# THE NUMERICAL SOLUTION OF SPARSE MATRIX EQUATIONS BY FAST METHODS AND ASSOCIATED COMPUTATIONAL TECHNIQUES

by

SAMUEL OKAFOR OKOLIE, B.Sc.,M.Sc.

A Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of Doctor of Philosophy

of the Loughborough University of Technology

November, 1978.

Supervisor: PROFESSOR D.J. EVANS, Ph.D.,D.Sc.
Department of Computer Studies.

# ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my supervisor, Professor D.J. Evans, for giving me the opportunity in the first place; and subsequently, for his advice, guidance, ideas, encouragement and useful push at every stage of this work.

I am also very grateful to all the staff of the Computer Studies Department, most especially to Drs. A. Benson, C. Rick and E. Lipitakis; and to my colleagues, C. Murphy, N. Missirlis and A. Danaee for the useful discussions I had with them at one time or the other.

The financial support of the Government of the Bendel State of Nigeria is hereby gratefully acknowledged.

I wish to express my special indebtedness to my parents, Chief and Mrs. T. Okolie-Okonkwo and my parents-in-law, Mr. and Mrs. J.O.K. Ikade for their constant encouragement; and to my wife, Victoria, for more than words can express.

Finally, I would like to thank Miss J. Briers for her very expert typing of this thesis.

# DECLARATION

I declare that the following thesis is a record of research work carried out by me, and that the thesis is of my own composition. I also certify that neither this thesis nor the original work contained therein has been submitted to this or any other institution for a degree.

S.O. OKOLIE.

# DEDICATION

*To My Family*

# CONTENTS


PAGE

Chapter 1: INTRODUCTION 1

Chapter 2: BASIC MATHEMATICAL CONCEPTS 7

2.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. 7

2.2 Finite Difference Approximations of Partial Differential Equations .. .. .. .. .. .. .. .. 8

2.3 Notations, Concepts and Properties of Matrices and Vectors.. .. .. .. .. .. .. .. .. 21

2.4 Basic Theory of Continued Fractions .. .. .. .. 33

Chapter 3: FAST ALGORITHMIC METHODS FOR THE SOLUTION OF PERIODIC TRIDIAGONAL MATRIX EQUATIONS 40

3.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. 40

3.2 Algorithms for the Solution of the General Periodic Tridiagonal Systems of Equation.. .. 41

3.3 Comparisons and Numerical Results .. .. .. .. .. 63

3.4 The Solution of Constant Term Cyclic Tridiagonal Matrix Systems .. .. .. .. .. .. 66

Chapter 4: SOME FAST ALGORITHMS ASSOCIATED WITH THE SOLUTION OF DIRICHLET'S AND NEUMANN'S BOUNDARY VALUE PROBLEMS 78

4.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. 78

4.2 The Algorithms of Thomas and Conte and Dames For Tridiagonal and Quindiagonal Matrix Systems.. .. .. .. .. .. .. .. .. .. .. .. .. 79

4.3 Fast Algorithms Based on the Reversed Tridiagonal Factorisation and Expansion (ReTriFE) Method.. .. .. .. .. .. .. .. .. .. 88

4.4 Recursive Point Partitioning (R.P.P.) Methods for the Fast Solution of Banded Matrix Equations Associated with Dirichlet's and Neumann's Boundary Conditions .. .. .. .. .. 105

4.5 Parallel Implementation of the Recursive Point Partitioning (R.P.P.) Algorithms .. .. .. .. 120

4.6 Rounding Error Analysis for the Recursive Point Partitioning Method.. .. .. .. .. .. .. 125

4.7 The Recursive Point Decoupling Algorithm.. .. .. 131

PAGE

Chapter 5: THE SOLUTION OF PERIODIC BOUNDARY VALUE PROBLEMS BY FAST METHODS 143

5.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. 143

5.2 The Solution of the Periodic Matrix Difference Equation by the Spectral Resolution Method .. .. .. .. .. .. .. .. 145

5.3 A Block Factorisation Method for the Solution of an Elliptic P.D.E. with Periodic Boundary Conditions in a Rectangular Region .. .. .. .. .. .. .. .. 149

5.4 The Solution of Periodic Parabolic Problems by Boundary Value Techniques.. .. .. .. .. 158

5.5 Application of a Cyclic Lower Block Matrix in the Solution of the Periodic Parabolic Problem .. .. .. .. .. .. .. .. 169

5.6 The Solution of the Fourth Boundary Value Self-adjoint Problem for Parabolic Partial Differential Equations .. .. .. .. 176

5.7 The Solution of Linear Periodic Parabolic P.D.E.'s in Two-Space Dimensions by Fast A.D.I. Methods.. .. .. .. .. .. .. .. 182

Chapter 6: FAST METHODS FOR THE SOLUTION OF DIRICHLET'S AND NEUMANN'S BOUNDARY VALUE PROBLEMS 186

6.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. 186

6.2 Some Simple One-Dimensional Problems of Mathematical Physics .. .. .. .. .. .. .. 187

6.3 Block Rectangular Factorisation Method for the Solution of the Discrete Poisson Equation with Dirichlet's and Neumann's Boundary Conditions in a Rectangular Region 193

6.4 Fast A.D.I. Methods for the Solution of the Linear Parabolic Partial Differential Equations with Dirichlet's and Neumann's Boundary Conditions.. .. .. .. .. .. .. .. 206

6.5 Fast Numerical Solution of the Biharmonic Equation with Mixed Boundary Conditions in a Rectangular Region .. .. .. .. .. .. 211

PAGE

Chapter 7: FURTHER RELATED TOPIC: METHODS FOR DETERMINING THE EIGENVALUES OF PERIODIC TRIDIAGONAL MATRICES 223

7.1 Introduction .. .. .. .. .. .. .. .. .. .. .. .. .. 223

7.2 Periodic Quotient-Difference (P.Q.D.) Method for the Determination of Eigenvalues of Periodic Symmetric Tridiagonal Matrices .. .. 226

7.3 A Reduction of a Periodic Tridiagonal Matrix to a Sparse Cyclic Lower Hessenberg Form by the P-Q Similarity Transformation .. .. .. 239

7.4 A New Approach to Bairstow's Method in Finding the Eigenvalues of an Unsymmetric Periodic Tridiagonal Matrix .. .. .. .. .. .. .. .. .. 244

7.5 The Determination of the Eigenvalues of Symmetric Periodic Tridiagonal Matrices by Newton's Method .. .. .. .. .. .. .. .. .. 262

7.6 Methods for the Determination of Eigenvectors of Sparse Matrices .. .. .. .. .. .. .. .. .. 267

Chapter 8: CONCLUDING REMARKS 274

REFERENCES 276

APPENDIX I 283

APPENDIX II 312

# CHAPTER 1

# INTRODUCTION

1.1

In the ensuing work, we consider several *fast direct* algorithmic methods for the numerical solution of linear matrix equation,

$$A\underline{u} = \underline{b} \tag{1.1.1}$$

where A is a large order ($n \times n$) sparse matrix, $\underline{b}$ is the corresponding right-hand side vector and $\underline{u}$, is the required solution vector.

Equations of the form (1.1.1) are derived, for example, from the finite difference approximation on a finite grid, of well known partial differential equations (p.d.e.) which occur frequently in Mathematical Physics and Engineering applications. For example, if we let R represent a bounded, connected region in the x-y plane with a boundary $\partial R$ and if $\phi(x,y)$ represents a function defined on $\partial R$, then a mathematical formulation of many problems in Physics and Engineering leads to the determination of a function $U(x,y)$ which is continuous in $R \cup \partial R$, twice differentiable in R and satisfies in R, the general second order p.d.e.,

$$a\frac{\partial^2 U}{\partial x^2} + b\frac{\partial^2 U}{\partial x \partial y} + c\frac{\partial^2 U}{\partial y^2} + d\frac{\partial U}{\partial x} + e\frac{\partial U}{\partial y} + fU = g \tag{1.1.2}$$

and on the boundary $\partial R$ a Dirichlet's condition of the form,

$$U(x,y) = \phi(x,y) \ . \tag{1.1.3}$$

The normal derivatives $\frac{\partial U}{\partial n}$, to give the Neumann boundary condition, or a linear combination of U and $\frac{\partial U}{\partial n}$ may also be specified on $\partial R$. The p.d.e. of the form (1.1.2), (usually denoting in most cases, the model form of one of the conservation principles of Physics) represents the rate of change of unknown quantities, (e.g. temperature, pressure, potential, etc.) with respect to two or more independent variables, (e.g. time, displacement or angle).

Equation (1.1.2) is said to be *linear* if the coefficients a,b,c,d, e,f and g are constants (including the value zero) or involve x and y alone; *quasi-linear* if any of the coefficients are functions of the independent variables x and y (e.g., a=a(x,y)) or involve $\partial U/\partial x$ and $\partial U/\partial y$ alone; and *non-linear* if any coefficients are functions of

the dependent variable U. Further, equation (1.1.2) is said to be *self-adjoint* if it can be written in the form,

$$\frac{\partial}{\partial x}(a(x)\frac{\partial U}{\partial x}) + \frac{\partial}{\partial y}(c(y)\frac{\partial U}{\partial y}) + fU = g \ ; \qquad (1.1.4)$$

and a necessary and sufficient condition for (1.1.2) to be of the form (1.1.4) is that

$$d = \frac{\partial}{\partial x}a(x) \quad \text{and} \quad e = \frac{\partial}{\partial y}c(y) \ .$$

Equations of the form (1.1.2) are also conventionally classified with respect to the sign of the quantity of discriminant $\Delta=b^2-4ac$. Specifically, (1.1.2) is said to be

(a) elliptic if $b-4ac<0$,

(b) parabolic if $b^2-4ac=0$,

and (c) hyperbolic if $b^2-4ac>0$,

for all x,y,U in the domain under consideration.

Typical and respective examples of the above class of p.d.e.'s are:

(a) Poisson equation, Laplace equation, $\left.\begin{matrix}\\ \\ \end{matrix}\right\} \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \left\{\begin{matrix} g(x,y), \\ 0 \end{matrix}\right.$ , (1.1.5)

(b) The heat conduction equation $\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2}$ , (1.1.6)

and (c) The wave equation, $\frac{\partial^2 U}{\partial t^2} = \frac{\partial^2 U}{\partial x^2}$ . (1.1.7)

Equation (1.1.5) is generally associated with equilibrium or steady state problems. For example, the velocity potential for the steady flow of an incompressible non-viscous fluid satisfies Laplace's equation and it is the mathematical model of the conservation principle; i.e., the rate at which such a fluid enters any given region is equal to the rate at which it leaves it. Equations involving the time(t) as one independent variable generally lead to parabolic or hyperbolic problems. Such problems usually result from oscillatory, diffusion or equilibrum processes. The simplest example of a parabolic equation, given by (1.1.6) governs the flow of heat in a bar or a rod. The majority of hyperbolic equations, on the other hand, arise from vibration problems, or those in

which discontinuities, such as shock waves, persist in time.

In the past, direct methods, e.g. Gaussian elimination, had been considered unsatisfactory, mainly on the grounds of storage, for the solution of the large order, sparse matrix equations (1.1.1) derived from the finite difference discretisation of commonly occurring p.d.e.'s (Forsythe and Wasow (1960)). Theoretical investigations have therefore been primarily directed towards the development of efficient iterative methods; some of the best known schemes are the Jacobi, Gauss-Seidel, Successive Overrelaxation (S.O.R.) methods and many variants of these. These methods, while they often require little computer storage, nevertheless, have a common feature by implication; i.e., the accuracy of the solution is determined by the number of iterative steps; and for most problems it may be too expensive on computer time to perform as many iterations as would give the desired accuracy.

Some of the major advances in the development of iterative schemes are due, amongst others, to the efforts of Stein and Rosenberg (1948), Frankel (1950), Young (1954) and Varga (1962). We shall however not be concerned in this thesis, to any great extent, with iterative methods for the solution of (1.1.1).

On the other hand, in recent years, renewed interest has been shown in the development of several direct computational techniques for the fast and accurate solution of the finite difference equation of the form (1.1.1). By taking advantage of the special block structure of the approximating discrete equation on a uniform rectangular mesh, fast methods have been proposed which can obtain the solution of the discrete equation with striking efficiency and accuracy. Some of the most successful developments in this direction, particularly in connection with the solution of Poisson equation in a rectangular region, include the use of *fast Fourier transforms* (Hockney (1965,1970)) which relies on the knowledge of a certain set of

trigonometric eigenvectors; the *cyclic reduction* method (Hockney (1965)) which takes advantage of the regular block tridiagonal structure of the coefficient matrix; a stable modification of the latter by Bunemann (1969); its extensions to irregular regions by Buzbee, Dorr, George and Golub (1971) and generalisations of the cyclic reduction strategy by Sweet (1974) and Swarztrauber (1974). A few other methods amonst others include the *spectral resolution* method (Buzbee, Golub and Nielson (1970)) which is a very efficient implementation of the idea of separation of variables, the *sparse factorisation* method (Evans (1971A)) and the marching technique employed by Bank (1976). A survey of some of these widely used methods is given in Dorr (1970).

Direct methods are attractive because in theory they yield the exact solution of the difference equation whereas commonly used iterative methods seek to approximate the solution by iterative procedure until convergence takes place. Secondly, most direct methods permit the solution of systems, $A\underline{u}_i = \underline{d}_i$, with several different right hand sides without having to repeat all the computation for each new right hand side, since the operation (e.g. factorisation) performed on the coefficient matrix need be done only once. Furthermore, in many physical situations accuracy and speed are crucial, as would be required in the modelling of physical systems; e.g., in Plasma studies, the calculation of the electrostatic potential distribution from a given charge distribution; in meteorology, the stream function from the vorticity; and in gravitation, the gravitational potential from a mass distribution. In these problems, the potential function calculation (usually requiring the solution of Poisson equation) may be only a necessary step towards the calculation of some other functions such as the self magnetic field; and it is often the case that the solution of the Poisson equation is time consuming and forms the bottle-neck in the modelling of these physical systems on a computer. In

such cases, the use of fast direct methods in the determination of the potential distribution is preferred to iterative schemes from the point of view of speed and accuracy.

The presentation in this thesis is as follows:

Chapter 2 contains introductory material in which we present basic mathematical concepts required in the later parts of this work.

In Chapter 3, we introduce various new fast algorithms and their variants for the solution of the one dimensional periodic tridiagonal matrix equation; the methods used being based essentially on different strategies, including Gaussian elimination, generalised sparse cyclic factorisation, cyclic reduction and rank-one perturbation concepts. This is done in order to provide background material for the work in Chapter 5. Comparisons of these algorithms based on numerical experiments are given. Further new fast algorithms and their extensions, based on the reversed triangular (or more appropriately rectangular) factorisation and expansion (ReTriFE), recursive point partitioning (R.P.P.) and recursive decoupling ideas for the solution of tridiagonal and/or other sparse banded matrix systems, are presented in Chapter 4. These algorithms can generally be applied in the solution of Dirichlet's and Neumann's boundary-value problems and thus provide further information to the solution of such problems considered in Chapter 6.

A number of model problems, including parabolic and elliptic p.d.e.'s defined over rectangular regions and with periodic boundary conditions are considered in Chapter 5. Specific problems considered include the Helmholtz equation, the one and two space heat conduction equations and a fourth boundary value self-adjoint parabolic problem. For the finite difference method of solution of these p.d.e.'s, we introduce direct block methods and the spectral resolution schemes for their solution. The block methods are based on the generalisation of the point form algorithms in Chapter 3 while the latter technique also makes extensive use of the fast periodic tridiagonal

matrix solver algorithms.

Further, discretised matrix equations derived from the transport, Poisson, heat-conduction and Biharmonic equations in one or two space dimensions under Dirichlet's and Neumann's boundary conditions in a rectangular region are considered in Chapter 6 for which block fast direct methods based on, or incorporating the ReTriFE and R.P.P. algorithmic ideas are developed for their fast solution.

The eigenvalue problems, considered as an extension of the equilibrium problem, in which critical (eigen-) values of certain parameters are required in addition to the steady-state configuration, is considered in Chapter 7. The Sturm Liouville problem with periodic boundary conditions is used as a model problem. By applying a sparse cyclic matrix factorisation involving a continued fraction expansion, to the periodic tridiagonal matrix of discretised equations, new similarity transformations (which are sparse form extensions of the Rutishauser's QD and LR schemes) are proposed and used in the determination of the eigenvalues of periodic tridiagonal matrices under certain special conditions.

Finally, the thesis concludes with a summary and recommendations for extension and further work.

# CHAPTER 2

# BASIC MATHEMATICAL CONCEPTS

## 2.1 INTRODUCTION

In this chapter we shall discuss some of the basic concepts of Numerical Analysis that will be made use of in subsequent chapters. The background material to be outlined include the following:

(a) finite difference schemes in common use for the discrete approximation of partial differential equations which occur in problems of Mathematical Physics;

(b) the notation, concepts and properties of matrices and vectors; and

(c) the basic theory of continued fractions.

In most cases, especially in sections (2.3) and (2.4), basic theorems will be introduced without proof. In such circumstances, the theorems have been given elsewhere in references which are indicated as appropriate.

## 2.2 FINITE DIFFERENCE APPROXIMATIONS OF PARTIAL DIFFERENTIAL EQUATIONS

In the solution of commonly occurring problems of Mathematical Physics it is usual to apply discrete methods of solution. This involves the approximation of the governing partial differential equations and the associated boundary conditions of the continuous system by finite discrete methods, which can be realised either by a physical or mathematical approach. In some limited problems, such as in a heat conduction problem, (for example), a heat-conducting slab could be replaced by a network of heat conducting rods. Such a physical approach can be useful in obtaining a solution if the discrete physical model is given the lumped physical characteristics of the continuous system. We shall, however, not pursue this approach further. On the other hand, when the continuous system can be represented by a mathematical formulation, usually a partial differential equation (p.d.e.), the mathematical method of solution is simpler and more flexible. In such a case, derivatives are usually replaced by finite difference approximations. Some other methods of discretisation include finite element methods, variational methods and methods of lines but in what follows, our interest would be based entirely on the finite difference representation. The latter has, not only inherent structural simplicity, but also wide applicability.

The basic aim of the finite difference approximation methods is the reduction of continuous systems to discrete systems which are more suitable for high-speed computer solution. The approximation involves the replacement of a continuous closed domain by a network or grid of discrete points within the closed domain R so that instead of developing a solution defined everywhere in R, we obtain approximations to the continuous solution at the isolated grid points. Other intermediate values may be obtained from the available discrete solution by additional interpolating techniques.

We consider, for example, and without loss of generality, an elliptic p.d.e., e.g. the two dimensional Poisson equation,

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f(x,y) \qquad (2.2.1)$$

where x and y are the independent variables. The region of the problem solution is a connected region R in the x-y plane, subject to given boundary conditions on the boundary $\partial R$ of R (e.g. U(x,y)=0 may be specified on the boundary $\partial R$).

If we let $\bar{R}=R\cup\partial R$ denote the closure of the region R with the boundary $\partial R$, then we can superimpose on $\bar{R}$ a system of rectangular meshes formed by two sets of equally spaced lines; one set parallel to the x-axis, and the other to the y-axis. An approximate solution to the differential equation can now be sought at the points of intersection of these parallel lines. Such points may be denoted as $P_{1,1}, P_{1,2}, \ldots, P_{i,j} \ldots$ as shown in Figure (2.1); and are called mesh (lattice, grid, nodal or pivotal) points. The distance h between two parallel lines is the mesh size assumed here to be uniform in both directions.

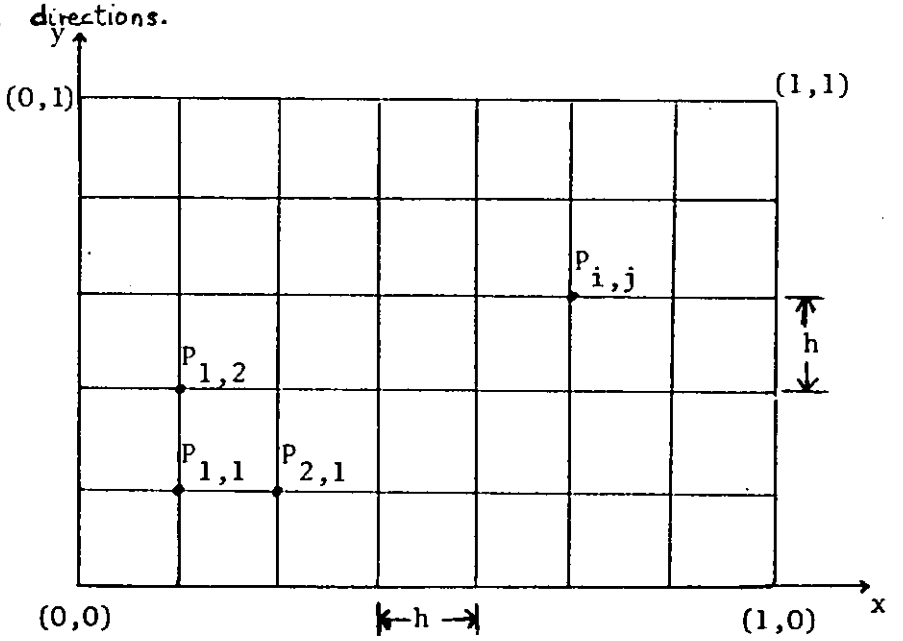



FIGURE 2.1

If there are N mesh points internal to $\bar{R}$, then the discrete numerical solution sought is obtained by approximating the given partial differential equation over the region R by N algebraic equations involving the approximate values of U at the N mesh points internal to $\partial R$. This approximation consists of replacing each derivative of the p.d.e. at the point $P_{i,j}$ (say) by a

finite difference approximation in terms of values of U at $P_{i,j}$, and at the neighbouring mesh and/or boundary points. When this is repeated for each of the internal nodal points the approximating algebraic difference equation is obtained. Simple approximation schemes for the derivatives obtained by the use of Taylor's series expansions will now be discussed.

Taylor's Series Expansion

We shall assume that U(x,y) is sufficiently differentiable in a sufficiently large neighbourhood of the point (x,y) and that the mesh sizes, h, are uniform in the chosen region. Then, by Taylor's theorem we have, at the neighbouring point (x+h,y),

$$U(x+h,y) = U(x,y)+h\frac{\partial U}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 U}{\partial x^2} + \frac{h^3}{3!}\frac{\partial^3 U}{\partial x^3} + \frac{h^4}{4!}\frac{\partial^4 U}{\partial x^4} + \ldots . \qquad (2.2.2)$$

Similarly, at the other neighbouring points we have,

$$U(x-h,y) = U(x,y)-h\frac{\partial U}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 U}{\partial x^2} - \frac{h^3}{3!}\frac{\partial^3 U}{\partial x^3} + \frac{h^4}{4!}\frac{\partial^4 U}{\partial x^4} - \ldots , \qquad (2.2.3)$$

$$U(x,y+h) = U(x,y)+h\frac{\partial U}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 U}{\partial y^2} + \frac{h^3}{3!}\frac{\partial^3 U}{\partial y^3} + \frac{h^4}{4!}\frac{\partial^4 U}{\partial y^4} + \ldots , \qquad (2.2.4)$$

and

$$U(x,y-h) = U(x,y)-h\frac{\partial U}{\partial y} + \frac{h^2}{2!}\frac{\partial^2 U}{\partial y^2} - \frac{h^3}{3!}\frac{\partial^3 U}{\partial y^3} + \frac{h^4}{4!}\frac{\partial^4 U}{\partial y^4} - \ldots , \qquad (2.2.5)$$

where the point (x,y) and its four neighbouring mesh points (x±h,y) and (x,y±h) are contained in $\bar{R}$.

Combinations of the formulae in (2.2.2) and (2.2.3) yield the following approximations,

$$\frac{\partial U}{\partial x} = \frac{U(x+h,y)-U(x-h,y)}{2h} + O(h^2) \quad , \qquad (2.2.6)$$

and

$$\frac{\partial^2 U}{\partial x^2} = \frac{U(x+h,y)-2U(x,y)+U(x-h,y)}{h^2} + O(h^2) \quad . \qquad (2.2.7)$$

Similarly, combinations of (2.2.4) and (2.2.5) give,

$$\frac{\partial U}{\partial y} = \frac{U(x,y+h)-U(x,y-h)}{2h} + O(h^2) \quad , \qquad (2.2.8)$$

and

$$\frac{\partial^2 U}{\partial y^2} = \frac{U(x,y+h)-2U(x,y)+U(x,y-h)}{h^2} + O(h^2) \quad , \qquad (2.2.9)$$

where these finite difference schemes are second order approximations and the quantities $O(h^2)$ denote the asymptotic notation for the truncation error in these approximations.

Further, if we denote a general lattice point (x,y) by (ih,jh), and U(x,y) by $U_{i,j}$, then, using the approximations for $\frac{\partial^2 U}{\partial x^2}$ and $\frac{\partial^2 U}{\partial y^2}$ in (2.2.7) and (2.2.9) respectively the Poisson equation (2.2.1) can be replaced at the point $(x_i,y_i)=(ih,jh)$ by the linear algebraic difference equation,

$$-U_{i-1,j}-U_{i,j-1}+4U_{i,j}-U_{i+1,j}-U_{i,j+1} = -h^2 f_{i,j} - \frac{h^4}{12}\left\{\frac{\partial^4 U}{\partial x^4} + \frac{\partial^4 U}{\partial y^4}\right\}_{i,j} - \ldots . \qquad (2.2.10)$$

The terms on the right hand side of (2.2.10) excluding $\{-h^2 f_{i,j}\}$, are defined as the local truncation error of formula (2.2.10) and the asymptotic notation $O(h^4)$ is the principal part of this error.

