure is effected by the program CONN. In outline, CONN performs the following operation:

A filed copy of a PIXIE ring structure is first read into a list space of maximum size determined by the user; all subsequent data management is performed within this area by the routines of the Ring Structure Processor.

The PIXIE structure is scanned once for Instances of Subpictures of the three types described in section 5.2 viz. subcircuit, circuit element, and node. For each such Instance, a
new object-block is formed and attached for future reference.

to the said Instance; we may call this block the shadow s(l) of

the Instance.

The process is completed in two more scans. In the first the coordinates of all the line segments are extracted and set in a table; together with a pointer to the Instance of the node to which the line segment belongs. In the last scan the branch attachment points are extracted in turn and each is compared with the line segment table for proximity to one of the lines therein; this proximity test consists in seeing if the terminal is within a fixed distance, $\epsilon$, of any point of the line. On a 'proximity match' being determined between branch B, say, and line L, the branch shadow, s(B), is inserted into a ring based on the node shadow s(N(L)) for the node N(L) of which line L is a constituent. In this way the connectivity rings of branches and nodes are built up along the shadow elements.

Finally the CONN-structure so formed is 'cast loose' from the parent PIXIE structure, which is discarded to the mercies of the garbage collector.

It should be added that print names and data are transferred from PIXIE Instances to their shadow CONN elements, as also are generic names from the PIXIE Subpictures (see section 5.5).

5.5. Generic types and replacement

Each built-in PIXIE symbol is assigned to one of 16 'generic classes' (Evans, et al., 1967); this being simply the type of component that the symbol represents. The built-in classes are thus

'RESISTOR', 'CAPACITOR', 'NULLATOR', 'SWITCH' etc.

In addition there is a mechanism for creating new generic classes and assigning to them both user defined objects and user-defined groups (subnetworks). It is also possible to reassign the built-in objects to such a new generic class.

Within a CONN-structure the generic class to which a branch object belongs is denoted by placing a pointer to the class name in the object description area. Clearly the generic type is necessary in, say, circuit analysis programs, which need to recognize the kind of circuit element represented. It also serves an important function in the operation of Replacement.

REPLACE (C, R1, R2, R3, . . . )

operates on a string of arguments—each the name of a (filed) CONN-structure.

C is a structure containing branches which belong to user-created generic classes.

R1, R2, R3 . . . are structures each comprising one or more subnetworks, belonging to distinct user-defined generic classes.

For each branch in C which belongs to the same generic class as a subnetwork of some Rn, a replacement of the branch in C by the subnetwork in Rn is performed.

5.6. Joining networks together

JOINUP (C1, C2) operates on two CONN-structures C1, C2. It scans C1 for nodes with a user-assigned name. For each such node it scans C2 for a similarly labelled node. Where a match occurs the two structures are blended together at the matching nodes. The treatment of C1 and C2 is not symmetrical since the whole of C1 always appears in the result, while only those subnetworks of C2 which contain node-matches will appear.

6. LADAN—analysis of an electric ladder network

LADAN takes as its input the CONN-structure for a PIXIE diagram of a filter of fairly general design. According to options selected by additional auxiliary input data, it will produce tables and graphs of various circuit parameters for sinusoidal excitation at the input terminals over a range of frequencies—also specified by the auxiliary input data.

The following circuit parameters (Terma, Pettit, 1952) are listed, over the specified range of frequencies, in a table.

- Input impedance (resistance and reactance)
- Insertion Loss (magnitude and phase)

In addition a graph may be plotted or displayed, of Insertion Loss (db down) vs. frequency.

6.1. Circuit syntax and auxiliary data

The ladder networks susceptible to treatment by PIXANAL are not quite as general as the conventional definition of a ladder allows. Roughly, only simple-series connected groups of elements are permitted along the shaft and rungs of the ladder; with the slight extension that simple-parallel groups can appear along the shaft.

In order that LADAN can get its bearings in analysing the topology of the drawing of a ladder we further demand that the three end-nodes be labelled IN, OUT, COMMON respectively (Fig. 7a).

For the purpose of our analysis, the ladder is imagined to be connected to a known voltage source at its input end and a known load at its output end. (Fig. 7b.)

As a consequence two additional resistances are involved in the computation, namely:

- R0, the internal resistance of the input voltage generator
- R1, the equivalent resistance of the load at the output terminals

The two values, together with a range of input signal frequencies of interest, are delivered to LADAN in a short list of auxiliary data. This list must also contain a code number to request one of the three possible tables or graphs in which the results of the analysis can appear.

6.2. Action of the program

The analysis proceeds by a cascade process; indeed, the circuit syntax is such that our ladder is a sequence or 'cascade' of shaft or rung segments.

The PIXIE structure of our drawing is first reduced to a CONN-structure. LADAN starts at IN then, in tracing down the ladder to OUT, identifies a segment at a time as being of type shaft or rung, computes its equivalent resistance, capacitance and inductance, constructs a 'stage' matrix and uses this to update the impedance matrix of the network to the current stage. Of course, when the trace arrives at OUT the impedance matrix for the current stage is just that for the whole network. The circuit parameters for the network listed above, are then calculated and the results are printed according to the option selected in the auxiliary input data. An outline of the flow of control in LADAN is given in Fig. 6.
7. LINNET—analysis of an electric network with linear active elements
In this program the techniques for a normal passive analysis are used. Active elements (transistors etc.) must be modelled in the circuit diagram by combinations of nullators and norators in equal numbers. The theory for passive networks is then extended to cover this case (Davies, 1966).