The left-hand side of (2.2.10) is often represented diagramatically in the form of a computational molecule or stencil as illustrated in Figure (2.2).

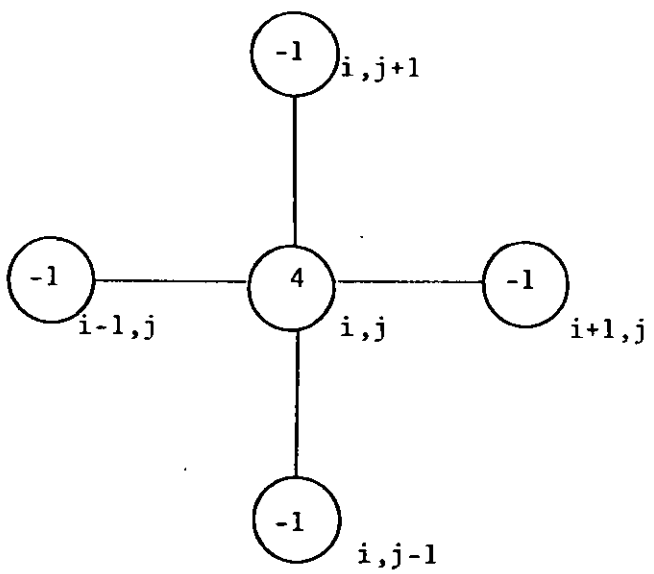



FIGURE 2.2

If the truncation error is neglected and the computational molecule in Figure (2.2) is applied to all the mesh points, we are then able to construct a set of linear equations in terms of the unknown functions $u_{i,j}$ (which denote the finite difference approximation of the exact solution $U_{i,j}$ at the point (ih,jh)). In matrix notation, we denote this

linear system of equations as

$$A\underline{u} = \underline{b} \qquad (2.2.11)$$

where $\underline{b}$ is a vector whose components are made up of known values $\{-h^2 f_{i,j}\}$ and the values of $U_{i,j}$ given at the boundary $\partial R$. If the region under consideration is square and has $N(=n^2)$ internal mesh points then the matrix A is a square matrix of order $n^2$ of the form,

$$A = \begin{bmatrix}
4 & -1 & & & -1 & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & \\
 & -1 & 4 & -1 & & & \ddots & & & & & \\
 & & \ddots & \ddots & \ddots & & & -1 & & & & \\
 & & & -1 & 4 & & & & -1 & & & \\
-1 & & & & & 4 & -1 & & & -1 & & \\
 & \ddots & & & & -1 & 4 & -1 & & & \ddots & \\
 & & \ddots & & & & \ddots & \ddots & -1 & & & -1 \\
 & & & -1 & & & & -1 & 4 & & & \\
 & & & & -1 & & & & & 4 & -1 & \\
 & & & & & \ddots & & & & -1 & 4 & -1 \\
 & & & & & & -1 & & & & -1 & 4
\end{bmatrix}$$

i.e., row-wise or column-wise ordering of the mesh points is assumed.

A is also written as an (n×n) block matrix of the form,

$$A = \begin{bmatrix}
B & -I & & & \\
-I & B & -I & & 0 \\
 & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & -I \\
 & 0 & & -I & B
\end{bmatrix}$$

where B is an (n×n) tridiagonal matrix,

$$B = \begin{bmatrix}
4 & -1 & & & \\
-1 & 4 & -1 & & 0 \\
 & \ddots & \ddots & \ddots & \\
 & & \ddots & \ddots & -1 \\
 & 0 & & -1 & 4
\end{bmatrix}$$

and I is the identity matrix.

Both $\underline{u}$ and $\underline{b}$ are $(n^2 \times 1)$ column vectors.

Further, we consider a one-space heat conduction (parabolic, initial-boundary value) problem defined by the equation

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} \qquad (2.2.12)$$

in the semi-infinite region $0 \leq x \leq a$; $t > 0$, subject to the initial condition

$$U(x,0) = f(x); \qquad 0 \leq x \leq a$$

and the boundary conditions

$$U(0,t) = g_0(t),\ U(a,t) = g_a(t); \quad t \geq 0 .$$

We shall assume that the partial derivatives of U are continuous and uniformly bounded in the semi-infinite region.

A rectangular grid can be overlayed over the semi-infinite domain with spacing h in the x-direction and $\ell$ in the t-direction as shown in Figure (2.3).

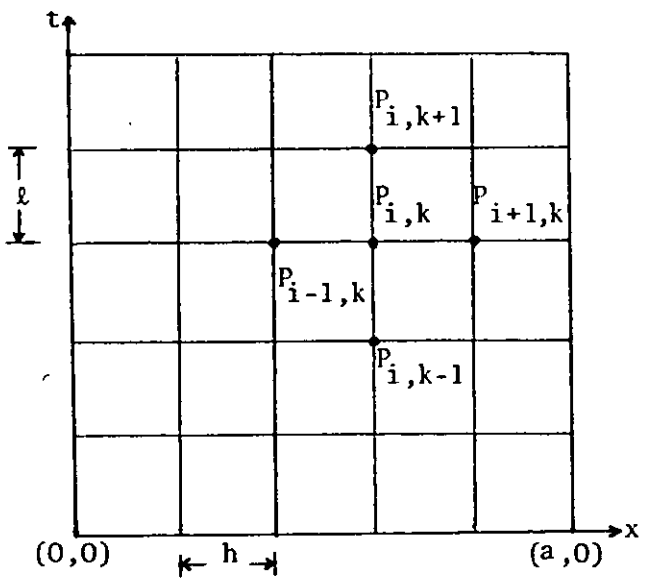



FIGURE 2.3

Let $U_{i,k}$ denote $U(ih,k\ell)$ at the point $(x_i,t_k)=(ih,k\ell)$. If we represent the space derivative $\frac{\partial^2 U}{\partial x^2}$ by the central second difference formula (2.2.7) and the time derivative $\frac{\partial U}{\partial t}$ by the forward difference approximation

$$\frac{\partial U}{\partial t} = \frac{(U_{i,k+1} - U_{i,k})}{\ell} ,$$

then the simplest finite difference replacement of (2.2.12) can be written as,

$$U_{i,k+1} = \sigma U_{i-1,k} + (1-2\sigma)U_{i,k} + \sigma U_{i+1,k} + O(\ell^2+\ell h^2) \qquad (2.2.13)$$

where $\sigma=\ell/h^2$ is the mesh ratio.

It may be noticed that the $U_{i,k+1}$ is obtained solely in terms of values of U at the $(k)^{th}$ time level; thus equation (2.2.13) is an equation for marching the solution ahead in time. Such a forward difference scheme is termed *explicit*; and it has the computational molecule shown in Figure (2.4). Generally, explicit formulae provide for non-iterative marching techniques for obtaining the solution at each point in terms of the known preceding and boundary values; and they are usually applicable to parabolic and hyperbolic equations which characteristically have open integration domains.

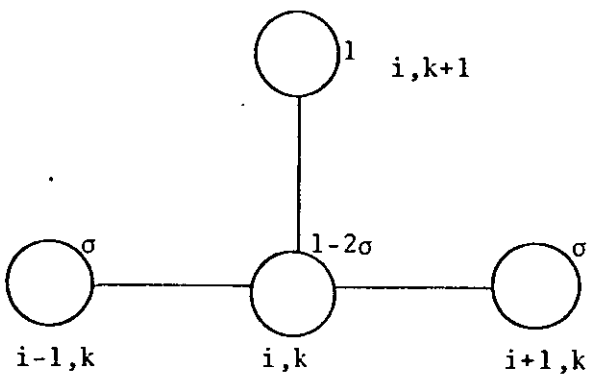



FIGURE 2.4

Neglecting the truncation error term in (2.2.13) and applying the molecule in Figure (2.4) to all the mesh points (in the interval $0<x<a$), in turn, we obtain a set of linear equations, which in matrix form is given by,

$$\underline{u}_{k+1} = A\underline{u}_k, \quad k\geq 0 \qquad (2.2.14a)$$

where A is a tridiagonal matrix of the form,

$$A = \begin{bmatrix} 1-2\sigma & \sigma & & & \\ \sigma & 1-2\sigma & \sigma & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \sigma \\ 0 & & & \sigma & 1-2\sigma \end{bmatrix} \qquad (2.2.14b)$$

and $\underline{U}_k=(u_{1,k},u_{2,k},\ldots,u_{n,k})^T$ denote the finite difference solution corresponding to the exact solution $\underline{U}_k$ at the points $(1,k),(2,k),\ldots,(n,k)$.

Well known stability analysis (see, for example the texts by Smith (1969) and von Rosenberg (1969)) shows that a very restrictive value for $\sigma$ ($\sigma\leq\frac{1}{2}$) must be satisfied in order for the solution of (2.2.14a) to approach that of the p.d.e. (2.2.12). The consequence of this restriction is that the size of the time increment $\ell$ must be very small (of order $h^2$) for the solution to be stable. A finite difference equation which does not have this restriction must therefore be considered.

We next consider the approximation of the derivative $\frac{\partial^2 U}{\partial x^2}$ in the $(k+1)^{th}$ row instead of the $k^{th}$ row as proposed in O'Brien et al (1951), so that we now have the p.d.e. (2.2.12) represented by the implicit scheme

$$-\sigma U_{i-1,k+1} + (1+2\sigma)U_{i,k+1}-\sigma U_{i+1,k+1} = U_{i,k} . \tag{2.2.15}$$

In Crank and Nicolson (1947) an average of the approximations for $\frac{\partial^2 U}{\partial x^2}$ in the $(k)^{th}$ and $(k+1)^{th}$ rows was proposed. The resulting difference scheme becomes,

$$-\sigma U_{i-1,k+1}+2(1+\sigma)U_{i,k+1}-\sigma U_{i+1,k+1}$$
$$= \sigma U_{i-1,k}+2(1-\sigma)U_{i,k}+\sigma U_{i+1,k}+O(\ell^2+h^2) . \tag{2.2.16}$$

A weighting factor $\omega$ can be introduced such that $\frac{\partial^2 U}{\partial x^2}$ is now represented in a more general form as,

$$\frac{\partial^2 U}{\partial x^2} = \frac{\omega}{h^2}(U_{i-1,k+1}-2U_{i,k+1}+U_{i+1,k+1})-$$
$$(\frac{1-\omega}{h^2})(U_{i-1,k}-2U_{i,k}+U_{i+1,k}) \quad \text{for } 0\leq\omega\leq 1. \tag{2.2.17}$$

Then, equation (2.2.12) becomes, on neglecting the local truncation error,

$$-\sigma\omega U_{i-1,k+1}+(1+2\sigma\omega)U_{i,k+1}-\sigma\omega U_{i+1,k+1}$$
$$= \sigma(1-\omega)U_{i-1,k}+\{1-2\sigma(1-\omega)\}U_{i,k}+\sigma(1-\omega)U_{i+1,k} . \tag{2.2.18}$$

The finite difference equations (2.2.15),(2.2.16) and (2.2.18) are such that the unknown values in the (k+1) row are specified in terms of the

known values in the $k^{th}$ row. Such schemes are called *implicit*. The weighted implicit scheme in (2.2.18), for example, has the computational molecule shown in Figure (2.5).

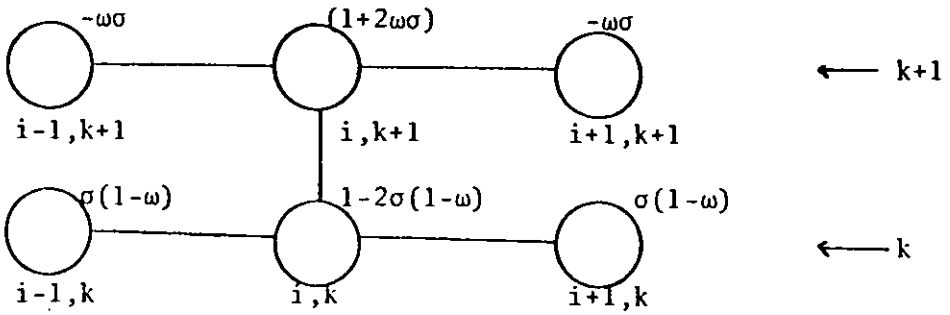



FIGURE 2.5

For ω=0, (2.2.18) gives the explicit scheme (2.2.13),

ω= ½, gives the Crank-Nicolson scheme (2.2.16)

and ω=1, gives the implicit scheme (2.2.15).

The finite difference matrix equation resulting from the Crank-Nicolson implicit scheme is of the form,

$$A\underline{u}_{k+1} = B\underline{u}_k, \qquad k>0 , \tag{2.2.19a}$$

where in the general case (2.2.18), A and B are tridiagonal matrices given by,

$$A = \begin{bmatrix} 1+2\omega\sigma & -\omega\sigma & & & \\ -\omega\sigma & 1+2\omega\sigma & -\omega\sigma & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\omega\sigma \\ 0 & & & -\omega\sigma & 1+2\omega\sigma \end{bmatrix} \tag{2.2.19b}$$

and

$$B = \begin{bmatrix} 1-2\sigma(1-\omega) & \sigma(1-\omega) & & & \\ \sigma(1-\omega) & 1-2\sigma(1-\omega) & \sigma(1-\omega) & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \sigma(1-\omega) \\ 0 & & & \sigma(1-\omega) & 1-2\sigma(1-\omega) \end{bmatrix} \tag{2.2.19c}$$

The implicit scheme (2.2.15) and the Crank-Nicolson scheme (2.2.16) are both stable and convergent (Smith, 1965) for any value of $\sigma > 0$. However, since large values of $\sigma$ would increase the truncation error terms to an unacceptable level, it is recommended that the value of $\sigma$ should be in the range $0 < \sigma < 4$.

The matrix equation (2.2.19) is a step by step (marching) procedure for the numerical integration of the heat conduction equation (2.2.12).

Next, we consider the two space-dimensional parabolic p.d.e. with constant coefficients, i.e.,

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{\partial U}{\partial t} \quad , \tag{2.2.20}$$

in the cylindrical region $R\times[0\leq t\leq T]$ where R is a closed, connected region in the x-y plane with continuous boundary $\partial R$. The initial and boundary conditions given at t=0 and $(\partial R\times 0\leq t\leq T)$ respectively are

$$U(x,y,0) = g(x,y), \quad x,y\in R,$$

and

$$U(x,y,t) = f(x,y,t), \quad x,y\in\partial R, \quad 0\leq t\leq T$$

where g(x,y), f(x,y,t) are prescribed functions of x,y and t.

Let the region in the (x,y,t) space be covered by a rectilinear grid with sides parallel to the axes, and let us assume a square grid of length h in the x-y plane, and a time step of length $\ell$ in the t-direction. The points $(x_i,y_j,t_k)$ are given, as usual, by $x_i=ih$, $y_j=jh$, $t_k=k\ell$; and $U(x_i,y_j,t_k)$ is denoted by $U_{i,j,k}$. Applying the Taylor's series expansion, we can derive finite difference representations of (2.2.20) by using either the explicit, implicit or Crank-Nicolson schemes in the same way as for the one-space dimensional case.

An explicit scheme for equation (2.2.20) is given (Mitchell (1976)) by,

$$U_{i,j,k+1} - U_{i,j,k} = \frac{\ell}{h^2}(\delta_x^2+\delta_y^2)U_{i,j,k} + O(\ell^2+\ell h^2) \tag{2.2.21}$$

where $\delta_x^2,\delta_y^2$ are the central difference operators, i.e.,

$$\delta_x^2 U_{i,j,k} = U_{i-1,j,k} - 2U_{i,j,k} + U_{i+1,j,k},$$

and

$$\delta_y^2 U_{i,j,k} = U_{i,j-1,k} - 2U_{i,j,k} + U_{i,j+1,k}.$$

This gives an explicit discretised form of (2.2.20) as,

$$U_{i,j,k+1} = (1-4\sigma)U_{i,j,k} + \sigma(U_{i-1,j,k}+U_{i-1,j,k}) + \sigma(U_{i,j-1,k}+U_{i,j+1,k}) \quad , \tag{2.2.22}$$

where $\sigma=\ell/h^2$ and $O(\ell^2+\ell h^2)$ truncation error term is neglected.

Similarly, an implicit finite difference scheme for equation (2.2.20) is given, on neglecting the truncation error, by

$$-\sigma U_{i-1,j,k+1}+(1+4\sigma)U_{i,j,k+1}-\sigma U_{i+1,j,k+1}-\sigma U_{i,j-1,k+1}$$

$$-\sigma U_{i,j+1,k+1} = U_{i,j,k} \qquad (2.2.23)$$

and the Crank-Nicolson scheme by,

$$-\sigma U_{i-1,j,k+1}-\sigma U_{i,j,k+1}+2(2\sigma+1)U_{i,j,k+1}-\sigma U_{i+1,j,k+1}-\sigma U_{i,j+1,k+1}$$

$$= \sigma U_{i-1,j,k}+\sigma U_{i,j-1,k}-2(2\sigma-1)U_{i,j,k}+\sigma U_{i+1,j,k}+\sigma U_{i,j+1,k}. \qquad (2.2.24)$$

If we assume that there are $N(=n^2)$ internal grid points in the closed domain R then on applying the implicit difference equation (2.2.23) to all the mesh points, leads to an $(N\times N)$ matrix equation,

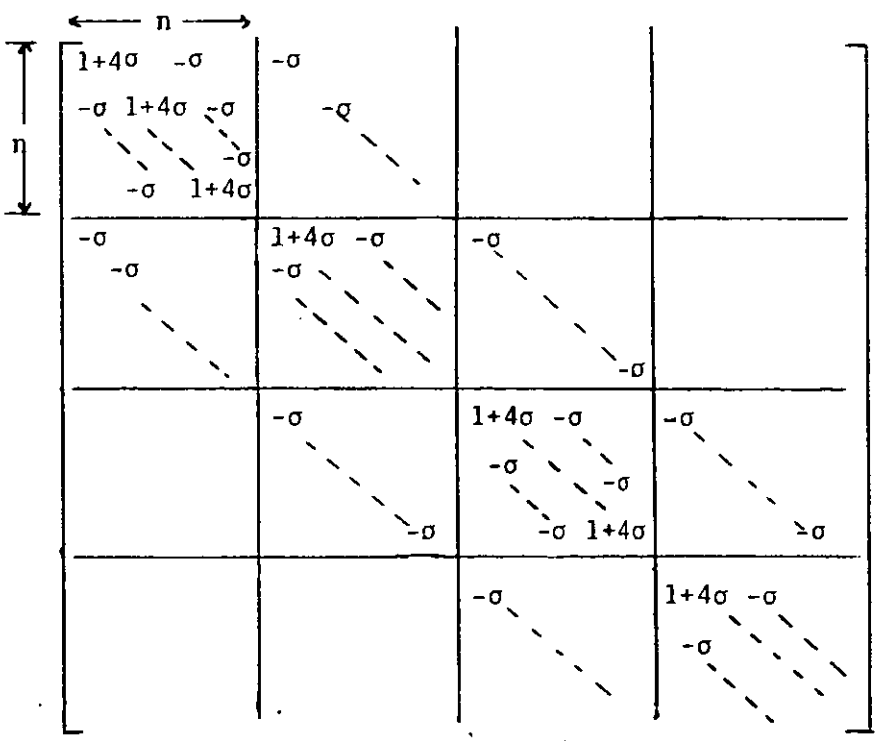
$$A\underline{u}_{k+1} = \underline{u}_k \qquad (2.2.25)$$

where A which may be denoted in the form,

$$A \equiv A[-\sigma;\ -\sigma,\ 1+4\sigma,\ -\sigma;-\sigma] \qquad (2.2.26)$$

is a sparse quindiagonal matrix,

$A[-\sigma;-\sigma,1+4\sigma,-\sigma;-\sigma]=$




(2.2.27)

and $\underline{u}_k = (u_{1,1}, u_{2,1}, \ldots, u_{n,1}, u_{1,2}, \ldots, u_{n,2}, \ldots, u_{1,j}, \ldots, u_{n,j}, \ldots$

$$\ldots u_{1,n}, \ldots, u_{n,n})^T \qquad (2.2.27)$$

denotes the approximate solution at the $k^{th}$ plane; and a similar notation applies to the approximate solution $\underline{u}_{k+1}$ at the $(k+1)^{th}$ plane.

Similarly, the Crank-Nicolson scheme (2.2.24) gives rise to the matrix equation,

$$A\underline{u}_{k+1} = B\underline{u}_k \qquad (2.2.28)$$

where both A and B are sparse quindiagonal matrices of the same form as (2.2.27) and may be denoted, using the notation of (2.2.26) as,

$$A = A[-\sigma;-\sigma,2(2\sigma+1),-\sigma;-\sigma]$$

and

$$B = B[\sigma;\sigma,-2(2\sigma-1),\sigma;\sigma]. \qquad (2.2.29)$$

Another finite difference scheme of frequent application in the numerical solution of problems of Mathematical Physics is the Alternating Direction Implicit (A.D.I.) method proposed by Peaceman and Rachford (1955). It consists of the use of two forms of difference equations alternately in successive time steps; in the first step the finite difference equations are implicit in the x-direction (say) and explicit in the y-direction; and in the second instance, the directions are interchanged. The method may be viewed as a factorised form of the Crank-Nicolson scheme.

Further discussion of the application of the A.D.I. method is made in sections (5.6) and (6.4), where fast algorithmic solutions of parabolic problems of the form (2.2.20) with periodic boundary conditions on the one hand, and Dirichlet's and Neumann's boundary conditions on the other, are discussed.

From the above discussion we have seen that by applying a specific finite difference approximation scheme to a given p.d.e. such as (2.2.1) or (2.2.20), leads to the solution of the matrix equation,

$$A\underline{u} = \underline{b}$$

which has a unique solution $\underline{u}=A^{-1}\underline{b}$ provided that A is non-singular (i.e.,

the determinant of A is non-zero). Quite often, the coefficient matrix A derived from the finite difference approximation of p.d.e.'s is sparse (i.e., many of its elements are zero). The exact structure of A depends on the type of p.d.e. under consideration, the finite difference scheme applied and the ordering of the mesh points. The techniques developed in the solution of the resulting matrix equations depend very much on the structure of the coefficient matrix and other computational considerations such as the need to minimize either the storage requirements or the arithmetic operation counts; or to achieve a high accuracy. We shall, in subsequent chapters, present fast algorithmic methods for the solution of such matrix systems which arise from the finite difference discretisation of the various problems of Mathematical Physics in regular regions.

The matrix A derived as above, is not only square and sparse, but very often has some other special properties such as positive definiteness, diagonal dominance and irreducibility.

In the next section, these and other properties of matrices will be defined and inter-related.

## 2.3 NOTATIONS, CONCEPTS AND PROPERTIES OF MATRICES AND VECTORS

Matrices which will be assumed square (unless otherwise stated) are denoted by capital letters. The determinant of a matrix A (say) will be denoted by det(A) or $|A|$. A is said to be singular if det(A)=0.

The matrix A is non-singular if its inverse ($A^{-1}$) exists and is defined by,

$$AA^{-1} = A^{-1}A = I$$

where I is the identity matrix.

The transpose of a matrix $A=(a_{i,j})$ is denoted by $A^T$ and is the matrix whose element in the $i^{th}$ row, $j^{th}$ column is $a_{j,i}$. If A is complex and the element $a^*_{i,j}$ is the complex conjugate of $a_{i,j}$ then the matrix A* (the conjugate transpose) is the matrix whose element in the $i^{th}$ row and $j^{th}$ column is $a^*_{j,i}$.

A number of special matrices have their elements inter-related in certain ways and may be defined as follows:

| | | |
|---|---|---|
| Hermitian | $A^* = A$ | (2.3.1) |
| Symmetric(real A) | $A^T = A$ | (2.3.2) |
| Unitary | $A^*A = I$ | (2.3.3) |
| Orthogonal(real A) | $A^TA = I$ | (2.3.4) |
| Normal | $A^*A = AA^*$ | (2.3.5) |

In subsequent chapters we shall be dealing with large order sparse matrices. The number of zero elements are large and hence cumbersome to write. Large collections of such zero elements will be represented by a single 0. For example a (6×6) tridiagonal matrix A could be denoted as,

$$A = \begin{bmatrix} c_1 & f_1 & & & & \\ e_2 & c_2 & f_2 & & 0 & \\ & e_3 & c_3 & f_3 & & \\ & & e_4 & c_4 & f_4 & \\ & & & e_5 & c_5 & f_5 \\ & 0 & & & e_6 & c_6 \end{bmatrix}$$

Vectors are denoted by underlined letters if they are column vectors. Elements of the vector will have the same letter, but with a lower subscript which gives the position of that element in the vector. The corresponding row vector will have a superscript T.

Example,

if $$\underline{U} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} ,$$

then $$\underline{U}^T = (u_1, u_2, u_3, u_4).$$

Vector and Matrix Norms

The norm of an n-dimensional vector (matrix) gives a measure of 'size' of the effect of the vector (matrix) as a linear operator.

The norm of a vector $\underline{x}$ (say) is denoted by $||\underline{x}||$ and satisfies the following relations;

$$||\underline{x}|| > 0, \text{ unless } \underline{x} = \underline{0} ,$$

$$||k\underline{x}|| = |k| \, ||\underline{x}|| , \text{ where k is a complex scalar}$$

and $$||\underline{x}+\underline{y}|| < ||\underline{x}|| + ||\underline{y}|| .$$

The general form of a vector norm $||\underline{x}||_p$ is given by

$$||\underline{x}||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p} , \quad p=1,2,\ldots,\infty. \tag{2.3.6}$$

The most useful of the norms defined by (2.3.6) are those for which

$$p=1, \quad ||\underline{x}||_1 = \sum_{i=1}^{n} |x_i| ; \tag{2.3.7}$$

$$p=2, \quad ||\underline{x}||_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} \tag{2.3.8}$$

and called the Euclidean length of $\underline{x}$; and

$$p=\infty, \quad ||\underline{x}||_\infty = \max_i |x_i| . \tag{2.3.9}$$

Similarly, the norm of a matrix A is a non-negative number, denoted by $||A||$ and satisfies the following relations:

$$||A|| > 0 \text{ unless } A=0 ,$$

$$||kA|| = |k|\ ||A||, \text{ where } k \text{ is a complex scalar,}$$

$$||A+B|| \leq ||A|| + ||B||$$

and

$$||AB|| \leq ||A||.||B||.$$

Definition (2.1)

A matrix norm $||A||$ is said to be compatible with a vector norm $||\underline{x}||$ if

$$||A\underline{x}|| \leq ||A||\ ||\underline{x}||. \tag{2.3.10}$$

From the above definition, matrix norms compatible with the vector norms can be derived. Using (2.3.10) it is logical to express

$$||A|| = \sup_{\underline{x} \neq \underline{0}} \frac{||A\underline{x}||}{||\underline{x}||} \tag{2.3.11}$$

which is equivalent to

$$||A|| = \sup_{||\underline{y}||=1} ||A\underline{y}|| \tag{2.3.12}$$

where 'sup' denotes the least upper bound for all $\underline{x} \neq \underline{0}$.

Definition (2.2)

A matrix norm constructed by means of (2.3.12) is said to be *subordinate* to the corresponding vector norm.

Theorem (2.1)

The subordinate norms associated with the 1,2 and $\infty$ vector norms are

$$||A||_1 = \max_j \sum_{i=1}^{n} |a_{i,j}| \quad \text{(maximum absolute column sum)}$$

$$||A||_2 = \{\text{maximum eigenvalue of } A^T A\}^{\frac{1}{2}} \quad \text{(spectral norm)}$$

$$||A||_\infty = \max_i \sum_{j=1}^{n} |a_{i,j}| \quad \text{(maximum absolute row sum)}$$

The proof of the above theorem is given in Noble (1969).

Normalised Vectors

A vector is normalised if it is multiplied by a scalar to keep the size

of the components down to figures usually less or equal to unity without changing the direction of the vector. There are many methods of achieving this. For example, if $\underline{x}$ is an n-component vector with $x_i$, i=1,2,...,n, then we determine a scalar $\alpha=(\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$ and the normalised vector is now given as,

$$\underline{x} = (\frac{x_1}{\alpha}, \frac{x_2}{\alpha}, \ldots, \frac{x_n}{\alpha})^T \quad . \tag{2.3.13}$$

In this case, the modulus of every element of the normalised vector is less than 1, and the sum of squares of the elements add up to 1. Also, $\underline{x}^T \underline{x} = 1$.

Irreducibility Property of Matrices

Definition (2.3)

An (n×n) matrix A is said to be irreducible (indecomposable) if for n≥2 there does not exist an (n×n) permutation matrix P such that $PAP^T$ has the form,

$$PAP^T = \begin{bmatrix} B & C \\ O & D \end{bmatrix} \tag{2.3.14}$$

where B, D are square sub-matrices.

The concept of irreducibility is best illustrated by the use of some elementary notions of graph theory.

Definition (2.4)

Let $A=(a_{i,j})$ be an (n×n) matrix and let us consider any n distinct points $N_1, N_2, \ldots, N_n$, called nodes, in the plane. If the element $a_{i,j}$ of the matrix A is non-zero, we connect the node $N_i$ to $N_j$ as shown in Figure (2.7a). If $a_{i,i} \neq 0$, we get a loop from $N_i$ to itself as shown in Figure (2.7b). By this process, with every (n×n) matrix A we can associate a finite *directed graph* G(A) of A.

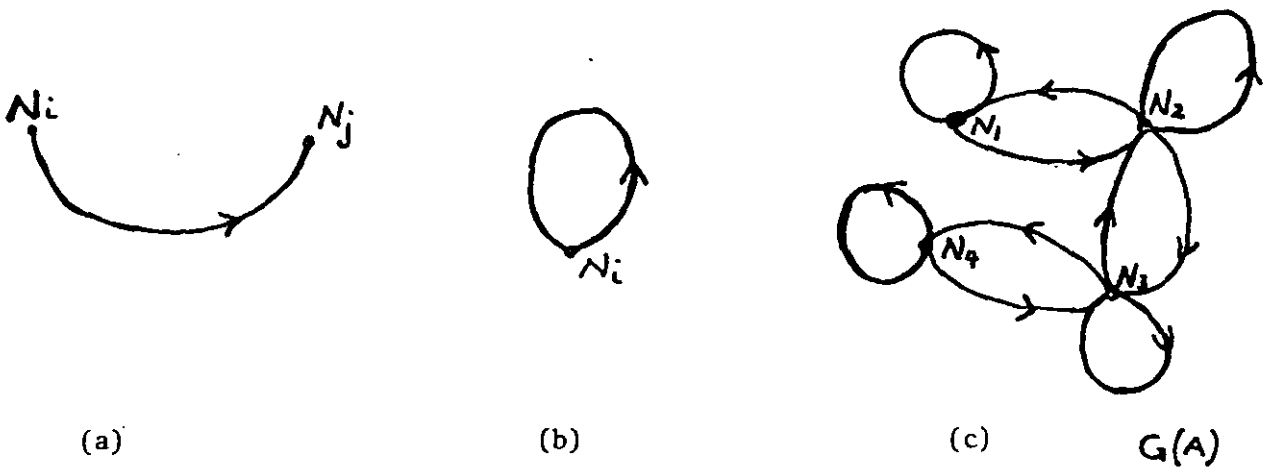



FIGURE 2.7

Example

Consider a (4×4) tridiagonal matrix,

$$A = \begin{bmatrix} 2 & 1 & & \\ 1 & 2 & 1 & \\ & 1 & 2 & 1 \\ & & 1 & 2 \end{bmatrix} \qquad (2.3.15)$$

The associated graph G(A) of the matrix A in (2.3.15) is given in Figure (2.7c).

Definition (2.5)

A directed graph is strongly connected, if, for every ordered pair of nodes $N_i$ and $N_j$ there exists a directed path

$$\overrightarrow{N_i N_{k_1}}, \quad \overrightarrow{N_{k_1} N_{k_2}}, \ \ldots \quad \overrightarrow{N_{k_{s-1}} N_{k_{s=j}}}$$

connecting $N_i$ and $N_j$.

The directed graph G(A) of the matrix in (2.3.15) is strongly connected.

Theorem (2.2)

An (n×n) matrix A is irreducible if and only if its directed graph G(A) is strongly connected.

Corollary

If A is an $(n \times n)$ tridiagonal matrix with $a_{i,j} \neq 0$ for $|i-j| \leq 1$, then A is irreducible.

The proof of Theorem (2.2) and its Corollary can be established by induction and by making use of definition (2.5). The Theorem indicates the equivalence of the matrix property of irreducibility with the concept of the strongly connected directed graph of a matrix.

Diagonal Dominance Property of Matrices

Definition (2.6)

A matrix $A=(a_{i,j})$ of order n is said to be *diagonally dominant* if

$$|a_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}| \quad , \quad 1 \leq i \leq n \ . \tag{2.3.16a}$$

A is said to be *strictly diagonally dominant* if

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}| \quad , \quad 1 \leq i \leq n \ ; \tag{2.3.16b}$$

and A is *irreducibly diagonally dominant* if A is irreducible and diagonally dominant, with strict inequality as in (2.3.16b) for at least one i.

Positive and Non-Negative Definite Matrices

Definition (2.7)

If A is a real matrix and $\underline{x}$ is a complex vector then A is said to be *positive definite* if

$$(\underline{x}, A\underline{x}) > 0 \quad \text{for all } \underline{x} \neq 0$$

where, for n-component complex vectors $\underline{x}$ and $\underline{y}$, we define

$$(\underline{x}, \underline{y}) = \sum_{i=1}^{n} x_i \bar{y}_i \ ;$$

where $\bar{y}_i$ is the complex conjugate of $y_i$.

Definition (2.8)

If A is real and $\underline{x}$ is complex, then A is *non-negative (positive-semi)* definite if $(\underline{x},A\underline{x})\geq 0$ for all $\underline{x}\neq 0$ with equality for at least one $\underline{x}\neq 0$ (Mitchell (1969), P8).

Theorem (2.3)

A real matrix A is positive (non-negative) definite if and only if it is symmetric and all its eigenvalues are positive (non-negative with at least one eigenvalue equal to zero) (see Noble (1969), P393).

Theorem (2.3) is sometimes used as a definition of positive (non-negative) definiteness. Its proof is given in Noble (1969).

The Jordan Canonical Form

A general matrix cannot always be reduced to a diagonal form by a similarity transformation. The Jordan canonical form of a matrix, even though it is of little computational importance, is the most compact form to which a matrix may be reduced by a similarity transformation. It helps to illustrate the structure of the systems of eigenvectors.

Definition (2.9)

A simple Jordan submatrix of a matrix A is a matrix of the form

$$J_r(\lambda_i)=\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & 0 \\ & & \ddots & \ddots \\ & 0 & & 1 \\ & & & \lambda_i \end{bmatrix} \qquad (2.3.17)$$

where $J_r(\lambda_i)$ is an $(r\times r)$ matrix with an eigenvalue $\lambda_i$ of multiplicity r, but only one eigenvector $\underline{x}$ where $\underline{x}^T=(1,0,\ldots,0)$.

Theorem (2.4)

If A is an $(n\times n)$ matrix having s distinct eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_s$ of multiplicity $m_1,m_2,\ldots,m_s$ such that

$$\sum_{i=1}^{s} m_i = n ,$$

then there exists a non-singular matrix P such that $J=P^{-1}AP$ has simple Jordan sub-matrices $J_r(\lambda_i)$ isolated along the diagonal with all other elements equal to zero. If there are k submatrices of order $r_j$, $j=1,2,\ldots,k$, associated with any $\lambda_i$ then

$$\sum_{j=1}^{k} r_j = m_i .$$

The matrix $J=P^{-1}AP$ which is a block diagonal matrix composed of Jordan simple matrices is the *Jordan canonical* form of A and it is unique apart from the ordering of the submatrices along the diagonal. The total number of independent eigenvectors is equal to the number of sub-matrices in the Jordan canonical form.

If an (n×n) matrix A has n distinct eigenvalues its Jordan canonical form is diagonal and its associated eigenvectors are unique and linearly independent. If A does not have n distinct eigenvalues, it may or may not possess n independent eigenvectors. If there are fewer than n linearly independent eigenvectors then the matrix is said to be *defective*.

A matrix for which there is more than one Jordan submatrix (which implies more than one eigenvector) associated with $\lambda_i$ for some value of i is said to be *derogatory*.

Eigenvalues and Eigenvectors of a Matrix

It is often convenient, from both theoretical and practical points of view, to locate the eigenvalues of a given matrix in bounded regions of the complex plane. Information of this type is useful in the subsequent application of a number of iterative methods for obtaining more precise eigenvalues. A fundamental result is the following theorem often called *Gershgorin circle theorem*. It is stated as Theorem (2.5) below; while the proof can be found in Varga (1962), or Noble (1969).

Theorem (2.5)

If $A=(a_{i,j})$ is an arbitrary $(n\times n)$ matrix, then all the eigenvalues of A lie within the union of the discs,

$$|z-a_{i,i}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| \quad , \quad 1\leq i\leq n. \tag{2.3.18}$$

Corollary

If A is an arbitrary $(n\times n)$ matrix with eigenvalues $\lambda_i$, $1\leq i\leq n$, and

$$\nu = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{i,j}|$$

$$\nu' = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{i,j}| \ ,$$

then $\rho(A)\leq \min(\nu,\nu')$

where $$\rho(A) = \max_{1\leq i\leq n} |\lambda_i| \ . \tag{2.3.19}$$

This Corollary follows from Theorem (2.5) and the fact that A and $A^T$ have the same eigenvalues.

Definition (2.10)

A set of vectors $\underline{x}_1,\underline{x}_2,\ldots,\underline{x}_n$ are said to form an orthogonal set if

$$(\underline{x}_i,\underline{x}_j) = 0 \qquad i\neq j.$$

Theorem (2.6)

If a matrix has n distinct eigenvalues, then the corresponding normalised eigenvectors $\underline{x}_i$ $(i=1,2,\ldots,n)$ form an *orthonormal* (orthogonal and normalised) set, i.e.,

$$\left.\begin{aligned} \underline{x}_i^T.\underline{x}_j &= 0 \qquad i\neq j \\ \text{and} \qquad \underline{x}_i^T.\underline{x}_i &= 1 \ . \end{aligned}\right\} \tag{2.3.20}$$

Proof

If we introduce a matrix $X=(\underline{x}_1,\underline{x}_2,\ldots,\underline{x}_n)$ whose columns are the orthonormal eigenvectors $\underline{x}_i$, then using (2.3.20) it is easily shown that

$$X^T X = I$$

and thus by (2.3.4) X is an orthogonal matrix.

Theorem (2.7)

Suppose a matrix A has n linearly independent eigenvectors, i.e.,

$$A\underline{x}_i = \lambda_i \underline{x}_i \quad ;$$

and suppose we introduce a matrix,

$$X = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$$

and a diagonal matrix,

$$\Lambda = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}$$

then

$$X^{-1}AX = \Lambda \tag{2.3.21}$$

and $$A = X\Lambda X^{-1} \quad ;$$

and if X is orthogonal (i.e. $X^T X = I$) then,

$$X^T AX = \Lambda \tag{2.3.22}$$

and $$A = X\Lambda X^T \quad .$$

Proof

Since $A\underline{x}_i = \lambda \underline{x}_i$, i=1,2,...,n then using the standard rules for manipulating partitioned matrices we have,

$$\begin{aligned} AX &= A[\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n] \\ &= [A\underline{x}_1, A\underline{x}_2, \ldots, A\underline{x}_n] \\ &= [\lambda_1 \underline{x}_1, \lambda_2 \underline{x}_2, \ldots, \lambda_n \underline{x}_n] \\ &= X\Lambda . \end{aligned}$$

Since the $\underline{x}_i$ are independent, X can be inverted and thus the results in (2.3.21) and (2.3.22) follow immediately.

An important generalisation of Theorem (2.7) is the following:

Theorem (2.8)

A general square matrix A can be reduced to diagonal form by a similarity transformation if and only if A possesses n linearly independent eigenvectors.

An immediate consequence of Theorem (2.8) is that if A is symmetric it always has n linearly independent eigenvectors and hence can always be diagonalised even if it has multiple eigenvalues.

Similarity Transformation

If a matrix A is transformed to the form $R^{-1}AR$, where R is a non-singular matrix, then this is known as a similarity transformation and the matrix A and $R^{-1}AR$ are said to be similar.

The main usefulness of similarity transformation is that the eigenvalues of a matrix are invariant under such a transformation. This can be shown very simply.

If $$A\underline{x} = \lambda\underline{x}$$

then $$P^{-1}A\underline{x} = \lambda P^{-1}\underline{x} ,$$

i.e., $$(P^{-1}AP)P^{-1}\underline{x} = \lambda P^{-1}\underline{x} ,$$

which shows that $P^{-1}AP$ has the same eigenvalues $\lambda$ as A but the former has an eigenvector which is pre-multiplied by $P^{-1}$.

Many transformation techniques form the basis of a number of methods for determining eigenvalues. We shall discuss here only the LR transformation due to Rutishauser (1958), a specialised form of which is employed in the determination of the eigenvalues of periodic tridiagonal matrices as proposed in Chapter 7.

The LR Transformation

This is a similarity transformation which consists of the factorisation of the matrix A (say) into the form,

$$A = LR , \qquad (2.3.23)$$

where usually, L is a unit lower triangular matrix and R is an upper

triangular matrix. The similarity transformation of A is then defined by

$$L^{-1}AL = L^{-1}(LR)L = RL \ .$$

Thus if the original matrix is $A_1$ then the LR-transformation method results in a series of similar matrices $\{A_s\}$ such that,

$$A_{s-1} = L_{s-1}R_{s-1} \ ,$$

and

$$A_s = R_{s-1}L_{s-1} \ .$$

It is shown in Rutishauser (1958) that under certain convergence conditions,

$$L_s \to I$$

and

$$R_2 \to \text{upper triangular matrix}$$

with the eigenvalues of A situated on the main matrix diagonal. This method has a number of drawbacks, such as slow convergence. Special techniques including a shift of origin and pivoting strategies have been introduced (Wilkinson (1965)) to obtain a modified and more efficient algorithm. In Chapter 7 a sparse, periodic matrix factorisation strategy will be incorporated into the Rutishauser's type of transformation to produce new methods for calculating the eigenvalues of periodic tridiagonal matrices.

## 2.4 BASIC THEORY OF CONTINUED FRACTIONS

The theory of continued fraction is treated in detail in the text of H.S. Wall (1948). The numerical aspects of the topic, including methods for estimating errors when calculating a continued fraction is the subject of Blanch (1964). Here we outline the basic ideas of the subject as background material which will be useful in Chapters 3 and 7.

Definition (2.11)

$$\text{Let} \quad T_0(\omega) = b_0 + \omega$$
$$T_p(\omega) = \frac{a_p}{b_p + \omega}, \qquad p=1,2,\ldots \tag{2.4.1}$$

such that $T_0, T_1, T_2, \ldots$ satisfy the linear transformation,

$$T_0T_1(\omega) = T_0(T_1(\omega)), \tag{2.4.2}$$
$$T_0T_1T_2(\omega) = T_0\{T_1T_2(\omega)\} = T_0[T_1\{T_2(\omega)\}] \text{ etc.}$$

Further let us define the quantity $\Gamma_\infty(\omega)$ as,

$$\Gamma_\infty(\omega) = \prod_{i=0}^{\infty} T_i(\omega) \tag{2.4.3}$$
$$= b_0 + \mathop{K}_{p=1}^{\infty} \frac{a_p}{b_p}$$

where

$$\mathop{K}_{p=1}^{\infty} \frac{a_p}{b_p} = \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \ddots + \cfrac{a_p}{b_p + \cfrac{a_{p+1}}{b_{p+1} + \ddots}}}}} \tag{2.4.4a}$$

and for compactness, may be rewritten as,

$$\mathop{K}_{p=1}^{\infty} \frac{a_p}{b_p} = \frac{a_1}{b_1+}\ \frac{a_2}{b_2+}\ \frac{a_3}{b_3+\ldots+}\ \frac{a_p}{b_p+\ldots}\ . \tag{2.4.4b}$$

Then, $\Gamma_\infty(\omega)$ is called an *infinite continued fraction.*

Next we define,

$$\Gamma_m(\omega) = \lim_{m\to\infty} \prod_{p=0}^{m} T_p(\infty) \tag{2.4.5a}$$

and $$\Gamma_m(0) = \lim_{m\to\infty} \prod_{p=0}^{m} T_p(0) \qquad (2.5.b)$$

If $\Gamma_m(0)$ $(=\Gamma_m(\infty)$, both being limits of an infinite sequence of images of a fixed point $\omega$ at infinity under the transformations $T_i$) exists and finite, then, it is defined as the value of the infinite continued fraction.

The elements $a_p, b_p$ are the elements of the continued fraction and can, in general, be complex numbers. The quotient $a_p/b_p$ is called the $p^{th}$ *partial quotient*, where $a_p, b_p$ are the $p^{th}$ *partial numerator* and *partial denominator* respectively.

The quantity $\Gamma_n(0) = b_0 + \underset{p=1}{\overset{n}{K}} \frac{a_p}{b_p}$ or

$$\Gamma_n(0) = b_0 + \frac{a_1}{b_1+} \frac{a_2}{b_2+} \frac{b_3}{b_3+\ldots} \frac{a_n}{b_n} \qquad (2.4.6)$$

is the $n^{th}$ *approximant* or *convergent*.

It is shown by induction (Wall, (1948)) that

$$\Gamma_n(\omega) = \prod_{i=0}^{n} T_i(\omega) = \frac{A_{n-1}\omega + A_n}{B_{n-1}\omega + B_n} \quad \text{for } n=0,1,2,\ldots \qquad (2.4.7)$$

where $A_{n-1}, A_n, B_{n-1}, B_n$ are independent of $\omega$ and are given by the following recurrence formulae,

$$\left.\begin{aligned} A_{-1} &= 1 \ , \quad B_{-1} = 0 \ , \\ A_0 &= b_0 \ , \quad B_0 = 1 \ , \\ A_{p+1} &= b_{p+1}A_p + a_{p+1}A_{p-1} \\ \text{and} \quad B_{p+1} &= b_{p+1}B_p + a_{p+1}B_{p-1} \end{aligned}\right\} p=0,1,2,\ldots \qquad (2.4.8)$$

From (2.4.7) it follows immediately that

$$\Gamma_n(0) = \frac{A_n}{B_n}$$

where,

$A_n$ is called the $n^{th}$ numerator of the continued fraction,

$B_n$ is the $n^{th}$ denominator of the continued fraction,

and $A_n/B_n$ represents the $n^{th}$ approximant.

Theorem (2.9)

An infinite continued fraction (2.4.3) is said to be *convergent* and its value exists if the following conditions are satisfied:

(a) At most a finite number of its denominators, $B_n$ vanish.

(b) Given a positive small number $\varepsilon$, there exists a number N, such that for $n \geq N$

$$\left| \frac{A_n}{B_n} - \frac{A_{n+k}}{B_{n+k}} \right| < \varepsilon \ , \quad \text{for all positive } k.$$

Otherwise it is said to be *divergent*.

If the continued fraction is convergent, then its value $\Gamma$ is given by

$$\Gamma = \lim_{n \to \infty} \frac{A_n}{B_n} \ . \tag{2.4.9}$$

For a divergent continued fraction, the value is not defined.

Periodic Continued Fraction

Definition (2.12)

Let

$$\tau^{(n)}(\omega) = \frac{a_1}{b_1+} \frac{a_2}{b_2+} \frac{a_3}{b_3+\cdots} \frac{a_n}{b_n+\omega} \tag{2.4.10}$$

be a continued fraction of level n.

Further let,

$$\Psi(\tau^{(n)}(\omega)) \equiv \underbrace{\frac{a_1}{b_1+} \frac{a_2}{b_2+} \frac{a_3}{b_3+\cdots} \frac{a_n}{b_n+}}_{\text{1st level}} \underbrace{\frac{a_1}{b_1+} \frac{a_2}{b_2+} \frac{a_3}{b_3+\cdots} \frac{a_n}{b_n+\cdots}}_{\text{2nd level}} \cdots \tag{2.4.11}$$

be an infinite continued fraction whose partial numerators and denominators are periodically repeated after every n levels of division. Then such an expression is said to be an *infinite periodic continued fraction.* It is also said to be generated by the linear fractional transformation, $\tau^{(n)}(\omega)$, given in (2.4.10).

As in (2.4.7) $\tau^{(n)}(\omega)$ can be expressed in the form,

$$\tau^{(n)}(\omega) = \frac{A_{n-1}\omega + A_n}{B_{n-1}\omega + B_n}$$

where $A_n, B_n$ denote the $n^{th}$ numerator and denominator of the continued fraction respectively and are given by the expression in (2.4.8) with $A_0=0$.

Definition (2.13)

Let the points $\delta$ satisfy the equation

$$\delta = \frac{A_{n-1}\delta + A_n}{B_{n-1}\delta + B_n} \quad .$$

The values of $\delta$, which in general, are the two roots $\delta_1, \delta_2$ of the quadratic equation,

$$B_{n-1}\delta^2 + (B_n - A_{n-1})\delta - A_n = 0 \quad , \tag{2.4.12}$$

are called the *fixed points* of the transformation (2.4.10) which generates the infinite continued fraction, (2.4.11).

Finally, we state an important theorem whose proof is in Wall (1948), p.37, and which forms the basis of the computational methods adopted later in Chapters 3 and 7 for the numerical evaluation of periodic continued fractions.

Theorem (2.10)

Let $\delta_1, \delta_2$ be the *fixed points* of the transformation (2.4.10) where $a_i, b_i$, i=1,2,...,n, are any real or complex numbers, with $a_i \neq 0$. Let $A_k/B_k$ denote the $k^{th}$ approximant of the periodic continued fraction (2.4.11). Then (2.4.11) converges if and only if $\delta_1$ and $\delta_2$ are finite numbers satisfying *one* of the following two conditions:

(a) $\delta_1 = \delta_2$

(b) $\left|\frac{A_{n-1}}{B_{n-1}} - \delta_2\right| > \left|\frac{A_{n-1}}{B_{n-1}} - \delta_1\right|$ , $\frac{A_p}{B_p} \neq \delta_2$ for p=0,1,...,n-1.

If the continued fraction converges, its value is equal to $\delta_1$.