The program operates in two stages. The first, RCL, is a
preprocessing stage which takes as input a CONN-structure of a network containing, optionally, nullators and norators in equal quantities. From this is constructed a set of R-C-L matrices (effectively, a frequency-free form of the network admittance matrix) together with a cross reference table for relating matrix entries to the node names in the original PIXIE and CONN-structures. This data may be filed or used immediately as input to the second stage, LINNET proper.

LINNET takes the R-C-L matrices and node dictionary compiled by RCL, together with a vector of currents injected at specified nodes and a frequency and/or damping factor. From this it computes and prints a list of the complex voltages induced on all nodes of the network.

A more detailed description of the operation of the program is given in the flow diagram of Figs. 8 and 9.

Acknowledgement

The author gratefully acknowledges the assistance of members of the staff of the Cambridge University Computer Laboratory generally. Especially Dr. Neil Wiseman for outlining the design.

![Flow diagram for RCL](https://academic.oup.com/comjnl/article-abstract/14/4/366/325120)
Fig. 9. Flow diagram for LINNET

Volume 14 Number 4
system, his general supervision of the project and assistance in preparing this paper. Also Heinz Lemke for implementing PIXIE on the PDP-7, Chris Cheney for implementing the TITAN RSP package and various run time and debugging aids, and Graham Shipley for providing the first draft of the program LINNET.

References


Book reviews


The text of this book is in German and, as the title suggests, the authors essentially concern themselves with problems in linear algebra whose solution involves a symmetric matrix. The book contains five chapters. The first is introductory and treats topics such as vector and matrix norms, positive-definite quadratic forms, condition number and the Cholesky decomposition. The second deals with iterative methods for the solution of equations with a positive-definite matrix of coefficients and covers the theory of gradient methods and the S.O.R. method, with determination of the optimum acceleration parameters for the special case of block tridiagonal matrices with diagonal blocks. The third chapter discusses the formulation and solution of least squares problems whilst chapter four treats the eigenvalue problem for symmetric matrices. This latter chapter is easily the longest in the book and provides a straightforward account of the methods of Jacobi, Givens and Householder followed by a description of the LR transformation. The special form this transformation takes when applied to tridiagonal matrices leads to a discussion of the QD algorithm and some of its applications. The chapter concludes with an account of the power method and simple treatments of the $A - AB$ eigenvalue problem, where $A$ and $B$ are symmetric with $B$ positive definite. The last chapter is concerned with the discretisation of elliptic boundary-value problems and the solution of the resulting algebraic equations. The reader is here introduced to Young's Property A in connection with the S.O.R. method and to consideration of block relaxation and A.D.I. schemes.

The book presents basic material in a clear manner. The methods described are well illustrated by simple numerical examples and ALGOL procedures are included at selected points in the text to provide concrete realisations of some of the algorithms discussed. The authors have obviously simplified their task by restricting their attention to symmetric matrices, yet within the chosen terms of reference they have produced a very useful and readable book which does not 'dress up' its material any more than is necessary for a sound understanding of the theory and methods presented.

E. L. Albasiny (Teddington)

Computational Methods in Partial Differential Equations, by A. R. Mitchell, 1969; 255 pages. (John Wiley & Sons Ltd., £4.00 (cloth), £2.25 (paper))

The preface to this book, although not the title, makes it clear that the computational procedures to be considered arise entirely from the use of finite-difference methods. Thus, for example, in dealing with hyperbolic equations no account is given of methods based on the use of characteristics, although it is stressed that the reader should become acquainted with their role. The text is aimed at second and third year science and engineering undergraduates and contains a large number of worked examples in addition to exercises for the student.

The book attempts to cover a considerable amount of ground. Thus topics considered are (i) parabolic equations in up to three space dimensions, (ii) two-dimensional elliptic equations together with Laplace's equation in three variables, (iii) first order hyperbolic systems in one and two space dimensions and (iv) second order hyperbolic equations in one, two and three space dimensions. An initial chapter on basic linear algebra and an interesting final chapter on applications in fluid mechanics and elasticity complete the book.

Operator-type methods of deriving formulae are favoured and, for two or more space variables, emphasis is placed on the ability to construct implicit schemes which can be 'split' into computationally simpler forms. In particular the book provides a good introduction to the locally one-dimensional methods developed in recent years by Russian numerical analysts for time-dependent problems, although these are by no means the only splitting schemes considered nor are they especially recommended. The book does indeed frequently detail a large number of possible approaches in various contexts. This can be confusing at times since the relative merits of the proposed schemes are not always clear. As the author himself comments, however, the situation is unlikely to be remedied until more experimental evidence becomes available.

The book is clearly written and deserves to do well. The reviewer noticed a few errors in the text. In particular the stability proof on p. 41 is based on an invalid argument and Table 7 appears to list results for time $t = 2.5$ rather than $t = 3$ as stated.

E. L. Albasiny (Teddington)