Periodic Continued Fraction Associated with Cyclic Factorisation of a Periodic Tridiagonal Matrix

For a periodic tridiagonal matrix A of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \qquad (2.4.13)$$

let us define and associate with the matrix A the infinite periodic continued fraction (P.C.F.),

$$\Gamma_A = \underbrace{\frac{\alpha_1}{\beta_1-}\frac{\alpha_2}{\beta_2-}\frac{\alpha_3}{\beta_3-\cdots}\frac{\alpha_n}{\beta_n-}}_{\text{1st cycle}}\underbrace{\frac{\alpha_1}{\beta_1-}\frac{\alpha_2}{\beta_2-\cdots}\frac{\alpha_n}{\beta_n-\cdots}}_{\text{2nd cycle}}\cdots \qquad (2.4.14)$$

where,

$$\left.\begin{aligned} &\alpha_1 = a_1 c_n, \quad \beta_1 = b_n, \\ &\left.\begin{aligned} \alpha_i &= c_{(n-i+1)}a_{(n-i+2)} \\ \beta_i &= b_{(n-i+1)} \end{aligned}\right\} \; i=2,3,\ldots,n \end{aligned}\right\} \qquad (2.4.15)$$

and $(j) \equiv j \pmod{n}$.

The infinite periodic continued fraction (2.4.14) can arise, for example, in a cyclic factorisation of the periodic tridiagonal matrix (2.4.13). Our immediate interest is to establish the conditions of the matrix A under which the associated P.C.F. (2.4.14) is always guaranteed to converge. Before then, we state the following useful theorems:

Theorem (2.11)

A continued fraction is unchanged in value if some partial numerator and partial denominator, along with the immediately succeeding partial numerator, are multiplied by the same non-zero constant.

Such a transformation is termed in Wall (1948) an *equivalence transformation*.

Proof

The proof of this theorem is given in Wall (1948).

By using successive equivalence transformations, the P.C.F. (2.4.14) can be transformed to a form which has unitary partial denominators and hence can be written as,

$$\Gamma' = \frac{\gamma_1}{1-}\frac{\gamma_2}{1-}\frac{\gamma_3}{1-\ldots}\ \frac{\gamma_n}{1-}\frac{\gamma_1}{1-}\frac{\gamma_2}{1-}\frac{\gamma_3}{1-\ldots}\ -\ \frac{\gamma_n}{1-\ldots} \tag{2.4.16}$$

where

$$\left.\begin{aligned} \gamma_1 &= \alpha_1/\beta_1 \ , \\ \gamma_i &= \alpha_i/\beta_{i-1}\beta_i \ , \qquad \beta_i, \beta_{i-1} \neq 0,\ i=2,3,\ldots,n. \end{aligned}\right\} \tag{2.4.17}$$

Theorem (2.12)

A sufficient condition for the convergence of the continued fraction,

$$\tau = \frac{\gamma_1}{1-}\frac{\gamma_2}{1-}\frac{\gamma_3}{1-\ldots}\ \frac{\gamma_n}{1-\ldots}$$

is that

$$0<\gamma_i\leq\frac{1}{4} \tag{2.4.18}$$

and the value of the continued fraction $\tau$ satisfies the condition,

$$0<\tau\leq\frac{1}{2} \ .$$

Proof

The proof of this theorem is given, by induction, in Blanch (1964).

Theorem (2.13)

For any diagonally dominant periodic tridiagonal matrix of the form (2.4.13) such that $|c_i/b_i|, |a_i/b_i| \leq \frac{1}{4}$, the associated continued fraction of the form (2.4.14) always converges.

Proof

On using the continued fraction (2.4.16), obtained from an equivalence transformation of the continued fraction (2.4.14), and by applying the convergence condition (2.4.18) of Theorem (2.12), we immediately obtain the sufficient conditions under which the P.C.F. (2.4.14) is always convergent, i.e.,

$$\left.\begin{aligned} &0 < \frac{\alpha_1}{\beta_1} \leq \frac{1}{4} \\ \text{and}\quad &0 < \frac{\alpha_i}{\beta_{i-1}\beta_i} \leq \frac{1}{4},\ i=2,\ldots,n \end{aligned}\right\} \qquad (2.4.19)$$

A substitution of $\alpha_i$ and $\beta_i$ from (2.4.15) into (2.4.19) gives,

$$\left.\begin{aligned} &0 < \frac{c_n a_1}{b_n} \leq \frac{1}{4} \\ \text{and}\quad &0 < \frac{c_{(n-i+1)}\, a_{(n-i+2)}}{b_{(n-i+1)}\, b_{(n-i)}} \leq \frac{1}{4},\ i=2,3,\ldots,n. \end{aligned}\right\} \qquad (2.4.20)$$

If $b_1>0$, then $\frac{c_n a_1}{b_n b_1} < \frac{c_n a_1}{b_n} < \frac{1}{4}$

and hence (2.4.20) can be expressed in the unified form,

$$0 < \frac{c_{(n-i+1)}}{b_{(n-i+1)}} \frac{a_{(n-i+2)}}{b_{(n-i)}} \leq \frac{1}{4},\ i=1,2,\ldots,n. \qquad (2.4.21)$$

By definition (2.6) the elements $a_i, b_i, c_i$ of an $(n\times n)$ diagonally dominant periodic tridiagonal matrix (2.4.13) satisfy the condition,

$$\left|\frac{c_i}{b_i}\right| + \left|\frac{a_i}{b_i}\right| \leq 1, \qquad i=1,2,\ldots,n. \qquad (2.4.22)$$

Since A is diagonally dominant and in addition,

$$\left|\frac{c_i}{b_i}\right| \leq \frac{1}{4},\ \left|\frac{a_i}{b_i}\right| \leq \frac{1}{4},\ i=1,2,\ldots,n, \qquad (2.4.23)$$

then the sufficient condition (2.4.21) for the convergence of the P.C.F. (2.4.14) is immediately satisfied and the proof of the theorem is completed.

# CHAPTER 3

# FAST ALGORITHMIC METHODS FOR THE SOLUTION OF PERIODIC TRIDIAGONAL MATRIX EQUATIONS

## 3.1 INTRODUCTION

We shall consider in this chapter, several algorithms for the solution of the system of equations,

$$A\underline{u} = \underline{d} \tag{3.1.1}$$

where, in general, A is an (n×n) diagonally dominant cyclic tridiagonal matrix of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & & a_1 \\ a_2 & b_2 & c_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & c_{n-1} \\ c_n & & & & a_n & b_n \end{bmatrix} \tag{3.1.2}$$

The general cyclic tridiagonal matrix equation in which the coefficient matrix A is of the form (3.1.2) arises, for example, in the solution of elliptic p.d.e.'s with periodic conditions when a non-uniform mesh size is used in the finite difference approximation process. They also arise in the finite difference representation of self-adjoint parabolic equations (see section 5.6) and in spline applications.

In section (3.2) we present various fast direct algorithmic methods for the solution of (3.1.2) by employing the different strategies, including Gaussian elimination, generalised sparse cyclic factorisation, generalised cyclic reduction and rank-one perturbation methods. A comparison of the speed and accuracy of the resulting algorithms is given numerically in section (3.3).

In many applications, when a uniform mesh size is used in the finite difference approximation of a given p.d.e., we obtain the more usual case in which the coefficient matrix is symmetric and circulant, and of the form,

$$A = \begin{bmatrix} b & a & & & a \\ a & b & a & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a \\ a & & & a & b \end{bmatrix} \quad . \qquad (3.1.3)$$

Quite often such constant term matrix systems have been solved by Gaussian elimination or one of its variants with the possibility of giving them more generality than is required to obtain fast solutions. Thus, in section (3.4), special purpose algorithms are presented which take advantage of the constant element structure of the coefficient matrix.

We shall, in general, assume that the system of equations have to be solved a number of times with different right-hand sides, but with the coefficient matrix unchanged; so that any coefficients required by the solution algorithm can be pre-computed and stored. We refer to such coefficients, which need to be computed only once, as pre-computed coefficients. The number of such coefficients in the algorithms presented is indicated, but is not, in general, included in our estimate of the arithmetic operation count required for the implementation of each algorithm.

## 3.2 ALGORITHMS FOR THE SOLUTION OF THE GENERAL PERIODIC TRIDIAGONAL SYSTEMS OF EQUATIONS

### Algorithm (3.1)

Consider system (3.1.1) which, in matrix form, is written as,

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} , \qquad (3.2.1)$$

Let us assume that $u_n$ is known; then the other elements, $u_1, u_2, \ldots u_{n-1}$ of the vector $\underline{u}$ are obtained in terms of $u_n$ by considering the first (n-1) equations of (3.2.1).

The first equation of (3.2.1) can be rewritten in the form,

$$u_1 - q_1 u_2 - s_1 u_n = h_1 \qquad (3.2.2)$$

where, on denoting $p_1$ by $1/b_1$, then we have,

$$q_1 = -c_1 p_1 \ ,$$

$$s_1 = -a_1 p_1 \ ,$$

and $$h_1 = d_1 p_1 \ .$$

By applying the Gaussian elimination method without a pivoting strategy the second equation of (3.2.1), with $u_1$ eliminated, is changed to the form

$$u_2 - q_2 u_3 - s_2 u_n = h_2 \ , \qquad (3.2.3)$$

where $$p_2 = 1/(b_2 + a_2 q_1) \ ,$$

$$q_2 = -p_2 c_2 \ ,$$

$$s_2 = -p_2 s_1 a_2$$

and $$h_2 = p_2(d_2 - a_2 h_1) \ .$$

Similarly, by using the new second equation (3.2.3) as the pivotal equation, the $u_2$ term is eliminated from the $3^{rd}$ equation of the original system.

In general, by successive eliminations of

$u_1$ from the second equation,

$u_2$ from the third equation,

.....

and finally, $u_{n-1}$ from the $n^{th}$ equation,

we obtain the following reduced systems,

$$u_k - q_k u_{k+1} - s_k u_n = h_k, \quad k=1,2,\ldots,n-1, \qquad (3.2.4)$$

where,

$$\left.\begin{aligned} & q_0 = 0, \quad s_0 = 1, \quad h_0 = 0, \\ & p_k = \frac{1}{b_k + a_k q_{k-1}}, \\ & q_k = -c_k p_k, \\ & s_k = -s_{k-1} a_k p_k, \\ \text{and} \quad & h_k = (d_k - a_k h_{k-1}) p_k \end{aligned}\right\} \quad k=1,\ldots,n-1. \qquad (3.2.5)$$

Effectively, the matrix equation (3.2.1) has been reduced by the above process to the form,

$$\begin{bmatrix} 1 & -q_1 & & & -s_1 \\ & 1 & -q_2 & & -s_2 \\ & & \ddots & \ddots & \vdots \\ & & & 1 & -q_{n-2} \\ & & & & -s_{n-2} \\ & & & 1 & -(q_{n-1}+s_{n-1}) \\ c_n & & & a_n & b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n-1} \\ d_n \end{bmatrix},$$

from which it is now possible to solve for $u_k, (k=1,\ldots,n-1)$ in terms of $u_n$. Each $u_k$ can, in general, be written as

$$u_k = t_k u_n + v_k \quad , \quad k=n-1,\ldots,2,1, \qquad (3.2.6)$$

hence on substituting (3.2.6) into (3.2.4) we obtain

$$\left.\begin{aligned} & t_k = q_k t_{k+1} + s_k \quad , \\ & v_k = q_k v_{k+1} + h_k \\ \text{and} \quad & t_n = 1, \; v_n = 0 \end{aligned}\right\} \quad k=n-1,\ldots,2,1. \qquad (3.2.7)$$

Further, by using (3.2.6) to substitute for $u_1$ and $u_{n-1}$ in the last equation of (3.2.1) we obtain $u_n$ as,

$$u_n = (d_n - a_n v_{n-1} - c_n v_1)/(c_n t_1 + a_n t_{n-1} + b_n) \; . \qquad (3.2.8)$$

With the value of $u_n$ now determined, the remaining elements of the solution vector, $u_{n-1}, u_{n-2}, \ldots, u_1$ can now be obtained by using (3.2.4) in what amounts to be a backward substitution process.

The implementation of algorithm (3.1) is given as program 1 of Appendix I, and may be summarised in an algorithmic form as follows:

(1) First, we compute the quantities,

$$\left.\begin{aligned} & q_0 = 0, \quad s_0 = 1, \quad h_0 = 0, \\ & p_k = 1/(b_k + a_k q_{k-1}), \\ & q_k = -c_k p_k \quad , \\ & s_k = -s_{k-1} a_k p_k \\ \text{and} \quad & h_k = (d_k - a_k h_{k-1}) p_k \end{aligned}\right\} \quad k=1,\ldots,n-1.$$

(2) Next, we compute the quantities,

$$\left.\begin{aligned} & t_n = 1, \quad v_n = 0, \\ & t_k = q_k t_{k+1} + s_k \quad , \\ \text{and} \quad & v_k = q_k v_{k+1} + h_k \quad . \end{aligned}\right\} \quad k=1,\ldots,n-1.$$

(3) The solution vector is then obtained from the formulae,

$$u_n = (d_n - a_n v_{n-1} - c_n v_1)/(c_n t_1 + a_n t_{n-1} + b_n)$$

and finally, $$u_k = h_k + q_k u_{k+1} + s_k u_n \quad , \qquad k=n-1,n-2,\ldots,1.$$

For the given periodic tridiagonal matrix of order n, the amount of arithmetic calculation required by this algorithm is 5n multiplications, 4n additions and 4n pre-computed coefficients (i.e., $p_k, q_k, s_k$ and $t_k$, k=0,1,...,n-1). The underlying strategy employed in this algorithm has been used by Ahlberg et al (1967) in connection with spline approximations.

In matrix notation, algorithm (3.1) is equivalent to the solution of the partitioned matrix equation,

$$\begin{pmatrix} E & \underline{f} \\ \underline{g}^T & b_n \end{pmatrix} \begin{pmatrix} \hat{\underline{u}} \\ u_n \end{pmatrix} = \begin{pmatrix} \hat{\underline{d}} \\ d_n \end{pmatrix} \qquad (3.2.9)$$

where E is the tridiagonal matrix of order (n-1) obtained by deleting the last row and column of A, and given by,

$$E = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ & 0 & & a_{n-1} & b_{n-1} \end{bmatrix} ,$$

$$\underline{f} = \begin{bmatrix} a_1 \\ 0 \\ \vdots \\ c_{n-1} \end{bmatrix}, \qquad \hat{\underline{d}} = \begin{bmatrix} d_1 \\ \vdots \\ d_{n-1} \end{bmatrix},$$

$$\underline{g}^T = (c_n, 0, \ldots, 0, a_n) \qquad \text{and} \quad \hat{\underline{u}} = (u_1, u_2, \ldots, u_{n-1})^T .$$

Manipulation of (3.2.9) gives,

$$u_n = (b_n - \underline{g}^T E^{-1} \underline{f})^{-1} (d_n - \underline{g}^T \underline{v}) \tag{3.2.10}$$

where $$\underline{v} = (v_1, v_2, \ldots, v_{n-1})^T = E^{-1} \hat{\underline{d}} \ , \tag{3.2.11a}$$

and $$\hat{u} = \underline{v} - E^{-1} \underline{f} u_n \ . \tag{3.2.11b}$$

It can be seen that equation (3.2.10) and (3.2.11) are mathematically equivalent to (3.2.8) and (3.2.4) respectively.

Algorithm (3.2)

The general sparse cyclic factorisation method

Here we propose a general factorisation of the cyclic tridiagonal matrix A given by,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \tag{3.2.12a}$$

into the product of a 'lower cyclic triangular' matrix, P and an 'upper cyclic triangular' matrix, Q such that

$$A = PQ \ , \tag{3.2.12b}$$

where

$$P = \begin{bmatrix} 1 & & & \ell_1 \\ \ell_2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & \ell_n & 1 \end{bmatrix} , \tag{3.2.13}$$

and

$$Q = \begin{bmatrix} v_1 & c_1 & & & 0 \\ & v_2 & c_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & c_{n-1} \\ 0 & & & & v_n \end{bmatrix} \quad . \qquad (3.2.14)$$

Therefore, instead of solving (3.1.1) we can now consider the alternative form,

$$PQ\underline{u} = \underline{d} \quad . \qquad (3.2.15)$$

The special merit of the above sparse factorisation approach is that both the form and sparsity of the original periodic matrix are preserved.

By forming the product PQ and equating the elements to the corresponding elements of A, a set of equations which yield the relations for the elements of P and Q is derived to give,

$$\left.\begin{aligned} \ell_i &= a_i/v_{i-1} \ , \\ v_i &= b_i - \ell_i c_{i-1} \ , \end{aligned}\right\} \quad i=1,2,\ldots,n \qquad (3.2.16)$$

and $\quad v_0 \equiv v_n \ , \ c_0 \equiv c_n \ .$

In order to evaluate $\ell_i, v_i$ $(i=1,2,\ldots,n)$, and hence complete the factorisation, one element (say $\ell_1$) must first be determined in some way before the other elements can be determined using the formula (3.2.16). We assume for the moment that $\ell_1$ has been determined in an efficient manner. Then the factorisation is completed by evaluating the other $\ell_i$ and $v_i$ terms from (3.2.16).

Derivation of algorithm solution

The solution of the factorised matrix equation (3.2.15) is obtained by considering the two alternative systems of equations,

$$P\underline{y} = \underline{d} \qquad (3.2.17a)$$

and $\quad Q\underline{u} = \underline{y} \qquad (3.2.17b)$

where $\underline{y}$ is introduced as an auxiliary vector.

In matrix form, (3.2.17a) can be written as,

$$\begin{bmatrix} 1 & & & \ell_1 \\ \ell_2 & 1 & 0 & \\ & \ddots & \ddots & \\ 0 & & \ell_n & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} . \qquad (3.2.18)$$

Then to solve for $\underline{y}$, we

multiply the $1^{st}$ equation of (3.2.18) by $-\ell_2$ and add the result to the second equation to obtain a new second equation.

Next, we

multiply the new $2^{nd}$ equation by $-\ell_3$ and add the result to the $3^{rd}$ equation to obtain a new $3^{rd}$ equation

.....

This process is continued as far as the nth equation.

Hence, the system (3.2.18) is reduced to the matrix form,

$$\begin{bmatrix} 1 & & & \phi_1 \\ & 1 & 0 & \phi_2 \\ & & \ddots & \vdots \\ 0 & & & \phi_{n-1} \\ & & & 1+\phi_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2' \\ \vdots \\ d_n' \end{bmatrix} , \qquad (3.2.19)$$

where

$$\left.\begin{aligned} \phi_i &= (-1)^{i-1} \prod_{j=1}^{i} \ell_j \\ d_i' &= d_i - \ell_i d_{i-1}' \\ d_0' &\equiv 0 \end{aligned}\right\} \quad i=1,2,\ldots,n . \qquad (3.2.20)$$

From (3.2.19), $y_i$ (i=1,2,...,n) is easily derived as,

$$\left.\begin{aligned} y_n &= d_n'/(1+\phi_n) , \\ y_i &= d_i' - \phi_i y_n , \qquad i=1,2,\ldots,n-1. \end{aligned}\right\} \qquad (3.2.21)$$

Next, we represent (3.2.17b) in the normalised matrix form as,

$$\begin{bmatrix} 1 & \theta_1 & & & \\ & 1 & \theta_2 & & 0 \\ & & \ddots & \ddots & \\ & & & \ddots & \theta_{n-1} \\ \theta_n & & 0 & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{3.2.22}$$

where

$$\left.\begin{aligned} \theta_i &= c_i/v_i \\ g_i &= y_i/v_i \end{aligned}\right\} \; i=1,\ldots,n. \tag{3.2.23}$$

Then to solve for $\underline{u}$, we

multiply the $n^{th}$ equation of (3.2.22) by $-\theta_{n-1}$ and add this to the $(n-1)^{th}$ equation to obtain a new $(n-1)^{th}$ equation.

Then we

multiply the new $(n-1)^{th}$ equation by $-\theta_{n-2}$ and add this to the $(n-2)^{th}$ equation to obtain a new $(n-2)^{th}$ equation,

.....

This process is continued until finally, we

multiply the new $2^{nd}$ equation by $-\theta_1$ and add this to the $1^{st}$ equation to obtain a new $1^{st}$ equation.

The result of the above process is the matrix equation of the form,

$$\begin{bmatrix} 1+\gamma_1 & & & & \\ \gamma_2 & 1 & & 0 & \\ \vdots & & \ddots & & \\ & & & \ddots & \\ \gamma_n & & 0 & & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1' \\ g_2' \\ \vdots \\ g_n' \end{bmatrix} \tag{3.2.24}$$

where

$$\left.\begin{aligned} \gamma_i &= (-1)^{i-1} \prod_{j=i}^{n} \theta_j \;, \quad i=1,\ldots,n \text{ (odd)} \\ &= (-1)^{i} \prod_{j=i}^{n} \theta_j \;, \quad i=1,\ldots,n \text{ (even)}, \\ g_n' &= g_n \;, \\ \text{and} \quad g_i' &= g_i - \theta_i g_{i+1}' \;, \quad i=n-1,n-2,\ldots,1. \end{aligned}\right\} \tag{3.2.25}$$

Finally, the solution vector $\underline{u}$ is obtained from (3.2.24) to give,

$$\left.\begin{aligned} u_1 &= g_1'/(1+\gamma_1) \\ u_i &= g_i'-\gamma_i u_1 , \qquad i=2,3,\ldots,n. \end{aligned}\right\} \tag{3.2.26}$$

The algorithmic procedure outlined above can be viewed algebraically as a variant of Gaussian elimination without pivoting. Both $\ell_i$ (3.2.18) and $\theta_i$ (3.2.33) were used as multipliers in the elimination process and therefore to guarantee a stable solution without pivoting for size, each one of the multipliers $\ell_i, \theta_i$ must have their absolute values less than unity; a condition readily satisfied from (3.2.16) and (3.2.23) if the coefficient matrix A is diagonally dominant.

The proposed algorithm requires 5n multiplications, 4n additions and 4n pre-computed coefficients provided that at the factorisation stage one element (e.g. $\ell_1$) of (3.2.16) is determined in the most efficient manner. One such method involves expressing and evaluating $\ell_1$ (say) as an infinite periodic continued fraction, which we now discuss below.

Determination of $\ell_1$ by the use of periodic continued fraction (P.C.F.) concepts

The element $\ell_1$ in equation (3.2.16), can be expressed as a function of the elements of the matrix A by a cyclic application of the formula (3.2.16) to give an infinite periodic continued fraction of the form,

$$c_n\ell_1 = \frac{a_1c_n}{b_n-}\ \frac{a_nc_{n-1}}{b_{n-1}-\cdots}\ \frac{a_2c_1}{b_1-}\ \frac{a_1c_n}{b_n-}\ \frac{a_nc_{n-1}}{b_{n-1}-\cdots}\ \frac{a_2c_1}{b_1-}\ \frac{a_1c_n}{b_n-\cdots} \tag{3.2.27}$$

$$\underbrace{\qquad\text{1st cycle}\qquad}\quad\underbrace{\qquad\text{2nd cycle}\qquad}\quad\underbrace{\qquad\cdots}$$

which for simplicity may be written as,

$$c_n\ell_1 = \frac{\alpha_1}{\beta_1-}\ \frac{\alpha_2}{\beta_2-\cdots}\ \frac{\alpha_n}{\beta_n-}\ \frac{\alpha_1}{\beta_1-}\ \frac{\alpha_2}{\beta_2-\cdots}\ \frac{\alpha_n}{\beta_n-}\ \frac{\alpha_1}{\beta_1-\cdots} \tag{3.2.28}$$

$$\underbrace{\qquad\text{1st cycle}\qquad}\quad\underbrace{\qquad\text{2nd cycle}\qquad}\quad\underbrace{\qquad\cdots}$$

where $\alpha_i = a_{(n-i+2)}c_{(n-i+1)}$,

$\beta_i = b_{(n-i+1)}$,

and $(k) \equiv k$ (modulo n).

The infinite P.C.F. in (3.2.28) is generated by the linear fractional transformation, (see Definition 2.12, Chapter 2), given by,

$$\tau^{(n)}(\omega) = \frac{\alpha_1}{\beta_1 -} \cdot \frac{\alpha_2}{\beta_2 -} \cdots \frac{\alpha_n}{\beta_n - \omega} \quad . \tag{3.2.29}$$

By theorem 2.10, Chapter 2, the value $\delta$, of the infinite periodic continued fraction (3.2.28) is given by

$$\delta = c_n \ell_1 = \max(\omega_1, \omega_2)$$

where $\omega_1, \omega_2$ are the roots of the quadratic equation,

$$\omega = \frac{E_{k-1}\omega + E_k}{F_{k-1}\omega + F_k}$$

or $$F_{k-1}\omega^2 + (F_k - E_{k-1})\omega - E_k = 0. \tag{3.2.30}$$

The terms $E_r, F_r$, $r=0,1,\ldots$, are given by the recurrence relations, (see equation (2.4.8) of Chapter 2):

$$\left.\begin{aligned} E_0 &= 0 , & F_0 &= 1, \\ E_1 &= \alpha_1 , & F_1 &= \beta_1 , \\ E_r &= \beta_r E_{r-1} - \alpha_r E_{r-2} , & & \\ \text{and} \quad F_r &= \beta_r F_{r-1} - \alpha_r F_{r-2} , & & \end{aligned}\right\} \quad r=2,3,\ldots k \leq n. \tag{3.2.31}$$

The two roots $\omega_1, \omega_2$ are the fixed points of the linear transformation (3.2.29) which must either be equal or must have unequal absolute value (See Theorem (2.10)).

The quotient $\frac{E_r}{F_r}$ is defined as the $r^{th}$ approximant of the P.C.F. in (2.2.28) and the sequence of approximants $\{E_r/F_r\}$ converges after the kth approximant if

$$\left| \frac{E_k}{F_k} - \frac{E_{k-1}}{F_{k-1}} \right| < \varepsilon \tag{3.2.32}$$

for a sufficiently small error tolerance $\varepsilon$.

In practical problems when the coefficient matrix A satisfies a strong diagonal dominance condition the approximants of the associated P.C.F. form a rapidly convergent sequence; and as a general rule, each

additional approximant (which implies additional level in the infinite division) yields approximately one more correct decimal place than the previous approximant. Thus, condition (3.2.32) can be satisfied and hence the sequence of approximants converge to the desired value only after the order of T levels of the infinite division, where T is the maximum number of decimal place accuracy of the computer in use.

Example

Consider for illustration the (30×30) periodic tridiagonal matrix,

$$A = \begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & 0 & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & 0 & & & \ddots & \ddots & 1 \\ 1 & & & & 1 & 4 & \end{bmatrix},$$

where A is diagonally dominant.

The linear fractional transformation which generates the infinite periodic continued fraction which we can associate with the matrix A is given by (see 3.2.27 and 3.2.29)

$$\tau^{(30)}(\omega) = \frac{1}{4-} \quad \frac{1}{4-\ldots} \quad \frac{1}{4-\omega} \qquad (3.2.33)$$

By using the recurrence formula (3.2.31), successive approximants of the continued fraction (3.2.33) are obtained and their values shown in Table (3.1).

| r | Partial Numerator $E_r$ | Partial Denominator $F_r$ | Approximants $E_r/F_r$ |
|---|---|---|---|
| 1 | 1 | 4 | 0.250 000 000 |
| 2 | 4 | 15 | 0.266 666 666 |
| 3 | 15 | 56 | 0.267 857 142 |
| 4 | 56 | 209 | 0.267 942 583 |
| 5 | 209 | 780 | 0.267 948 717 |
| 6 | 780 | 2911 | 0.267 949 158 |
| 7 | 2911 | 10864 | 0.267 949 190 |
| 8 | 10864 | 40545 | 0.267 949 192 |
| 9 | 40545 | 151316 | 0.267 949 192 |
| 10 | 151316 | 564719 | 0.267 949 192 |

TABLE 3.1

The above simple example illustrates the point that it is only necessary to evaluate the recurrence relation (3.2.31) to a level r=T where $10^{-T}$ is the desired truncation error tolerance (e.g. $10^{-12}$). This implies that despite the order of the matrix (and hence the maximum level of recurrence of the approximants) only a few levels of the recursion is necessary to achieve convergence of the form (3.2.32). Thus, the computational effort required to determine $\ell_1$, using the proposed continued fraction approach is relatively inexpensive, particularly for a large order system.

The generalised cyclic factorisation method for the solution of the periodic tridiagonal matrix equation (3.1.1) is implemented as program 2 of Appendix I. It is denoted as the PQFACT algorithm and may be summarised as follows:

PQFACT Algorithm (3.2)

Step 1 Determine $\ell_1$ as follows:-

(a) Evaluate the recurrence relation (3.2.31) for r=1,2,...T (where $\varepsilon \simeq 10^{-T}$).

(b) Obtain the real roots, $\omega_1, \omega_2$ of the quadratic equation (3.2.30), and hence compute $\ell_1$ from $\ell_1 = \frac{1}{c_n}[\max(\omega_1, \omega_2)]$.

Step 2 Evaluate the following pre-computed coefficients:-

(a) $v_i = b_i - \ell_i c_{i-1}$, $\quad i=1,\ldots,n;\ c_0 \equiv c_n$,

$\ell_i = a_i / v_{i-1}$, $\quad i=2,3,\ldots,n$,

(b) $\phi_i = (-1)^{i-1} \prod_{j=1}^{i} \ell_j$, $\quad i=1,2,\ldots n$,

(c) $\theta_i = \frac{c_i}{v_i}$, $\quad i=1,2,\ldots,n$,

(d) $\gamma_i = (-1)^{i-1} \prod_{j=i}^{n} \theta_j$, $\quad i=1,2,\ldots n$ (odd)

$= (-1)^{i} \prod_{j=i}^{n} \theta_j$, $\quad i=1,2,\ldots n$ (even).

Step 3 Compute the following:-

$$\text{(a)}\quad d'_i = d_i - \ell_i d'_{i-1}, \qquad i=1,2,\ldots n,$$

$$d'_0 \equiv 0,$$

$$\text{(b)}\quad y_n = d'_n/(1+\phi_n),$$

$$y_i = d'_i - \phi_i y_n, \qquad i=1,2,\ldots n-1,$$

$$\text{(c)}\quad g_i = y_i/v_i, \qquad i=1,2,\ldots n,$$

$$\text{(d)}\quad g'_n = g_n,$$

$$g'_i = g_i - \theta_i g'_{i+1}, \qquad i=n-1,n-2,\ldots 1,$$

and finally,

$$\text{(e)}\quad u_1 = g'_1/(1+\gamma_1)$$

and $$u_i = g'_i - \gamma_i u_1, \qquad i=2,3,\ldots n$$

where $\underline{u} = (u_1, u_2, \ldots, u_n)^T$ is the solution vector.

Implementation of the PQFACT Algorithm

In the solution of periodic parabolic p.d.e.'s (e.g. a general diffusion equation with periodic conditions (see Section 5.6)) by the use of an implicit finite difference scheme in a marching procedure, there occurs the need to solve repeatedly the general cyclic tridiagonal matrix equation of the form (3.1.1). In such a case, the coefficient matrix A remains unchanged as the solution is advanced from one time-step to the next. With this type of application in mind, the PQFACT algorithm is best implemented, as a fast solver, in the form of three simple subroutines corresponding to the three computational steps outlined above. The reason for the three subroutines (instead of a single one), is that in a typical line-by-line solution of the discretised parabolic problem, the factorisation of the coefficient matrix (which remains unchanged) and the evaluation of the associated continued fraction need to be performed only once. This leaves the subroutine for step 3 as the only section of code required for a subsequent solution in the marching procedure.

Algorithm(3.3)

The Generalised Cyclic Reduction Method

Consider further the solution of the system,

$$\underline{Au} = \underline{d} \tag{3.2.34a}$$

where A is a cyclic tridiagonal matrix of order $n=2^k$ (k is any positive integer) given by,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} . \tag{3.2.34b}$$

Here, we introduce a generalisation of the cyclic reduction method of Hockney (1965) proposed by Golub to obtain a solution of (3.2.34).

If we consider any three consecutive equations of the system (4.2.34), i.e., the $(i-1)^{th}$, $i^{th}$ and $(i+1)^{th}$ equations for i even, then we have,

$$\begin{aligned} a_{i-1}u_{i-2}+b_{i-1}u_{i-1}+c_{i-1}u_i &= d_{i-1} \\ a_iu_{i-1}+b_iu_i+c_iu_{i+1} &= d_i \\ a_{i+1}u_i+b_{i+1}u_{i+1}+c_{i+1}u_{i+2} &= d_{i+1} . \end{aligned}$$

In order to eliminate the odd terms of the solution vector, i.e., $u_{i-1}, u_{i+1}$ terms between these three consecutive equations, we multiply the $(i-1)^{th}$ equation by a constant $k_1$ and the $(i+1)^{th}$ equation by another constant, $k_2$, and then add all three consecutive equations together to obtain a new equation of the form,

$$\begin{aligned} k_1a_{i-1}u_{i-2}+(k_1c_{i-1}+b_i+k_2a_{i+1})u_i+k_2c_{i+1}u_{i+2} \\ = k_1d_{i-1}+d_i+k_2d_{i+1} \end{aligned}$$

where $k_1 = \dfrac{-a_i}{b_{i-1}}$ and $k_2 = \dfrac{-c_i}{b_{i+1}}$ .

After some simplification, the new reduced system of equation becomes,

$$a_i^{(1)}u_{i-2}+b_i^{(1)}u_i+c_i^{(1)}u_{i+2} = d_i^{(1)} \tag{3.2.35a}$$

where for i=2, step 2 until $2^k-2$, ($k=\log_2 n$),

$$\left.\begin{aligned} a_i^{(1)} &= b_{i+1}a_i a_{i-1}\,, \\ b_i^{(1)} &= b_{i+1}a_i c_{i-1} - b_{i-1}b_{i+1}b_i + c_i b_{i-1}a_{i-1}\,, \\ c_i^{(1)} &= b_{i-1}c_i c_{i+1}\,, \\ \text{and } d_i^{(1)} &= b_{i+1}a_i d_{i-1} - b_{i-1}b_{i+1}d_i + b_{i-1}c_i d_{i+1}\,. \end{aligned}\right\} \qquad (3.2.35b)$$

The reduced system of equations represented by (3.2.35a) now has $2^{k-1}$ of the unknown components of the solution vector $\underline{u}$.

If this reduction process is continued, then at the $s^{th}$ level we have the following reduced system,

$$a_i^{(s)}u_{i-2^s} + b_i^{(s)}u_i + c_i^{(s)}u_{i+2^s} = d_i^{(s)}\,, \qquad (3.2.36)$$

$$i=2^s \text{ step } 2^s \text{ until } 2^k-2^s$$

where the coefficient terms $a_i^{(s)}, b_i^{(s)}, c_i^{(s)}$ are given by,

$$\left.\begin{aligned} a_i^{(s)} &= b_{i+2^{s-1}}^{(s-1)}a_i^{(s-1)}a_{i-2^{s-1}}^{(s-1)}\,, \\ b_i^{(s)} &= b_{i+2^{s-1}}^{(s-1)}a_i^{(s-1)}c_{i-2^{s-1}}^{(s-1)} - b_{i-2^{s-1}}^{(s-1)}b_{i+2^{s-1}}^{(s-1)}b_i^{(s-1)} \\ &\qquad + c_i^{(s-1)}b_{i-2^{s-1}}^{(s-1)}a_{i+2^{s-1}}^{(s-1)}\,, \\ c_i^{(s)} &= b_{i-2^{s-1}}^{(s-1)}c_i^{(s-1)}c_{i+2^{s-1}}^{(s-1)} \\ \text{and } d_i^{(s)} &= b_{i+2^{s-1}}^{(s-1)}a_i^{(s-1)}d_{i-2^{s-1}}^{(s-1)} - b_{i-2^{s-1}}^{(s-1)}b_{i+2^{s-1}}^{(s-1)}d_i^{(s-1)} \\ &\qquad + b_{i-2^{s-1}}^{(s-1)}c_i^{(s-1)}d_{i+2^{s-1}}^{(s-1)}\,. \end{aligned}\right\} \qquad (3.2.37)$$

After k levels of reduction, we are left with only one equation which is given by,

$$a_i^{(k)}u_{i-2^k} + b_i^{(k)}u_i + c_i^{(k)}u_{i+2^k} = d_i^{(k)}\,, \qquad i=2^k \qquad (3.2.38)$$

By the cyclic nature of the reduction process, $u_0=u_n=u_{2n}$; hence, from (3.2.38) we immediately obtain,

$$u_n = d_n^{(k)}/(a_n^{(k)}+b_n^{(k)}+c_n^{(k)})\,. \qquad (3.2.39)$$

The remaining intermediate values of the solution vector are then obtained in a backward substitution process from the recursion formula (3.2.40) which is derived from (3.2.36) and given by,

$$u_i = (d_i^{(s)} - a_i^{(s)} u_{i-2^s} - c_i^{(s)} u_{i+2^s})/b_i^{(s)} \quad , \qquad (3.2.40)$$

$$s=k-1,k-2,\ldots 0,$$
$$i=2^s \text{ step } 2^{s+1} \text{ until } 2^k-2^s,$$

where the subscripts are interpreted cyclically, i.e., $u_{n+i} \equiv u_i$.

For an efficient evaluation of the recursive formula (3.2.37) it is necessary to compute the quantities,

$$b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} \quad ,$$

$$b_{i-2^{s-1}}^{(s-1)} b_{i+2^{s-1}}^{(s-1)} \quad ,$$

and

$$b_{i-2^{s-1}}^{(s-1)} c_i^{(s-1)}$$

first, then they are used where needed to compute the new variables $a_i^{(s)}$, $b_i^{(s)}$, $c_i^{(s)}$ and $d_i^{(s)}$.

The reduction stage, (3.2.37), of the outlined algorithm requires 11 multiplications, 4 additions per level of reduction, giving for $k(=\log_2 n)$ levels, the computational effort as k(11 multiplications, 4 additions).

The back substitution process requires 3n multiplications and 2n additions. Thus, the entire solution involves $(11 \log_2 n+3n)$ multiplications and $(4 \log_2 n+2n)$ additions. For repeated solutions, with the coefficient matrix remaining unchanged, the arithmetic operation count required becomes, $(3 \log_2 n+3n)$ multiplications and $(2 \log_2 n+2n)$ additions.

The implementation of this algorithm is given in Appendix I as program 3.

During the reduction stage, the intermediate quantities, $d_i^{(s)}$ (see (3.2.37)), grow rapidly in size because it involves terms which contain multiples of $b_i^{(s)}$ which also grows rapidly. This could result in overflow

or create instability in the algorithmic solution.

We must therefore discuss a modification of the cyclic reduction algorithm (see Buzbee et al (1970)) proposed by Buneman to produce a revised algorithm in which the right hand side plays the important role of determining the stability of solution. The main idea behind the Buneman modification strategy is to avoid a multiplication by $b_i^{(s)}$ in evaluating $d_i^{(s)}$ as in (3.2.37) so as to prevent the right hand side term $d_i^{(s)}$ from becoming excessively large.

Algorithm (3.4)

The Buneman Modified Algorithm

We introduce the coefficients $p_i, q_i$ such that,

$$d_i^{(s-1)} = b_i^{(s-1)} p_i^{(s-1)} + q_i^{(s-1)}, \tag{3.2.41a}$$

$$d_{i+2^{s-1}}^{(s-1)} = b_{i+2^{s-1}}^{(s-1)} p_{i+2^{s-1}}^{(s-1)} + q_{i+2^{s-1}}^{(s-1)} \tag{3.2.41b}$$

and

$$d_{i-2^{s-1}}^{(s-1)} = b_{i-2^{s-1}}^{(s-1)} p_{i-2^{s-1}}^{(s-1)} + q_{i-2^{s-1}}^{(s-1)} . \tag{3.2.41c}$$

By using (3.2.37) the term $d_i^{(s)}$ can be rewritten to involve the expression for the term $b_i^{(s)}$, resulting in an expression for $d_i^{(s)}$ as follows:-

$$d_i^{(s)} = b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} d_{i-2^{s-1}}^{(s-1)} - b_{i-2^{s-1}}^{(s-1)} b_{i+2^{s-1}}^{(s-1)} d_i^{(s-1)} + b_{i-2^{s-1}}^{(s-1)} c_i^{(s-1)} d_{i+2^{s-1}}^{(s-1)}$$

$$= (b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} c_{i-2^{s-1}}^{(s-1)} + b_{i-2^{s-1}}^{(s-1)} c_i^{(s-1)} a_{i+2^{s-1}}^{(s-1)} - b_{i-2^{s-1}}^{(s-1)} b_i^{(s-1)} b_{i+2^{s-1}}^{(s-1)})($$
$$d_i^{(s-1)}/b_i^{(s-1)}) + b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} d_{i-2^{s-1}}^{(s-1)} + b_{i-2^{s-1}}^{(s-1)} c_i^{(s-1)} d_{i+2^{s-1}}^{(s-1)}$$
$$-(d_i^{(s-1)}/b_i^{(s-1)})(b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} c_{i-2^{s-1}}^{(s-1)} + b_{i-2^{s-1}}^{(s-1)} c_i^{(s-1)} a_{i+2^{s-1}}^{(s-1)}).$$

On substituting for $(d_i^{(s-1)}/b_i^{(s-1)})$ and $d_i^{(s)}$ etc. from (3.2.41) we have the result,

$$d_i^{(s)} = b_i^{(s)}(p_i^{(s-1)} + q_i^{(s-1)}/b_i^{(s-1)}) + b_{i+2^{s-1}}^{(s-1)} a_i^{(s-1)} (b_{i-2^{s-1}}^{(s-1)} p_{i-2^{s-1}}^{(s-1)} + q_{i-2^{s-1}}^{(s-1)}$$

$$+b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}(b^{(s-1)}_{i+2^{s-1}}p^{(s-1)}_{i+2^{s+1}}+q^{(s-1)}_{i+2^{s-1}})-(b^{(s-1)}_{i+2^{s-1}}a^{(s-1)}_{i}c^{(s-1)}_{i-2^{s-1}}+$$

$$b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}a^{(s-1)}_{i+2^{s-1}})(p^{(s-1)}_{i}+q^{(s-1)}_{i}/b^{(s-1)}_{i}).$$

Further, by substituting for $b^{(s-1)}_{i-2^{s-1}}b^{(s-1)}_{i}b^{(s-1)}_{i+2^{s-1}}$ in terms of $b^{(s)}_{i}$ from (3.2.37), and simplifying the resulting expression, yields,

$$d^{(s)}_{i} = b^{(s)}_{i}p^{(s)}_{i}+q^{(s)}_{i} \tag{3.2.42}$$

where,

$$\left.\begin{aligned} p^{(s)}_{i} &= p^{(s-1)}_{i}+(q^{(s-1)}_{i}-a^{(s-1)}_{i}p^{(s-1)}_{i-2^{s-1}}-c^{(s-1)}_{i}p^{(s-1)}_{i+2^{s-1}})/b^{(s-1)}_{i} \\ \text{and}\quad q^{(s)}_{i} &= b^{(s-1)}_{i+2^{s-1}}a^{(s-1)}_{i}q^{(s-1)}_{i-2^{s-1}}+b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}q^{(s-1)}_{i+2^{s-1}}- \\ &\quad (b^{(s-1)}_{i+2^{s-1}}a^{(s-1)}_{i}c^{(s-1)}_{i-2^{s-1}}+b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}a^{(s-1)}_{i+2^{s-1}})p^{(s)}_{i}. \end{aligned}\right\} \tag{3.2.43}$$

Hence, if initially we set,

$$\left.\begin{aligned} p^{(0)}_{i} &= d_{i}/b_{i} \\ \text{and}\quad q^{(0)}_{i} &= 0 \end{aligned}\right\} \quad i=1,2,\ldots,n,$$

then the reduction process of the Buneman algorithm becomes:

for $s=1,2,\ldots,k$,

$i=2^{s}$ step $2^{s}$ until $2^{k}-2^{s}$,

$$\left.\begin{aligned} a^{(s)}_{i} &= b_{i+2^{s-1}}a^{(s-1)}_{i}a_{i-2^{s-1}} \\ b^{(s)}_{i} &= b^{(s-1)}_{i+2^{s-1}}a^{(s-1)}_{i}c^{(s-1)}_{i-2^{s-1}}-b^{(s-1)}_{i-2^{s-1}}b^{(s-1)}_{i+2^{s-1}}b^{(s-1)}_{i}+c^{(s-1)}_{i} \\ c^{(s)}_{i} &= b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}c^{(s-1)}_{i+2^{s-1}} \\ p^{(s)}_{i} &= p^{(s-1)}_{i}+(q^{(s-1)}_{i}-a^{(s-1)}_{i}p^{(s-1)}_{i-2^{s-1}}-c^{(s-1)}_{i}p_{i+2^{s-1}})/b^{(s-1)}_{i} \\ \text{and}\quad q^{(s)}_{i} &= b^{(s-1)}_{i+2^{s-1}}a^{(s-1)}_{i}q^{(s-1)}_{i-2^{s-1}}+b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}q^{(s-1)}_{i+2^{s-1}}- \\ &\quad (b_{i+2^{s-1}}a_{i}c_{i-2^{s-1}}+b^{(s-1)}_{i-2^{s-1}}c^{(s-1)}_{i}a^{(s-1)}_{i+2^{s-1}})p^{(s)}_{i}. \end{aligned}\right\} \tag{3.2.44}$$

The solution vector is then obtained by proceeding in a similar manner as

for the cyclic reduction method if $d_i^{(s)}$ is replaced by $b_i^{(s)}p_i^{(s)}+q_i^{(s)}$. Thus, using (3.2.39) and (3.2.40), the solution vector is obtained from the expressions,

$$u_n = (b_n^{(k)}p_n^{(k)}+q_n^{(k)})/(a_n^{(k)}+b_n^{(k)}+c_n^{(k)}) \qquad (3.2.45)$$

and

$$u_i = p_i^{(s)}+(q_i^{(s)}-a_i^{(s)}u_{i-2^{s-1}}-c_iu_{i+2^{s-1}})/b_i^{(s)} \quad , \qquad (3.2.46)$$

for $s=k-1,k-2,\ldots 0$

and $i=2^s$ step $2^{s+1}$ until $2^k-2^s$.

In order to evaluate the recursive formula (3.2.44) as efficiently as possible, it is necessary to calculate the quantities,

$$b_{i+2^{s-1}}^{(s-1)}a_i^{(s-1)} \quad ,$$

$$b_{i-2^{s-1}}^{(s)}c_i^{(s)}$$

and

$$b_{i+2^{s-1}}^{(s-1)}a_i^{(s-1)}c_{i-2^{s-1}}^{(s-1)}+b_{i-2^{s-1}}^{(s-1)}c_i^{(s-1)}a_{i+2^{s-1}}^{(s-1)}$$

first, since they appear at least twice in the subsequent evaluation of new values of $a_i^{(s)}, b_i^{(s)}, c_i^{(s)}, p_i^{(s)}$ and $q_i^{(s)}$. The reduction formula (3.2.44) requires 15 multiplications, 8 additions and 1 division per iteration. The backward substitution (3.2.46) requires 2 multiplications, 1 division and 3 additions for each unknown $u_i$. Altogether, the total computational requirement of the Buneman's modified algorithm for the solution of the general periodic tridiagonal system is, for $k=\log_2 n$, (15k+2n) multiplications, (k+n) divisions and (8k+3n) additions for the first solution; and for subsequent solutions only (5k+2n) multiplications, (k+n) divisions and (5k+3n) additions.

Both the cyclic reduction and Buneman's modified algorithms for solving the Poisson equation over a rectangle with periodic boundary conditions have appeared in the literature (see, for example, Hockney (1965), Buneman (1969), and Buzbee et al (1970)). Generalisations of these have also been given in Sweet (1974) and Swarztrauber (1974). However, the generalisation which

applies to the point-form general periodic tridiagonal matrix, as given in algorith(3.3) is not known to have appeared elsewhere.

Algorithm(3.5)

Rank-One Perturbation Algorithm

A general unsymmetric matrix of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} , \quad a_i c_i > 0 ,$$

can sometimes be reduced by a diagonal similarity transformation to a symmetric matrix. If we take

$$B = DAD^{-1}, \qquad D=\mathrm{diag}(d_i) , \tag{3.2.47}$$

then B is symmetric if

$$(\frac{d_{k+1}}{d_k})a_{k+1} = (\frac{d_k}{d_{k+1}})c_k , \quad \begin{cases} k=0,1,\ldots n-1, \\ d_0 \equiv d_n . \end{cases}$$

By multiplying these relations together we obtain,

$$\prod_{k=1}^{n} a_k = \prod_{k=1}^{n} c_k \qquad . \tag{3.2.48}$$

If relation (3.2.48) is satisfied then the $d_k$ are determined by the relationships,

$$d_1=1, \; d_{k+1} = \pm d_k (c_k/a_{k+1})^{\frac{1}{2}}, \quad k=1,\ldots n-1. \tag{3.2.48}$$

Hence, we shall assume that the given general periodic tridiagonal matrix equation has been reduced, using (3.2.47) to a symmetric matrix system of the form,

$$B\underline{u} = \underline{d} \tag{3.2.49}$$

where

$$B = \begin{bmatrix} b_1 & a_2 & & & a_1 \\ a_2 & b_2 & a_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a_n \\ a_1 & & & a_n & b_n \end{bmatrix}$$

Now the matrix B can be written as a rank one pertubation of a symmetric tridiagonal matrix, (Bjorck and Golub (1977)), i.e.,

$$B = T + \omega \underline{z}\,\underline{z}^T \quad , \qquad \omega = \pm a_1 \tag{3.2.50}$$

where

$$T = \begin{bmatrix} b_1 \mp a_1 & a_2 & & & \\ a_2 & b_2 & a_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a_n \\ & & & a_n & b_n \mp a_1 \end{bmatrix} \quad \text{and} \quad \underline{z} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \pm 1 \end{bmatrix} .$$

By using (3.2.50) and the well-known Sherman-Morrison formula (Householder, (1964)),

$$B^{-1} = (T+\omega \underline{z}\underline{z}^T)^{-1} = T^{-1} - \frac{T^{-1}\underline{z}\omega \underline{z}^T T^{-1}}{1+\omega \underline{z}^T T^{-1}\underline{z}} \quad ,$$

the solution of (3.2.49) can be written as a linear combination of the solution of two tridiagonal systems of equations.

Hence,

$$\underline{u} = B^{-1}\underline{d} = T^{-1}\underline{d} - \beta T^{-1}\underline{z} \tag{3.2.51}$$

where $\beta = \dfrac{\omega \underline{z}^T T^{-1}\underline{d}}{1+\omega \underline{z}^T T^{-1}\underline{z}}$ .

Since T is symmetric, $T^{-1}$ is also symmetric; hence

$$\underline{z}^T(T^{-1}\underline{d}) = (T^{-1}\underline{z})^T\underline{d} \quad .$$

Let $\quad \underline{v}^T = \omega \underline{z}^T T^{-1}$

then $\quad T\underline{v} = \omega \underline{z}$

and $\quad \beta = \dfrac{\underline{v}^T.\underline{d}}{1+\underline{v}^T\underline{z}} = \dfrac{\underline{v}^T.\underline{d}}{1+v_1+v_n}$ $\Bigg\}$ (3.2.52)

Thus, the solution of (3.2.49) is achieved by solving the two tridiagonal systems obtained from (3.2.51) and (3.2.52) and given by,

$$\left.\begin{aligned} T\underline{v} &= \omega \underline{z} \quad , \\ T\underline{u} &= \underline{d} - \beta \underline{z} \quad , \qquad \beta = \frac{\underline{v}^T \underline{d}}{1+v_1+v_n} \end{aligned}\right\} \qquad (3.2.53)$$

For the purpose of estimating the amount of arithmetic operations required to implement the algorithm we assume that the two tridiagonal matrix equations (3.2.53) are solved by Thomas algorithm (see Algorithm 4.1, Chapter 4). Then the rank-one modification algorithm requires 6n multiplications, 3n divisions, 5n additions, in addition to the calculation necessary to initially convert the non-symmetric matrix A to the symmetric matrix B. Subsequent solutions with unchanged coefficient matrix but different right-hand sides require a further 5n multiplications, 2n divisions and 4n additions.

A number of other methods for solving the general periodic tridiagonal matrix equation have been proposed (see, for example, Evans and Atkinson (1970)). However, the ones presented here are believed to be some of the best known strategies for obtaining fast and accurate solutions of the matrix equation (3.1.1).

## 3.3 COMPARISONS AND NUMERICAL RESULTS

The four algorithms presented in the previous section (i.e., algorithms (3.1), (3.2), (3.3) and (3.5)) were all programmed in Fortran and run using single precision arithmetic, on the Loughborough University ICL 19004S computer in order to compare these algorithms on the basis of efficiency and accuracy. We will generally assume that the periodic tridiagonal systems have to be solved a number of times with different right hand sides, so that any coefficients required by the solution algorithms are pre-calculated and stored. This would normally apply, for example, in the solution of time-dependent partial differential equations involving the use of marching techniques (e.g. see Section (5.6), or (5.7)).

For a numerical experiment with each algorithm, the elements of the (n×n) coefficient matrix A $(a_i,b_i,c_i)$ were generated randomly with entries in the range $0.0 \leq a_i,b_i,c_i \leq 10.0$, i=1,2,...n provided A is diagonally dominant. A random solution vector $\underline{u}$ was also generated such that $0 \leq u_i \leq 10.0$. The corresponding right hand side vector $\underline{d}=A\underline{u}$ was then computed and input into the algorithms. For each algorithm, the approximate solution was then re-computed using this right hand side.

Each algorithm was timed for 60 randomly generated right-hand side vectors and for different values of n. In order to accommodate the cyclic reduction algorithm (3.3) n was chosen to be a power of 2 even though the other algorithms are subject to no such restrictions on the values of n. The average execution times (in mill-units) required by each algorithm for the 60 right-hand side solutions are presented in Table 3.2. The results show that the cyclic reduction method is generally the fastest while the rank-one modification algorithm is the slowest with the two other algorithms intermediate in speed. However, for smaller values of n, the cyclic reduction is slowed down by its more complicated structure; whilst for very large values of n (e.g. $n=2^8$) there is the problem of overflow and

hence the need for the Buneman's modification). The relative cost of evaluating an infinite continued fraction in order to obtain one of the pre-computed coefficients makes the generalised cyclic factorisation algorithm rather slow if the solution is required for only one right-hand side vector. However, since the continued fraction need be computed only once, this no longer has very adverse slowing-down effects on the algorithm if solution is for many right-hand sides.

Execution Times (in mill-secs.) for Solution of a Periodic Tridiagonal System of Order n for 60 Right-Hand Sides

| n | Gaussian Elimination Algorithm (3.1) Program (1) | Generalised Cyclic Factorisation Algorithm (3.2) Program (2) | Cyclic Reduction Algorithm (3.3) Program (3) | Rank One Modification Algorithm (3.5) Program (5) |
|---|---|---|---|---|
| 16 | 9 | 10 | 11 | 11 |
| 32 | 20 | 20 | 19 | 21 |
| 64 | 40 | 41 | 39 | 43 |
| 128 | 79 | 82 | 70 | 83 |
| 256 | 156 | 163 | - * | 166 |

TABLE 3.2

**Overflow*

The accuracy of the four algorithms was also compared for various values of n. The maximum error of the solution vector (i.e. the maximum absolute difference between the randomly generated (known) components of the solution vector and their computed equivalents) was found; and by repeating the process for 60 different randomly generated right-hand side vectors, the average maximum error was obtained for various values of n. The results are summarised in Table (3.3).

Mean Maximum Errors in the Solution of a Periodic Tridiagonal System of Order n for 60-Right-Hand Sides
(Unit: $10^{-10}$)

| n | Gaussian Elimination Algorithm (3.1) Program (1) | Generalised Cyclic Factorisation Algorithm (3.2) Program (2) | Cyclic Reduction Algorithm (3.3) Program (3) | Rank-One Modification Algorithm (3.5) Program (5) |
|---|---|---|---|---|
| 16 | 1.7 | 1.6 | 1.8 | 2.6 |
| 32 | 2.1 | 1.8 | 2.6 | 4.1 |
| 64 | 2.4 | 2.0 | 3.0 | 5.9 |
| 128 | 2.6 | 2.2 | 3.7 | 7.3 |
| 256 | 2.9 | 2.5 | -* | 9.2 |

TABLE 3.3

**Overflow*

It can be seen that the generalised cyclic factorisation and the Gaussian elimination algorithms are about the most accurate and the rank-one modification method the least. All the algorithms are stable with respect to rounding errors under diagonal dominance condition of the coefficient matrix; though, as has been pointed out, the cyclic reduction algorithm has overflow problems for large values of n (see entry * in Tables 3.2 and 3.3).

For practical purposes, there is very little to choose between the algorithms. However, for periodic tridiagonal matrix problems of order 200 or higher with multiple right hand sides, the generalised cyclic factorisation method is highly recommended on the basis of a combination of speed, accuracy and stability. The complicated recursive structure of the cyclic reduction algorithm makes it the most difficult of the four algorithms to program.

## 3.4 THE SOLUTION OF CONSTANT TERM CYCLIC TRIDIAGONAL MATRIX SYSTEMS

### Algorithm (3.6)

Consider the matrix equation,

$$\underline{Cu} = \underline{d} \tag{3.4.1}$$

where C is a constant term, symmetric cyclic tridiagonal matrix of the form,

$$C = \begin{bmatrix} b & a & & & a \\ a & b & a & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & a \\ a & & & a & b \end{bmatrix} \tag{3.4.2}$$

We seek to obtain algorithms that take advantage of the special structure of the coefficient matrix (3.4.2). First, we consider simplified variants of the generalised cyclic factorisation method of algorithm (3.2).

Following the cyclic factorisation method in (3.2.12), the matrix C can be decomposed into the product of $\overline{P}$ and $\overline{Q}$ such that

$$C = \overline{P}\,\overline{Q} \tag{3.4.3}$$

where

$$\overline{P} = \begin{bmatrix} 1 & & & \ell \\ \ell & 1 & & \\ & \ddots & \ddots & \\ & & \ell & 1 \end{bmatrix},$$

and

$$\overline{Q} = \begin{bmatrix} v & a & & \\ & v & a & \\ & & \ddots & \ddots \\ a & & & v \end{bmatrix}.$$

From (3.4.3), the elements $\ell$,v are related in the following form,

$$\ell v = a$$
$$\text{and} \quad v + \ell a = b. \tag{3.4.4}$$

Now either $\ell$ or v can be expressed as an infinite periodic continued fraction. We consider v, which is then expressed in the form,

$$v = \frac{a^2}{b-}\ \frac{a^2}{b-}\ \frac{a^2}{b-}\ \frac{a^2}{b-}\ \frac{a^2}{b...} \qquad (3.4.5a)$$

with unit cycle length in the periodicity of the continued fraction.

By definition (2.12), the infinite continued fraction (3.4.5a) is generated by the linear fractional transformation,

$$\tau^{(1)}(\omega) = \frac{a^2}{b-\omega}\ , \qquad (3.4.5b)$$

whose fixed points $\omega_1,\omega_2$ are given by the roots of the quadratic equation,

$$\omega = \frac{a^2}{b-\omega}$$

or $\omega^2-b\omega+a^2=0$.

Hence, the value of v, defined as $\max(\omega_1,\omega_2)$ becomes,

$$\left.\begin{aligned} v &= (b+\sqrt{b^2-4a^2})/2\ , && b\geq 2a \\ \text{and}\quad \ell &= \frac{a}{v} = (b-\sqrt{b^2-4a^2})/2a, && b\geq 2a. \end{aligned}\right\} \qquad (3.4.6)$$

It follows immediately from (3.4.6) that if $b\geq 2a$, then $\ell\leq 1$.

The values of v and $\ell$ could simply have been obtained from (3.4.4) directly without the need to express any of them as an infinite continued fraction, but this has been done to maintain the uniformity of approach with the earlier strategy in the generalised factorisation of Algorithm (3.2).

By following the algorithmic method (3.2.17) to (3.2.26) but replacing $v_i,\ell_i,c_i$ by v,$\ell$ and a respectively; and by using the relation $\ell=a/v$, we obtain the following algorithmic solution of equation (3.4.1), which we shall refer to as the PQFACT1 algorithm. It is summarised below and given as program (6) of Appendix I.

PQFACT1 Algorithm (3.6)

Step 1 Compute: $\ell=(b-\sqrt{b^2-4a^2})/2a, \qquad b\geq 2a$ (3.4.7)

$\rho=\ell/a$.

Step 2: Compute the following pre-computed coefficients,

$$\left.\begin{aligned} \phi_i &= \ell^i, && \text{if } i \text{ is odd} \\ &= -\ell^i, && \text{if } i \text{ is even} \\ \gamma_i &= \phi_{n-i+1} \end{aligned}\right\} \quad i=1,2,\ldots n. \tag{3.4.8}$$

Step 3: Calculate the following:-

$$\text{(a)}\quad d_0' = 0,\ d_i' = d_i - \ell d_{i-1}', \qquad i=1,2,\ldots,n \tag{3.4.9a}$$

$$\left.\begin{aligned} \text{(b)}\quad y_n &= d_n'/(1+\phi_n), \\ y_i &= d_i' - \phi_i y_n, && i=2,3,\ldots n-1, \end{aligned}\right\} \tag{3.4.9b}$$

$$\left.\begin{aligned} \text{(c)}\quad g_n' &= \rho y_n, \\ g_i' &= \rho y_i - \ell g_{i+1}', && i=n-1,n-2,\ldots 1, \end{aligned}\right\} \tag{3.4.9c}$$

and

$$\left.\begin{aligned} \text{(d)}\quad u_1 &= g_i'/(1+\gamma_1), \\ u_i &= g_i' - \gamma_i u_1, && i=2,3,\ldots,n \end{aligned}\right\} \tag{3.4.9d}$$

where $\underline{u}=(u_1,u_2,\ldots,u_n)^T$ is the required solution vector.

Steps 1 and 2 are computed only once during the first solution; thereafter for subsequent solutions only step 3 need be repeated. Thus, the algorithm requires 5n multiplications and 4n additions, in addition to n pre-computed coefficients.

The stability of this algorithm is always guaranteed whenever $|\ell|<1$, (since $\ell$ is used as a multiplier in an elimination process) which is easily shown to be satisfied (using (3.4.7)) whenever the coefficient matrix is strictly diagonally dominant, i.e. $|b|>|2a|$.

If a=-1, then $\ell=(-b+\sqrt{b^2-4})/2$, b>2 and hence (3.4.9c) becomes,

$$\rho = -\ell$$

$$g_n' = -\ell y_n$$

and

$$g_i' = -\ell(y_i + g_{i+1}'), \qquad i=n-1,n-2,\ldots 1$$

which results in a saving of n extra multiplications to give the required arithmetic operation count as 4n multiplications and 4n additions for the solution of (3.4.1) when the coefficient matrix C is of the form,

$$C = \begin{bmatrix} b & -1 & & & -1 \\ -1 & b & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & -1 \\ -1 & & & -1 & b \end{bmatrix} .$$

Similarly, if a=1, then

$$\ell = (b-\sqrt{b^2-4})/2, \qquad b>2, \qquad \ell=\rho$$

and equation (3.4.9c) becomes,

$$g_n' = \ell y_n ,$$

and $$g_i' = \ell(y_i - g_{i+1}'), \qquad i=n-1,n-2,\ldots 1.$$

Algorithm (3.7)

When the coefficient matrix C is unsymmetric but has constant elements of the form,

$$C = \begin{bmatrix} b & c & & & a \\ a & b & c & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & c \\ c & & & a & b \end{bmatrix} \qquad (3.4.10)$$

then the solution of

$$C\underline{u} = \underline{d} \qquad (3.4.11)$$

can be similarly derived by a further variant of the PQFACT algorithm.

By applying the factorisation,

$$C = \hat{P}\,\hat{Q}$$

where

$$\hat{P} = \begin{bmatrix} 1 & & & \ell \\ \ell & 1 & 0 & \\ & \ddots & \ddots & \\ 0 & & \ell & 1 \end{bmatrix} .$$

and

$$\hat{Q} = \begin{bmatrix} v & c & & & \\ v & c & & 0 & \\ & \ddots & \ddots & & \\ & & & c & \\ 0 & & & v & \\ c & & & & \end{bmatrix}$$

we readily obtain the following relations,

$$\begin{aligned} \ell v &= a \\ v+\ell c &= b \end{aligned} \qquad (3.4.12)$$

from which we obtain,

$$\ell = \frac{b-\sqrt{b^2-4ac}}{2c} \quad \text{and} \quad v = \frac{b+\sqrt{b^2-4ac}}{2c}, \quad b^2 \geq 4ac$$

If $v_i, \ell_i, a_i, c_i$ in (3.2.17) to (3.2.26) are replaced by $v, \ell, a, c$ respectively, then the solution of (3.4.11) is obtained as a special case of the PQFACT algorithm. We denote this variant as the PQFACT2 algorithm and is summarised as follows:

PQFACT2 Algorithm

Step 1 Compute the following quantities,

$$\left.\begin{aligned} &\ell=(b-\sqrt{b^2-4ac})/2c \\ &\rho=\ell/a \quad \text{and} \quad \mu=c\rho \end{aligned}\right\} \qquad (3.4.13)$$

Step 2 Next, compute the following coefficients,

$$\left.\begin{aligned} \phi_i &= \ell^i, && i \text{ odd} \\ &= -\ell^i, && i \text{ even} \\ \gamma_i &= \sigma\mu^{n-i+1}, \end{aligned}\right\} \quad i=1,2,\ldots n \qquad (3.4.14)$$

where $\sigma=+1$ if n and i are both odd or both even; $\sigma=-1$, otherwise.

Step 3 Evaluate,

$$\text{(a)} \quad d'_0 = 0, \; d'_i = d_i - \ell d'_{i-1}, \; i=1,2,\ldots,n, \qquad (3.4.15a)$$

$$\text{(b)} \quad y_n = d'_n/(1+\phi_n),$$

$$y_i = d'_i - \phi_i y_n, \quad i=1,\ldots,n-1, \qquad (3.4.15b)$$

$$\text{(c)} \quad g'_n = \rho y_n,$$

$$g'_i = \rho y_i - \mu g'_{i+1}, \quad i=n-1,n-2,\ldots 1 \qquad (3.4.15c)$$

and finally,

$$\left.\begin{aligned} \text{(d)}\quad u_1 &= g_1'/(1+\gamma_1) \\ u_i &= g_i'-\gamma_i x_1\,, \qquad i=2\ldots n. \end{aligned}\right\} \tag{3.4.15d}$$

Again, this algorithm requires 5n multiplications, 4n additions, in addition to the pre-computed coefficients in steps 1 and 2.

Algorithm (3.8)

In a number of applications such as the solution of the first order periodic boundary transport (hyperbolic) equation and in the solution of heat-conduction (parabolic) p.d.e. with periodic boundary conditions by elliptic boundary-value techniques, it is often necessary to solve the matrix equation of the form,

$$C\underline{u} = \underline{d} \tag{3.4.16}$$

where C is a positive definite matrix of order n given by,

$$C = \begin{bmatrix} b & c & & & -a \\ -a & b & c & 0 & \\ & \ddots & \ddots & \ddots & \\ & & 0 & \ddots & c \\ c & & & -a & b \end{bmatrix} . \tag{3.4.17}$$

Then, we can apply the factorisation,

$$C = \tilde{P}\,\tilde{Q}$$

where $\tilde{P}$ and $\tilde{Q}$ are (n×n) cyclic matrices of the form,

$$\tilde{P} = \begin{bmatrix} 1 & & & -\ell \\ -\ell & 1 & 0 & \\ & \ddots & \ddots & \\ 0 & & -\ell & 1 \end{bmatrix} \qquad \text{and } \tilde{Q} = \begin{bmatrix} v & c & & 0 \\ & v & \ddots & \\ & & \ddots & c \\ c & 0 & & v \end{bmatrix}$$

The elements $\ell$ and u in the above factorisation process are obtained by multiplying $\tilde{P}$ and $\tilde{Q}$ together and equating to C to give the relationships,

$$\ell v = a$$
$$v - \ell c = b$$

from which we obtain, directly or by the periodic continued fraction expansion,

$$\ell = (-b+\sqrt{b^2+4ac})/2c \quad ,$$

and $$v = (b+\sqrt{b^2+4ac})/2 \, .$$

An algorithm to solve (3.4.16) can again be derived as a simplified variant of the PQFACT algorithm if $v_i, \ell_i$ and $c_i$ in (3.2.17) to (3.2.26) are replaced by $v, -\ell$ and $c$ respectively; and if we use the relation, $v=a/\ell$. The resulting algorithm denoted as the PQFACT3 algorithm is given below:

PQFACT3 Algorithm (3.8)

Step 1 Compute,

$$\left.\begin{aligned} \ell &= (-b+\sqrt{b^2+4ac})/2c \\ \rho &= 1/v = \ell/a \\ \mu &= c/v = c\rho \end{aligned}\right\} \qquad (3.4.18)$$

Step 2: Determine the following pre-computed coefficients,

$$\phi_i = (-1)^{i-1}(-\ell)^i = -\ell^i, \qquad i=1,2,\ldots n. \qquad (3.4.19a)$$

$$\left.\begin{aligned} \gamma_i &= (-1)^{i-1}\,\mu^{n-i+1} \quad (n \text{ odd}) \\ \gamma_i &= (-1)^{i}\,\mu^{n-i+1} \quad (n \text{ even}) \end{aligned}\right\} = \sigma\mu^{n-i+1}, \; i=1,2,\ldots n \qquad (3.4.19b)$$

where $\sigma=-1$ if n and i are both odd or both even

and $\sigma=1$ , otherwise.

Step 3: Compute:

$$\text{(a)} \quad d'_0 = 0, \quad d'_i = d_i+\ell d'_{i-1} \, , \quad i=1,2,\ldots n; \qquad (3.4.20a)$$

$$\left.\begin{aligned} \text{(b)} \quad y_n &= d'_n/(1+\phi_n) \, , \\ y_i &= d'_i-\phi_i y_n \, , \quad i=1,2,\ldots n \end{aligned}\right\} \qquad (3.4.20b)$$

$$\left.\begin{aligned} \text{(c)} \quad g'_n &= \rho y_n \, , \\ g'_i &= \rho y_n - \mu g'_{i+1} \, , \quad i=n-1,n-2,\ldots 1, \end{aligned}\right\} \qquad (3.4.20c)$$

$$\left.\begin{aligned} \text{(d)} \quad u_1 &= g'_1/(1+\gamma_1) \, , \\ \text{and} \quad u_i &= g'_i-\gamma_i x_1, \qquad i=2,3,\ldots n. \end{aligned}\right\} \qquad (3.4.20d)$$

This algorithm requires, apart from the initial 2n pre-computed coefficients only 5n multiplications and 4n additions.

Algorithm (3.9)

If the coefficient matrix C is skew-symmetric of the form,

$$C = \begin{bmatrix} b & 1 & & & -1 \\ -1 & b & & 0 & \\ & & \ddots & & \\ & 0 & & & 1 \\ 1 & & & -1 & b \end{bmatrix}_{(n\times n)} \qquad (3.4.21)$$

then a further simplification of Algorithm (3.8) and a reduction in the arithmetic operation count can be achieved by putting a=1, c=1 (and hence $\rho=\mu=\ell$) in the PQFACT3 algorithm. We denote the resulting algorithmic solution of the system (3.4.17) when the coefficient matrix C is of the form (3.4.21) as the PQFACT4 algorithm. It is summarised below and given as program (7) of Appendix I.

PQFACT 4 Algorithm

Step 1 Compute

$$\ell = \tfrac{1}{2}(-b+\sqrt{b^2+4}) \qquad (3.4.22)$$

Step 2 Compute

$$\phi_i = -\ell^i \quad , \quad i=1,2,\ldots,n, \qquad (3.4.23a)$$

$$\gamma_i = \sigma\phi_{n-i+1}, \quad i=1,2,\ldots,n \qquad (3.4.23b)$$

where $\sigma=-1$ if n and i are both even or both odd,

$=1$, otherwise.

Step 3 Compute

$$d_i' = d_i + \ell d_{i-1}' \qquad i=1,\ldots,n \qquad (3.4.24a)$$

$$d_0' \equiv 0$$

$$y_n = d_n'/(1+\phi_n) \ ,$$

$$y_i = d_i' - \phi_i y_n \quad , \quad i=1,2,\ldots,n-1, \qquad (3.4.24b)$$

$$g_n' = \ell y_n$$

$$g_i' = \ell(y_i - g_{i+1}'), \quad i=n-1,n-2,\ldots,1 \qquad (3.4.24c)$$

and finally,

$$u_1 = g_1'/(1+\gamma_1) \ ,$$

$$u_i = g_i'-\gamma_i x_1 \quad , \qquad i=2,3,\ldots,n. \tag{3.4.24d}$$

The arithmetic operation count is now 4 multiplications, 4 additions per unknown $u_i$, in addition to the n pre-computed coefficients.

Algorithm (3.10)

In Evans (1970) a neat algorithm was proposed which is based on first rewriting the given symmetric constant term matrix equation into a form whereby a unique Cholesky factorisation of the rewritten coefficient matrix is readily obtained.

We consider the matrix equation,

$$C\underline{u} = \underline{d} \quad , \tag{3.4.25}$$

or

$$\begin{bmatrix} b & a & & & a \\ a & b & a & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & a \\ a & & & a & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{3.4.26}$$

The system (3.4.25) can be rewritten as,

$$\tilde{C}\underline{u} = \underline{g} \quad , \tag{3.4.27}$$

or

$$\begin{bmatrix} 1+\ell^2 & -\ell & & & -\ell \\ -\ell & 1+\ell^2 & -\ell & & \\ & \ddots & \ddots & \ddots & \\ & & -\ell & 1+\ell^2 & -\ell \\ -\ell & & & -\ell & 1+\ell^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{3.4.28}$$

$$\left.\begin{aligned} &\text{where } \ell = \frac{-b\pm\sqrt{b^2-4a^2}}{2a} = \frac{-2a}{b+\sqrt{b^2-4a^2}} \ ; \ b>2a \\ &g_i = \mu^{-1} d_i \\ &\text{and} \quad \mu = b/(1+\ell^2) \end{aligned}\right\} \tag{3.4.29}$$

By a simple multiplication process, it can be easily shown that the matrix C has a unique Cholesky factorisation of the form,

$$C = \tilde{P}\,\tilde{P}^T \tag{3.4.30}$$

where

$$\tilde{P} = \begin{bmatrix} 1 & & & -\ell \\ -\ell & 1 & & \\ & \ddots & \ddots & \\ 0 & & -\ell & 1 \end{bmatrix}$$

is a cyclic matrix of order n.

Thus, the system (3.4.27) is now easily solved by rewriting it in the form,

$$\tilde{P}\,\tilde{P}^T\underline{u} = \underline{g}$$

which yields the two alternative systems,

$$\left.\begin{aligned} \tilde{P}\underline{y} &= \underline{g} \\ \text{and} \quad \tilde{P}^T\underline{u} &= \underline{y} \end{aligned}\right\} \tag{3.4.31}$$

where $\underline{y}$ is an auxiliary (n×1) vector.

By definition, b>2a. Hence, from (3.4.29), we readily find that

$$|\ell| < 1 .$$

It is therefore possible to use $\ell$ as a multiplier to perform simple stable Gaussian elimination processes on the systems (3.4.31) without the risk of accumulating rounding errors.

The result of solving (3.4.31) is the following algorithmic solution (see Evans (1970 & 1971)):

(1) Determine $y_1$ from

$$y_1 = (d_1+\ell^{n-1}d_2+\ldots\ell^3 d_{n-2}+\ell^2 d_{n-1}+\ell d_n)/(1-\ell^n)\mu \tag{3.4.32a}$$

(2) Compute the recursive sequence

$$y_i = \mu^{-1}d_i+\ell y_{i-1} \qquad 2\leq i\leq n \tag{3.4.32b}$$

(3) Determine $u_n$ from the formula,

$$u_n = (\ell y_1+\ell^2 y_2+\ldots\ell^{n-1}y_{n-1}+y_n)/(1-\ell^n) \tag{3.4.32c}$$

(4) Finally, obtain the following,

$$u_i = y_i + \ell u_{i+1}, \qquad i=n-1,n-2,\ldots,1. \tag{3.4.32d}$$

Using a nesting technique to compute (3.4.32a) and (3.4.32c) which involves $\ell$ as a multiplier, the algorithm (3.4.32a) to (3.4.32d) requires 5n multiplications, 4n additions in addition to 3 coefficients, $\mu$,$\ell$ and $(1-\ell^n)$, which need to be pre-computed.

Algorithm (3.11)

Finally, we outline another method proposed by Temperton (1975) for solving the (n×n) matrix equation,

$$C\underline{u} = \underline{d} \tag{3.4.32}$$

which in matrix form is,

$$\begin{bmatrix} b & 1 & & & 1 \\ 1 & b & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & 1 & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{3.4.33}$$

If it is assumed that $u_1$ is already known then (3.4.33) can be reduced to a strictly tridiagonal matrix system of dimension (n-1) of the form,

$$\begin{bmatrix} b & 1 & & \\ 1 & b & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & b \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_2-u_1 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n-u_1 \end{bmatrix} \tag{3.4.34}$$

which can be solved using the standard tridiagonal algorithm (see Algorithm 4.1) in 2(n-1) multiplications and 2(n-1) additions with (n-1) pre-computed coefficients.

Let $\underline{\omega} = (\omega_1,\omega_2,\ldots,\omega_n)$ be the first row of $C^{-1}$ (which exists since C is assumed to be non-singular), then, $u_1$ is given by the scalar product,

$$u_1 = \underline{\omega}.\underline{d}$$

or

$$u_1 = \sum_{i=1}^{n} \omega_i d_i \quad . \tag{3.4.35}$$

which can be evaluated in n multiplications, (n-1) additions. But since C is a symmetric circulant matrix, so is $C^{-1}$ and hence,

$$\omega_i = \omega_{n+2-i} \quad , \qquad 2 \leq i \leq n \ . \tag{3.4.36}$$

Thus we can rewrite the summation (3.3.35) as

$$u_1 = \omega_1 d_1 + \sum_{i=2}^{m} \omega_i (d_i + d_{n+2-i}) \quad \text{if n is odd,}$$

or

$$u_1 = \omega_1 d_1 + \sum_{i=2}^{m} \omega_i (d_i + d_{n+2-i}) + \omega_{m+1} d_{m+1} \quad \text{if n is even} \tag{3.4.37}$$

where m is the integer part of (n+1)/2. The total number of operations in evaluating $u_1$ is now reduced to approximately n/2 multiplications and n additions.

The major problem of this algorithm is the accurate determination of the vector $\underline{\omega}$. For a constant term matrix C, $\omega_i$ (i=1,...,n) is obtained by rewriting C in the form (3.4.28), from which it is shown (Temperton (1975)) that,

$$\omega_{i+1} = \sigma(\ell^i + \ell^{n-1}), \quad i=0,\ldots,m \tag{3.4.38}$$

where

$$\sigma = \frac{1}{2(\ell+\ell^{n-1})+b(1+\ell^n)}$$

and

$$\ell = \tfrac{1}{2}(-b \pm (b^2-4)^{\frac{1}{2}}) = \frac{2}{b+(b^2-4)^{\frac{1}{2}}} \ , \ b>2a.$$

If $u_1$ is evaluated using (3.4.37) instead of (3.4.35), then the total number of operations required by this algorithm is 5n/2 multiplications, 3n additions and approximately 3n/2 pre-computed coefficients. However, since this method requires the determination of the first row of the inverse of the coefficient matrix, the basic strategy of this algorithm cannot easily be extended to general cyclic matrix systems in which the coefficient matrix is neither symmetric nor circulant.

# CHAPTER 4

# SOME FAST ALGORITHMS ASSOCIATED WITH THE SOLUTION OF DIRICHLET'S AND NEUMANN'S BOUNDARY VALUE PROBLEMS

## 4.1 INTRODUCTION

In section (4.2), we outline two well-known algorithms for the fast solution of a tridiagonal and quindiagonal matrix system due to Thomas (1949) and Conte and Dames (1958) respectively. These algorithms are presented because they form the basis of our comparison with the new techniques developed in this chapter for solving the respective systems of equations.

A factorisation method, i.e., the reversed triangular factorisation and expansion (ReTriFE) (alternatively referred to as 'reversed rectangular factorisation') proposed by Evans (1972) for the solution of the constant element tridiagonal systems of equation is extended in section (4.3) by a generalisation which gives a direct fast solution of the more general tridiagonal matrix system. A further extension of the same idea is considered for the solution of the quindiagonal matrix system; and the special skew-symmetric constant element tridiagonal system which often arise, for example, in the solution of the one dimensional transport equation, or in parabolic p.d.e.'s solved by boundary value techniques (see section 5.4).

In section (4.4), another algorithmic approach, the recursive point partitioning (R.P.P.) idea, is introduced and developed to obtain fast solutions of tridiagonal, quindiagonal and other sparse banded systems associated with both Dirichlet's and Neumann's boundary conditions.

The inherent parallelism of the tridiagonal matrix solver algorithm resulting from the R.P.P. strategy, and the implementation of this in a program designed for execution on a two processor machine are discussed in section (4.5).

In section (4.6), we establish, by a rounding error analysis, the stability of the newly introduced recursive point partitioning (R.P.P.) algorithm.

Finally, by using a varying block-method strategy in a partitioning technique, a recursive point decoupling (R.P.D.) scheme, which possesses multiple parallelism, is formulated for the solution of a tridiagonal matrix equation.

## 4.2 THE ALGORITHMS OF THOMAS, AND CONTE AND DAMES FOR SOLVING TRIDIAGONAL AND QUINDIAGONAL MATRIX SYSTEMS

### Algorithm 4.1

We begin with the algorithm which was first presented by Thomas (1949) and has since appeared in many other references, such as Varga (1962) and Marchuk (1973). It is based on Gaussian elimination method without pivoting for size.

Thus, we consider the system,

$$A\underline{u} = \underline{d} \tag{4.2.1}$$

where A is an (n×n) tridiagonal matrix of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix}$$

The algorithmic solution is as follows (Varga (1962))

Define:

$$\left.\begin{aligned} \alpha_1 &= \frac{c_1}{b_1}, & \alpha_i &= \frac{c_i}{b_i - a_i\alpha_{i-1}}, \\ \text{and} \quad g_1 &= \frac{d_1}{b_1}; & g_i &= \frac{d_i - a_i g_{i-1}}{b_i - a_i\alpha_{i-1}}, \end{aligned}\right\} \quad i=2,3,\ldots,n. \tag{4.2.2}$$

Then, the components $u_i$ of the solution vector $\underline{u}$ are given recursively by,

$$u_n = g_n, \quad u_i = g_i - \alpha_i u_{i+1}, \quad i=n-1,n-2,\ldots,1. \tag{4.2.3}$$

The above algorithm requires 3n multiplications, 2n divisions and 3n additions. If the system (4.2.1) needs to be solved more than once with the coefficient matrix A unchanged, then the amount of work reduces to 2n multiplications, n divisions and 2n additions since the $\alpha_i$ (i=1,...n) terms need only be calculated once in such a situation.

The direct inversion method of (4.2.2)-(4.2.3) can be immediately generalised so as to apply to the block matrix equivalent of (4.2.1) where A is now a block tridiagonal matrix of order m given by,

$$A = \begin{bmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & C_{m-1} \\ 0 & & & A_m & B_m \end{bmatrix} \qquad (4.2.4)$$

where $A_i, B_i, C_i$ are square submatrices of order n.

Hence, if the vectors $\underline{u}$ and $\underline{d}$ are partitioned relative to the matrix A then the block form (Varga (1962)) is obtained as follows:

Algorithm 4.2

Define the matrices,

$$W_1 = B_1^{-1}C_1 \quad , \quad W_i = (B_i - A_i W_{i-1})^{-1} C_i \; ; \; i=2,3,\ldots,m, \qquad (4.2.5)$$

and

$$G_1 = B_1^{-1}\underline{d}_1 \quad , \quad G_i = (B_i - A_i W_{i-1})^{-1}(\underline{d}_i - A_i G_{i-1}), \; i=2,3,\ldots,m. \qquad (4.2.6)$$

Then, the components $\underline{u}_i$ of the solution vector are given recursively by,

$$\underline{u}_m = G_m \; ; \; \underline{u}_i = G_i - W_i \underline{u}_{i+1}, \; i=m-1,m-2,\ldots,1. \qquad (4.2.7)$$

The algorithm (4.2.5)-(4.2.7) is quite costly in terms of storage and arithmetic operation count because of the 2m matrix inversions involved, and the storage of each sub-matrix, $W_i$, i=1,2,...,m. It is therefore not computationally feasible to apply this direct block method for the general block tridiagonal matrix (4.2.4). However, in special cases, such as the solution of the Poisson equation with Dirichlet's or Neumann's boundary conditions in a rectangle, the sub-matrix blocks $A_i, B_i, C_i$ commute (i.e. possess a common set of orthogonal eigenvectors). In such situations, simplifications of the cumbersome general algorithm (4.2.5)-(4.2.7) are possible, resulting in substantial reductions in both the total computing effort and in coefficient storage. In Chapter 6 we shall develop new methods for the fast solution of such tridiagonal block matrix systems which possess commutative sub-matrix elements.

Algorithm 4.3

Next, we consider the system,

$$A\underline{u} = \underline{h} \tag{4.2.8}$$

where A is an (n×n) quindiagonal matrix of the form,

$$A = \begin{bmatrix} c_1 & d_1 & e_1 & & & & \\ b_2 & c_2 & d_2 & e_2 & & & \\ a_3 & b_3 & c_3 & d_3 & e_3 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & e_{n-2} \\ & & & \ddots & \ddots & \ddots & d_{n-1} \\ & & & & a_n & b_n & c_n \end{bmatrix} \tag{4.2.9}$$

The matrix in (4.2.9) can arise in place of the tridiagonal matrix, for example, when a 2-line ordering of the mesh points is adopted in the finite difference approximation of elliptic p.d.e.'s on a rectangular region, under Dirichlet's or Neumann's boundary conditions.

An algorithmic solution of (4.2.8) based on an elimination scheme proposed by Conte and Dames (1958) is given as follows:

Let the following quantities be expressed as,

$$\left.\begin{aligned} &w_1 = c_1 \;;\quad \beta_1 = d_1/w_1 \;,\quad \beta_0 = 0,\quad \beta_n = 0 \;, \\ &\gamma_1 = e_1/w_1,\ \gamma_0 = 0,\quad \gamma_n = \gamma_{n-1} = 0 \end{aligned}\right\} \tag{4.2.10}$$

then, we define recursively,

$$\left.\begin{aligned} \delta_i &= b_i - a_i\beta_{i-2} \;, \\ w_i &= c_i - a_i\gamma_{i-1} - \delta_i\beta_{i-1} \;, \\ \beta_i &= (d_i - \delta_i\gamma_{i-1})/w_i \;, \qquad i = 2,3,\ldots,n-1 \;, \\ \text{and}\quad \gamma_i &= e_i/w_i \;, \qquad i = 2,3,\ldots n-2. \end{aligned}\right\} \quad i = 2,3,\ldots,n, \tag{4.2.11}$$

Next, we form,

$$\left.\begin{aligned} &g_0 = 0,\quad g_1 = h_1/w_1 \;, \\ &g_i = (h_i - a_i g_{i-2} - \delta_i g_{i-1})/w_i,\quad i=2,3,\ldots,n. \end{aligned}\right\} \tag{4.2.12}$$

Finally, the solution vector $\underline{u}$ is obtained successively from the formulae,

$$u_{n+1} = 0, \quad u_n = g_n$$
$$u_i = g_i - \beta_i u_{i+1} - \gamma_i u_{i+2}, \quad i=n-1,n-2,\ldots,1. \tag{4.2.13}$$

This algorithm (4.2.10)-(4.2.13) requires 8n multiplications, 3n divisions and 8n additions; and a further 4n multiplications, n divisions and 4n additions for subsequent applications if the coefficient matrix A remains unchanged.

## 4.3 FAST ALGORITHMS BASED ON THE REVERSED TRIANGULAR FACTORISATION AND EXPANSION (ReTriFE) METHOD

### Algorithm 4.4

First, we outline an algorithm given by Evans (1972) for the solution of the constant term symmetric tridiagonal matrix equation based on the rectangular (reversed triangular) factorisation and expansion of the transformed coefficient matrix.

Consider the constant term, diagonally dominant, symmetric tridiagonal matrix equation,

$$A\underline{u} = \underline{d} \tag{4.3.1a}$$

which may be written in matrix form as,

$$\begin{bmatrix} b & a & & & 0 \\ a & b & a & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & a \\ 0 & & & a & b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}, \quad b \geq 2a. \tag{4.3.1b}$$

The system (4.3.1a) can be rewritten as,

$$\tilde{A}\underline{u} = \underline{g} \tag{4.3.2a}$$

i.e.,

$$\begin{bmatrix} 1+\ell^2 & -\ell & & & 0 \\ -\ell & 1+\ell^2 & -\ell & & \\ & \ddots & \ddots & \ddots & \\ & & -\ell & 1+\ell^2 & -\ell \\ 0 & & & -\ell & 1+\ell^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{4.3.2b}$$

where the following transformation has been made:

$$\left.\begin{aligned} \frac{a}{b} &= \frac{-\ell}{1+\ell^2}, \quad \text{i.e.,} \quad \ell = \frac{-2a}{b+\sqrt{b^2-4a^2}} \\ \text{and} \quad g_i &= (1+\ell^2)\frac{d_i}{b}, \quad i=1,2,\ldots,n. \end{aligned}\right\} \tag{4.3.3}$$

It is easy to verify by direct multiplication that the matrix $\tilde{A}$ possesses a unique rectangular factorisation of the form,

$$\tilde{A} = QQ^T$$

where Q is the (n×n+1) rectangular matrix given by,

$$Q = \begin{bmatrix} 1 & -\ell & & & \\ & 1 & -\ell & & 0 \\ & & \ddots & \ddots & \\ 0 & & & 1 & -\ell \\ & & & & 1 \quad -\ell \end{bmatrix} \qquad (4.3.5)$$

Thus, the system (4.3.2b) is easily solved by rewriting it in the form,

$$QQ^T\underline{u} = \underline{g} \qquad (4.3.6)$$

which, with the introduction of an auxiliary (n+1)-component vector $\underline{v}$, yields the two triangular systems,

$$Q\underline{v} = \underline{g} \qquad (4.3.7a)$$

and

$$Q^T\underline{u} = \underline{v} \quad , \qquad (4.3.7b)$$

which have to be solved to obtain the solution vector $\underline{u}$.

Derivation of Algorithmic Solution

By definition, $b \geq 2a$, hence from (4.3.3) we have

$$|\ell| \leq 1 \ .$$

It is therefore sound numerically to use $\ell$ as a multiplier in an elimination process without the risk of rounding error accumulation.

Thus, in (4.3.7a), by multiplying the second equation by $\ell$, the third equation by $\ell^2$,..., the $i^{th}$ equation by $\ell^{i-1}$ and finally the $n^{th}$ equation by $\ell^{n-1}$, the following system of equations is obtained,

$$\begin{bmatrix} 1 & -\ell & & & \\ & \ell & -\ell^2 & & 0 \\ & & \ell^2 & -\ell^3 & \\ & & & \ddots & \\ 0 & & & -\ell^{n-1} & \\ & & & \ell^{n-1} & -\ell^n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} g_1 \\ \ell g_2 \\ \vdots \\ \ell^{n-1} g_n \end{bmatrix} . \qquad (4.3.8)$$

If we add together all the n equations of (4.3.8) we obtain an expression, relating $v_1$ with $v_{n+1}$ in terms of $g_i$, and given by,

$$v_1 - \ell^n v_{n+1} = g_1 + \ell g_2 + \ldots + \ell^{n-1} g_n \,. \tag{4.3.9a}$$

By following a similar reduction process applied in the same manner but commencing on the second, third,... and finally $n^{th}$ equation we obtain the following sequence of equations,

$$\left.\begin{aligned} v_2 - \ell^{n-1} v_{n+1} &= g_2 + \ell g_3 + \ldots + \ell^{n-2} g_n \,, \\ v_3 - \ell^{n-2} v_{n+1} &= g_3 + \ell g_4 + \ldots + \ell^{n-3} g_n \,, \\ \ldots\ldots & \qquad \ldots\ldots\ldots\ldots\ldots \\ v_n - \ell v_{n+1} &= g_n \end{aligned}\right\} \tag{4.3.9b}$$

relating each of the quantities $v_2, v_3, \ldots v_n$ with $v_{n+1}$ in terms of $g_i$.

Next, we consider the system (4.3.7b) and by a similar elimination process, we obtain immediately the expressions for $u_1, u_2, \ldots u_n$ in the form,

$$\left.\begin{aligned} u_1 &= v_1 \,, \\ u_2 &= v_2 + \ell v_1 \,, \\ \ldots & \qquad \ldots\ldots\ldots\ldots \\ u_i &= v_i + \ell v_{i-1} + \ldots + \ell^{i-1} v_1 \,, \\ \ldots & \qquad \ldots\ldots\ldots\ldots \\ u_n &= v_n + \ell v_{n-1} + \ldots + \ell^{n-1} v_1 \,. \end{aligned}\right\} \tag{4.3.10}$$

Substitution of $v_n, v_{n-1} \ldots, v_2$ and $v_1$ from (4.3.9) in terms of $v_{n+1}$ into the last equation of (4.3.10) together with the connecting equation,

$$-\ell u_n = v_{n+1} \tag{4.3.11}$$

yields the final expression for $u_n$ as,

$$\begin{aligned} (1+\ell^2+\ell^4+\ldots\ell^{2n})u_n = {} & g_n(1+\ell^2+\ell^4+\ldots+\ell^{2(n-1)}) + \ell g_{n-1}(1+\ell^2+\ell^4+\ldots+\ell^{2(n-2)}) \\ & + \ldots + \ell^i g_{n-i}(1+\ell^2+\ell^4+\ldots+\ell^{2(n-i+1)}) \\ & + \ldots + \ell^{n-1} g_1 \end{aligned} \tag{4.3.12}$$

which simplifies, for $|\ell|<1$, to give,

$$\begin{aligned} u_n = {} & (g_n + \ell g_{n-1} + \ldots \ell^i g_{n-i} + \ldots \ell^{n-1} g_1 - \ell^{n+1} g_1 - \ell^{n+2} g_2 \\ & - \ldots - \ell^{2n} g_n)/(1-\ell^{2n+2}) \end{aligned} \tag{4.3.13}$$

together with $v_{n+1} = -\ell u_n$. (4.3.14a)

A backsubstitution process, using (4.3.7a) yields the components of the auxiliary solution vector $\underline{v}$ as,

$$v_i = g_i + \ell v_{i+1} \,, \quad i=n,n-1,\ldots,1. \tag{4.3.14b}$$

And finally, the solution vector $\underline{u}$ is obtained from a forward substitution process using (4.3.7b) to give,

$$\begin{aligned} u_1 &= v_1 \quad , \\ u_i &= v_i + \ell u_{i-1} \quad , \quad i=2,3,\ldots,n. \end{aligned} \tag{4.3.15}$$

Thus, the solution of matrix equation (4.3.1) is obtained by the algorithm (4.3.13)-(4.3.15) which requires

$5n$ multiplications

and $4n$ additions

provided that the expression in (4.3.13) is evaluated using a nesting technique. Subsequent applications require $4n$ multiplications and $4n$ additions, if the coefficient matrix is unchanged.

Special Case 1

When $b=2a$, then from (4.3.3) $\ell=-1$ and equation (4.3.13) is indeterminate but $u_n$ can now be obtained from the alternative expression (4.3.12) to give,

$$u_n = [ng_n-(n-1)g_{n-1}+(n-2)g_{n-2}-\ldots+(-1)^{n-1}g_1]/(n+1) \tag{4.3.16a}$$

together with

$$v_{n+1} = u_n, \quad v_i = g_i - v_{i+1}, \quad i=n,n-1,\ldots,1., \tag{4.3.16b}$$

and finally,

$$u_1 = v_1 \ , \ u_i = v_i - u_{i-1}, \quad i=2,3,\ldots,n-1. \tag{4.3.16c}$$

Special Case 2

When $b=-2a$, then $\ell=1$ and hence we have,

$$u_n = [ng_n+(n-1)g_{n-1}+(n-2)g_{n-2}+\ldots g_1]/(n+1) \tag{4.3.17a}$$

together with

$$\begin{aligned} v_{n+1} &= -u_n, \\ v_i &= g_i + v_{i+1}, \quad i=n,n-1,\ldots 1, \end{aligned} \tag{4.3.17b}$$

and finally,

$$\begin{aligned} u_1 &= v_1 \ , \\ u_i &= v_i + v_{i-1} \ , \quad i=2,3,\ldots,n-1. \end{aligned} \tag{4.3.17c}$$

These two special cases considered above require only $n$ multiplications and $3n$ additions which is a remarkably special fast method of solution of (4.3.1) when $b=\pm 2a$.

Algorithm 4.5

The method of algorithm (4.4) can be extended to obtain the solution of the special (n×n) symmetric tridiagonal matrix equation,

$$A\underline{u} = \underline{d} \tag{4.3.18}$$

where A is now of the form,

$$\begin{bmatrix} b_1 & c_1 & & & \\ c_1 & b_2 & c_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & \\ 0 & & c_{n-1} & b_n & \end{bmatrix}, \quad b_i \geq 2c_i, \quad b_i = c_i^2 + 1 \tag{4.3.19}$$

An easily factorable transformation of the system (4.3.18) is achieved by considering a new system (4.3.21), on multiplying (4.3.18) by $D$ and (4.3.21) by $\hat{D}$ where $D = \text{diag}(b_1^{-1}, \ldots b_n^{-1})$ and $\hat{D} = \text{diag}[(1+\ell_1^2)^{-1}, \ldots, (1+\ell_n^2)^{-1}]$ and comparing the elements of the resulting non-symmetric systems, we have,

$$\begin{aligned} c_i/b_i &= -\ell_i/(1+\ell_i^2), \quad i=1,2,\ldots,n-1 \\ \text{and} \quad \frac{c_{n-1}}{b_n} &= \frac{-\ell_{n-1}}{1+\ell_n^2} \end{aligned} \tag{4.3.20}$$

where the new system is

$$\hat{A}\underline{u} = \underline{g} \tag{4.3.21}$$

which, in matrix form, is given by,

$$\begin{bmatrix} 1+\ell_1^2 & -\ell_1 & & & \\ -\ell_1 & 1+\ell_2^2 & -\ell_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & -\ell_{n-1} & \\ 0 & & -\ell_{n-1} & 1+\ell_n^2 & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}$$

From (4.3.20),

$$\ell_i = \frac{-2c_i}{b_i + \sqrt{b_i^2 - 4c_i^2}}, \quad b_i \geq 2c_i, \quad i=1,2,\ldots,n-1, \tag{4.3.22a}$$

$$\ell_n = \sqrt{\frac{b_n \ell_{n-1}^{-1}}{c_{n-1}}}, \quad b_n > c_{n-1} \ . \tag{4.3.22b}$$

and

$$g_i = \frac{(1+\ell_i^2)}{b_i} d_i, \quad i=1,2,\ldots,n. \tag{4.3.22c}$$

The transformed matrix $\hat{A}$ can be easily verified to possess a unique rectangular factorisation of the form,

$$\hat{A} = \hat{P}\hat{P}^T \tag{4.3.23}$$

where $\hat{P}$ is the $n\times(n+1)$ rectangular matrix given by,

$$\hat{P} = \begin{bmatrix} 1 & -\ell_1 & & & & \\ & 1 & -\ell_2 & & 0 & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & 0 & & & -\ell_{n-1} & \\ & & & & 1 & -\ell_n \end{bmatrix}_{n\times(n+1)}$$

The solution of (4.3.21) is then obtained more easily by rewriting it in the form,

$$\hat{P}\hat{P}^T\underline{u} = \underline{g} \tag{4.3.24}$$

which yields the two systems,

$$P\underline{y} = \underline{g} \tag{4.3.25a}$$

and

$$P^T\underline{u} = \underline{y} \tag{4.3.25b}$$

where $\underline{y}=(y_1,y_2,\ldots,y_{n+1})^T$ is an intermediate vector.

The factorisation scheme adopted above is valid for each $b_i^2 \geq 4c_i^2$, $i=1,2,\ldots,n-1$ and hence by using (4.3.22a) we immediately have $|\ell_i|\leq 1$ $(i=1,2,\ldots,n-1)$. It is thus possible to use these $\ell_i$ terms as multipliers in an elimination process without the risk of rounding error growth.

Thus, by using the $\ell_i$ $(i=1,\ldots,n-1)$ as multipliers in an elimination procedure identical to that adopted in the analysis of algorithm (4.4), the linear systems (4.3.25) can be shown to give the following expression for $u_n$:

$$\begin{aligned}
&u_n(1+\ell_n^2+\ell_n^2\ell_{n-1}^2+\ell_n^2\ell_{n-1}^2\ell_{n-2}^2+\ldots\ell_n^2\ell_{n-1}^2\ldots\ell_1^2] = \\
&g_n[1+\ell_{n-1}^2+\ell_{n-1}^2\ell_{n-2}^2+\ell_{n-1}^2\ell_{n-2}^2\ell_{n-3}^2+\ldots+\ell_{n-1}^2\ell_{n-2}^2\ldots\ell_1^2] + \\
&+\ell_{n-1}g_{n-1}[1+\ell_{n-2}^2+\ell_{n-2}^2\ell_{n-3}^2+\ell_{n-2}^2\ell_{n-3}^2\ell_{n-4}^2+\ldots+\ell_{n-2}^2\ell_{n-3}^2\ldots\ell_1^2] + \\
&+\ell_{n-1}\ell_{n-2}g_{n-2}[1+\ell_{n-3}^2+\ell_{n-3}^2\ell_{n-4}^2+\ell_{n-3}^2\ell_{n-4}^2\ell_{n-5}^2+\ldots+\ell_{n-3}^2\ell_{n-4}^2\ldots\ell_1^2] + \\
&+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots+ \\
&\ell_{n-1}\ell_{n-2}\ldots\ell_{n-i}g_{i-1}[1+\ell_{n-i-1}^2+\ell_{n-i+1}^2\ell_{n-i-2}^2+\ldots\ell_{n-i-1}^2\ell_{n-i-2}^2\ldots\ell_1^2] + \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots+ \\
&(\ell_{n-1}\ell_{n-2}\ldots\ell_2)g_2[1+\ell_1^2]+(\ell_{n-1}\ell_{n-2}\ldots\ell_1)g_1
\end{aligned} \tag{4.3.26}$$

together with the connecting equation,

$$-\ell_n u_n = y_{n+1} \,. \tag{4.3.27}$$

By defining the following recurrence relation,

$$\begin{aligned} s_0 &= 1, \\ s_i &= (1+\ell_i^2+\ell_i^2\ell_{i-1}^2+\ell_i^2\ell_{i-1}^2\ell_{i-2}^2+\ldots+\ell_i^2\ell_{i-1}^2\ldots\ell_1^2) \,, \\ &\qquad\qquad i=1,2,\ldots,n \,, \end{aligned} \tag{4.3.28}$$

and
$$t_i = \ell_i\ell_{i+1}\ell_{i+2}\ldots\ell_{n-1} \,, \quad i=1,2,\ldots,n-1, \tag{4.3.29}$$
$$t_n = 1 \,,$$

a more computationally convenient expression for (4.3.26) becomes,

$$\begin{aligned} s_n u_n = {} & s_{n-1}g_n+s_{n-2}g_{n-1}t_{n-1}+s_{n-3}g_{n-2}t_{n-2}+\ldots \\ & +s_1g_2t_2+s_0g_1t_1 \,. \end{aligned} \tag{4.3.30}$$

Further, from the recurrence relations (4.3.28) and (4.3.29) we immediately obtain,

$$s_i = \ell_i^2 s_{i-1} + 1$$

and
$$t_{i+1} = t_i/\ell_i$$

and hence the following scheme gives a neat computational method for evaluating $u_n$:

$$\left.\begin{aligned} t_1 &= \ell_1\ell_2\ldots\ell_{n-1} \,, \quad T_0 = 0, \; s_0 = 1 \\ T_i &= T_{i-1}+g_i t_i s_{i-1} \\ t_{i+1} &= t_i/\ell_i \\ s_i &= \ell_i^2 s_{i-1}+1 \end{aligned}\right\} \quad i=1,2,\ldots,n \tag{4.3.31}$$

and hence, $u_n = T_n/s_n$.

The back substitution process using (4.3.25a) yields the components of the auxiliary solution vector, i.e.,

$$y_{n+1} = -\ell_n u_n$$

and
$$y_i = g_i+\ell_i y_{i+1}, \quad i=n,n-1,\ldots 1 \tag{4.3.32}$$

whilst the components of the solution vector $\underline{u}$ are given by

$$u_1 = y_1$$

and finally a forward substitution of the form,

$$u_i = y_i+\ell_{i-1}u_{i-1}, \quad i=2,3,\ldots,n-1 \,. \tag{4.3.33}$$

Algorithm (4.5) which we denote as the Generalised ReTriFE (GENRET) algorithm is summarised below and given as Program 8 of Appendix I.

Generalised ReTriFE Algorithm (4.5)

Step 1 First, we compute the following coefficients,

$$\ell_i = \frac{-2c_i}{b_i+(b_i^2-4c_i^2)^{\frac{1}{2}}} \quad , \; b_i^2>4c_i^2 \; , \quad i=1,2,\ldots,n-1,$$

$$\ell_n = \left[\frac{-b_n\ell_{n-1}-c_{n-1}}{c_{n-1}}\right]^{\frac{1}{2}} \quad , \; b_n>c_{n-1} \; ,$$

$$t_1 = \prod_{j=1}^{n-1}\ell_j \; , \quad t_{i+1} = t_i/\ell_i, \quad i=1,2,\ldots,n-1$$

and $T_0 = 0, \; s_0 = 1 .$

Step 2 Next, we compute,

$$\left.\begin{aligned} g_i &= \frac{d_i}{b_i}(1+\ell_i^2) \; , \\ T_i &= T_{i-1}+s_{i-1}g_it_i \; , \\ s_i &= \ell_i^2 s_{i-1}+1 \end{aligned}\right\} \quad i=1,2,\ldots,n \; ,$$

and then $u_n = T_n/s_n \; .$

Step 3 Finally, we obtain,

$$y_{n+1} = -\ell_n u_n \; ,$$

$$y_i = g_i+\ell_i y_{i+1}, \quad i=n,n-1,\ldots 1,$$

and $$u_1 = y_1, \; u_i = y_i+\ell_{i-1}u_{i-1}, \; i=2,3,\ldots,n.$$

Altogether, this algorithm requires approximately 6n multiplications, and 4n additions, in addition to the pre-computed coefficients $\ell_i, t_i$ which need to be calculated just once if more than one solution is required.

Algorithm 4.6

Next we adopt the approach of the last two algorithms to obtain the solution of the constant term symmetric quindiagonal matrix equation,

$$A\underline{u} = \underline{h} \; . \tag{4.3.34}$$

which in matrix form may be written as,

$$\begin{bmatrix} d & e & f & & & & \\ e & d & e & f & & 0 & \\ f & e & d & e & f & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & f \\ & & & & & & e \\ & 0 & & & & & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} \tag{4.3.35}$$

where d,e,f are assumed to be non-zero.

If each equation of (4.3.35) is multiplied by $\frac{1+\alpha^2+\beta^2}{d}$, and by using the simple transformations,

$$\frac{e}{d} = \frac{-\alpha+\alpha\beta}{1+\alpha^2+\beta^2} \tag{4.3.36a}$$

and

$$\frac{f}{d} = \frac{-\beta}{1+\alpha^2+\beta^2} \tag{4.3.36b}$$

the system (4.3.34) is transformed to the new form,

$$\hat{A}\underline{u} = \underline{g} \tag{4.3.37}$$

which in matrix form is,

$$\begin{bmatrix} 1+\alpha^2+\beta^2 & -\alpha+\alpha\beta & -\beta & & & & \\ -\alpha+\alpha\beta & 1+\alpha^2+\beta^2 & -\alpha+\alpha\beta & -\beta & & 0 & \\ -\beta & -\alpha+\alpha\beta & 1+\alpha^2+\beta^2 & -\alpha+\alpha\beta & -\beta & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & -\beta \\ & & & & & & -\alpha+\alpha\beta \\ & 0 & & & & & 1+\alpha^2+\beta^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix} \tag{4.3.38}$$

where $g_i = (1+\alpha^2+\beta^2)\frac{h_i}{d}$, $\quad i=1,2,\ldots,n,$ (4.3.39)

and from (4.3.36a) and (4.3.36b),

$$1+\alpha^2+\beta^2 = -\alpha(1-\beta)\frac{d}{e} \quad , \tag{4.3.40a}$$

and

$$1+\alpha^2+\beta^2 = -\beta\, d/f \ . \tag{4.3.40b}$$

Hence we have,

$$\alpha = \frac{\beta e}{(1-\beta)f} \ , \tag{4.3.41a}$$

or

$$\beta = \frac{f\alpha}{e+f\alpha} \ . \tag{4.3.41b}$$

A substitution for β in (4.3.4oa) gives the following quartic equation in terms of α, i.e.,

$$f^2\alpha^4+2ef\alpha^3+(2f^2+e^2+df)\alpha^2+(2ef+de)\alpha+e^2 = 0. \qquad (4.3.42)$$

Similarly we obtain,

$$\beta^4+(\frac{d}{f}-2)\beta^3+(2-2\frac{d}{f}+\frac{e^2}{f^2})\beta^2+(\frac{d}{f}-2)\beta+1 = 0 \qquad (4.3.43)$$

In the algorithmic solution to follow, $\alpha$ and $\beta$ are to be used in the usual manner as multipliers in an elimination process and hence for stability reasons we must choose values of $\alpha$ and $\beta$ (amongst the possible values they can take) such that

$$|\alpha|<1, \quad \text{and} \quad |\beta|<1 .$$

In order to determine the values of $\alpha$ and $\beta$, we choose to solve for $\alpha$ from equation (4.3.42) by using the Newton-Raphson iterative method which obtains the root of the polynomial equation $\phi(\alpha)=0$, by the formula,

$$\alpha^{(k+1)} = \alpha^{(k)} - \frac{\phi(\alpha^{(k)})}{\phi'(\alpha^{(k)})}, \quad \phi'(\alpha^{(k)}) \neq 0 \qquad (4.3.44)$$

at the $(k+1)^{th}$ iteration step.

Since we require $|\alpha|<1$, then $\alpha^{(0)}=0$ is a good starting value and the convergence of the iterative scheme is attained when

$$|\alpha^{(k+1)}-\alpha^{(k)}|<\varepsilon \quad , \quad k=0,1,2,...$$

for a given small number, $\varepsilon$(e.g. $5\times10^{-10}$).

If we put $\alpha^{(k)}=\xi+k\varepsilon$ where $\phi(\xi)=0$, $\phi'(\xi)\neq0$ then it can be shown (Froberg (1965),p.21) that

$$\varepsilon_{k+1} = \frac{\phi''(\xi)}{2\phi'(\xi)}\varepsilon_k^2 ,$$

which implies that the iterative scheme (4.3.44) has a quadratic convergence. The number of correct decimals is thus approximately doubled at every iteration, (if the factor $\phi''(\xi)/2\phi'(\xi)$ is not too large) and this perhaps explains why it takes on the average 5 iterations for $\alpha$ to converge from

$$\alpha^{(0)} = 0 \text{ to } \alpha^{(k)} = \xi\pm\varepsilon \ (\xi<1, \text{ being an exact root}).$$

Next, we now proceed with the algorithmic solution under the assumption that $|\alpha|$ and $|\beta|<1$ have been determined.

Algorithmic Solution

The matrix $\hat{A}$ can be shown by direct multiplication to possess a unique rectangular factorisation given by,

$$\hat{A} = RR^T \quad , \tag{4.3.45}$$

where R is the $(n\times(n+2))$ upper rectangular matrix of the form,

$$R=\begin{bmatrix} 1 & -\alpha & -\beta & & & & \\ & 1 & -\alpha & -\beta & & 0 & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & -\beta & \\ & 0 & & & 1 & -\alpha & -\beta \\ & & & & & 1 & -\alpha & -\beta \end{bmatrix}_{(n)\times(n+2)} \tag{4.3.46}$$

The system (4.3.37) can now be solved in two stages, by introducing the $(n+2)\times 1$ auxiliary vector $\underline{y}$ to give,

$$R\underline{y} = \underline{g} \tag{4.3.47a}$$

and

$$R^T\underline{u} = \underline{y} \ . \tag{4.3.47b}$$

Thus, if we now consider the system (4.3.47a), which, in matrix form, is given by

$$\begin{bmatrix} 1 & -\alpha & -\beta & & & & \\ & 1 & -\alpha & -\beta & & 0 & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & -\beta & \\ & 0 & & & 1 & -\alpha & -\beta \\ & & & & & 1 & -\alpha & -\beta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \\ y_{n+2} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{4.3.48}$$

then, we can obtain $y_i$, $(i=1,2,\ldots,n)$ in terms of $y_{n+1}, y_{n+2}$. To do this, we apply a recurrence backward substitution process to the system (4.3.48). The $n^{th}$ equation of (4.3.48) is given by,

$$y_n = g_n + \alpha y_{n+1} + \beta y_{n+2}$$

which we can express as

$$y_n = b_1 + a_2 y_{n+1} + a_1 \beta y_{n+2} \ , \tag{4.3.49}$$

where $b_1=g_n$, $a_1=1$, and $a_2=\alpha$.

Similarly, the $(n-1)^{th}$ equation of (4.3.48) is given by

$$y_{n-1} = g_{n-1} + \alpha y_n + \beta y_{n+1}$$

which, when expressed in terms of $y_{n+1}$ and $y_{n+2}$ by substituting for $y_n$ from (4.3.49), yields,

$$y_{n-1} = b_2 + a_3 y_{n+1} + a_2 \beta y_{n+2} \tag{4.3.50}$$

where,

$$b_2 = g_{n-1} + \alpha g_n = g_{n-1} + \alpha b_1$$

$$a_2 = \alpha$$

and $$a_3 = \alpha^2 + \beta = \alpha a_2 + \beta a_1 \ .$$

By a similar process, we obtain in general, the expression for $y_{n+1-j}$ $(j=1,2,\ldots,n)$ in terms of the two unknowns, $y_{n+1}$ and $y_{n+2}$ as follows:

$$y_{n+1-j} = b_j + a_{j+1} y_{n+1} + a_j \beta y_{n+2} \ , \tag{4.3.51}$$

where $a_j, b_j$ are given by the recurrence relation,

$$\left.\begin{aligned} a_1 &= 1 \ , \\ a_2 &= \alpha \ , \\ a_j &= \alpha a_{j-1} + \beta a_{j-2}, \quad j=3,4,\ldots,n+1, \end{aligned}\right\} \tag{4.3.52a}$$

and

$$\left.\begin{aligned} b_1 &= g_n \ , \\ b_2 &= g_{n-1} + \alpha b_1 \ , \\ b_j &= g_{n-j+1} + \alpha b_{j-1} + \beta b_{j-2}, \quad j=3,\ldots,n. \end{aligned}\right\} \tag{4.3.52b}$$

In matrix notation, (4.3.51) represents the result of a backward elimination process on the system (4.3.48) to give,

$$\begin{bmatrix} 1 & & & -a_{n+1} & -\beta a_n \\ & 1 & 0 & \vdots & \vdots \\ & & \ddots & & \\ & 0 & & -a_3 & -\beta a_2 \\ & & & -a_2 & -\beta a_1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \\ y_{n+2} \end{bmatrix} = \begin{bmatrix} b_n \\ b_{n-1} \\ \vdots \\ b_2 \\ b_1 \end{bmatrix} . \tag{4.3.53}$$

Further, from the system (4.3.47b) which, in matrix form, is given by,

$$\begin{bmatrix} 1 & & & & \\ -\alpha & 1 & & & \\ -\beta & -\alpha & 1 & & \\ & \ddots & \ddots & \ddots & \\ & 0 & & -\alpha & 1 \\ & & & -\beta & -\alpha \\ & & & & -\beta \end{bmatrix}_{(n+2)\times n} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \\ y_{n+2} \end{bmatrix} \tag{4.5.54}$$

we obtain,

$$u_1 = y_1 \quad ,$$

$$u_2 = y_2+\alpha u_1 = y_2+\alpha y_1 = a_1y_2+a_2y_1 \quad .$$

Similarly, $u_3 = y_3+\alpha u_2+\beta u_1$ ,

which on substituting for $u_1$ and $u_2$ gives,

$$u_3 = a_1y_3+a_2y_2+a_3y_1$$

where $a_1,a_2,a_3$ are given by (4.3.52a).

Generally, by a successive forward substitution process we have,

$$u_i = \sum_{j=1}^{i} a_j y_{i-j+1} \quad , \qquad i=1,2,\ldots,n. \tag{4.3.55}$$

We now proceed to solve for the two unknown values $y_{n+1}$ and $y_{n+2}$.

From (4.3.55), we have

$$u_{n-1} = \sum_{j=1}^{n-1} a_j y_{n-j} \tag{4.3.56a}$$

and

$$u_n = \sum_{j=1}^{n} a_j y_{n+1-j} \quad . \tag{4.3.56b}$$

Hence, substituting for $y_{n-k}$ and $y_{n+1-k}$ from (4.3.51) into (4.3.56a) and (4.3.56b) respectively, we obtain,

$$u_{n-1} = \sum_{j=1}^{n-1} a_j (b_{j+1}+a_{j+2}y_{n+1}+\beta a_{j+1}y_{n+2}) \tag{4.3.57a}$$

and

$$u_n = \sum_{j=1}^{n} a_j (b_j+a_{j+1}y_{n+1}+\beta a_j y_{n+2}) \quad . \tag{4.3.57b}$$

Further, the two connecting equations are obtained from the system (4.3.54) to give,

$$-\beta u_n = y_{n+2} \quad , \tag{4.3.58}$$

and $$-\beta u_{n-1}-\alpha u_n = y_{n+1}$$

i.e., $$-\beta^2 u_{n-1}-\alpha\beta u_n = \beta y_{n+1}$$

i.e., $$-\beta^2 u_{n-1} = \beta y_{n+1}+\alpha\beta u_n \quad .$$

On substituting for $\beta u_n$ from (4.3.58), we have,

$$-\beta^2 u_{n-1} = \beta y_{n+1}-\alpha y_{n+2} \quad . \tag{4.3.59}$$

Next, we combine equations (4.3.58) and (4.3.57b) to obtain,

$$y_{n+2} = -\beta u_n = -\beta \sum_{j=1}^{n} a_j \{b_j+a_{j+1}y_{n+1}+\beta a_j y_{n+2}\}$$

which yields the result,

$$y_{n+2}(1+\beta^2 \sum_{j=1}^{n} a_j^2) = -(\beta \sum_{j=1}^{n} a_j a_{j+1})y_{n+1} - (\beta \sum_{j=1}^{n} a_j b_j) \ . \qquad (4.3.60)$$

Similarly, by combining (4.3.59) and (4.3.57a) we obtain

$$\beta y_{n+1} - \alpha y_{n+2} = -\beta^2 u_{n-1} = -\beta^2 \sum_{j=1}^{n-1} a_j (b_{j+1} + a_{j+2} y_{n+1} + \beta a_{j+1} y_{n+2})$$

which yields

$$y_{n+2}(\beta^3 \sum_{j=1}^{n-1} a_j a_{j+1} - \alpha) = (\beta + \beta^2 \sum_{j=1}^{n-1} a_j a_{j+2})y_{n+1} - (\beta^2 \sum_{j=1}^{n-1} a_j b_{j+1}) \ . \qquad (4.3.61)$$

Both equations (4.3.60) and (4.3.61) can be represented respectively as

$$y_{n+2} Z = -(T y_{n+1} + V) \ , \qquad (4.3.62)$$

and

$$y_{n+2} S = -(U y_{n+1} + W) \ , \qquad (4.3.63)$$

where

$$Z = 1+\beta^2 \sum_{j=1}^{n} a_j^2 \ ,$$

$$S = \beta^3 \sum_{j=1}^{n-1} a_j a_{j+1} - \alpha \ ,$$

$$T = \beta \sum_{j=1}^{n} a_j a_{j+1} \ ,$$

$$U = \beta + \beta^2 \sum_{j=1}^{n-1} a_j a_{j+2} \ ,$$

$$V = \beta \sum_{j=1}^{n} a_j b_j \ ,$$

and

$$W = \beta^2 \sum_{j=1}^{n-1} a_j b_{j+1} \ .$$

From (4.3.62) and (4.3.63) we immediately obtain the value,

$$y_{n+1} = \frac{ZW-SV}{ST-ZU} \ , \quad \text{provided } ST-ZU \neq 0 \qquad (4.3.64)$$

and

$$y_{n+2} = -(T y_{n+1} + V)/Z, \quad Z \neq 0. \qquad (4.3.65)$$

With the values of $y_{n+1}, y_{n+2}$ known from (4.3.64) and (4.3.65) respectively, we return to the system (4.3.48), and use a backward substitution process to obtain the results,

$$y_{n-i+1} = g_{n-i+1} + \alpha y_{n-i+2} + \beta y_{n-i+3} \ , \quad i=1,2,\ldots,n \ . \qquad (4.3.66)$$

Finally, we use the system (4.3.54) in a forward substitution process to obtain the final solution vector as,

$$u_1 = y_1 \ ,$$

$$u_2 = y_2+\alpha y_1 \ ,$$

and

$$u_i = y_i+\alpha u_{i-1}+\beta u_{i-2}, \quad i=3,4,\ldots,n. \tag{4.3.67}$$

Some Simplifications

In the evaluation of Z,S,T, and U the amount of arithmetic calculation can be drastically reduced by the use of the following properties:-

T can be rewritten as,

$$T = \beta \sum_{j=1}^{n} a_j a_{j+1} = \beta\left(\sum_{j=1}^{n-1} a_j a_{j+1}+a_n a_{n+1}\right) .$$

Also by using the relation for $a_j$ in (4.3.52a) we have,

$$a_{j+2} = \alpha a_{j+1}+\beta a_j \ , \qquad j=1,2,\ldots,n-1,$$

hence $a_j a_{j+2} = \alpha a_j a_{j+1}+\beta a_j^2$,

i.e.,
$$\sum_{j=1}^{n-1} a_j a_{j+2} = \alpha \sum_{j=1}^{n-1} a_j a_{j+1}+ \beta \sum_{j=1}^{n-1} a_j^2$$
$$= \alpha \sum_{j=1}^{n-1} a_j a_{j+1}+ \beta\left(\sum_{j=1}^{n} a_j^2-a_n^2\right) .$$

Thus, if we define the quantities

$$\gamma = \sum_{j=1}^{n-1} a_j a_{j+1} \ , \tag{4.3.68}$$

and

$$\rho = \sum_{j=1}^{n} a_j^2 \ , \tag{4.3.69}$$

then Z,S,T,U,V and W can be expressed as,

$$\left.\begin{aligned} Z &= 1+\beta^2\rho \ , \\ S &= \beta^3\gamma-\alpha \ , \\ T &= \beta(\gamma+a_n a_{n+1}) \ , \\ U &= \beta+\beta^2\alpha\gamma+\beta^3\rho-\beta^2 a_n^2 \ , \\ V &= \beta \sum_{j=1}^{n} a_j b_j \ , \end{aligned}\right\} \tag{4.3.70}$$

and

$$W = \beta^2 \sum_{j=1}^{n-1} a_j b_{j+1} \ .$$

We can now summarise the above given quindiagonal matrix solver algorithm as follows:

Quindiagonal (ReTriFE) algorithm (4.6)

Step 1 Determine $\alpha<1$, from the quartic equation, $f^2\alpha^4+2ef\alpha^3+(2f^2+e^2+df)\alpha+(2ef+de)\alpha+e^2=0$ using the Newton-Raphson iterative scheme (4.3.44). Then, compute $\beta = \frac{f\alpha}{e+f\alpha}$ and each of $g_i=(1+\alpha^2+\beta^2)\frac{h_i}{d_i}$, $i=1,2,\ldots,n$.

Step 2 Compute the recurrence relations,

$$a_1 = 1 , \qquad b_i = g_n ,$$

$$a_2 = \alpha , \qquad b_2 = g_{n-1}+\alpha b_1 ,$$

$$a_j = \alpha a_{j-1}+\beta a_{j-2}, \quad j=3,4,\ldots,n+1 ,$$

$$b_j = g_{n-j+1}+\alpha b_{j-1}+\beta b_{j-2} , \quad j=3,4,\ldots,n,$$

Step 3 Compute

$$\gamma = \sum_{j=1}^{n-1} a_j a_{j+1} , \qquad \rho = \sum_{j=1}^{n} a_j^2 ,$$

$$Z = 1+\beta^2\rho ,$$

$$S = \beta^3\gamma-\alpha ,$$

$$T = \beta(\gamma+a_n a_{n+1}) ,$$

$$U = \beta+\beta^2\alpha\gamma+\beta^3\rho-\beta^2 a_n^2 ,$$

$$V = \beta^2 \sum_{j=1}^{n} a_j b_j$$

and

$$W = \beta^2 \sum_{j=1}^{n-1} a_j b_{j+1} .$$

Step 4 Compute $y_{n+1} = (ZW-SV)/(ST-ZU)$ ,

$$y_{n+2} = -(Ty_{n+1}+V)/Z$$

and $y_{n-i+1} = g_{n-i+1}+\alpha y_{n-i+2}+\beta y_{n-i+3}$ , $i=1,2,\ldots,n$.

Finally, the solution vector is obtained from,

$$u_1 = y_1, \qquad u_2 = y_2+\alpha y_1$$

and

$$u_i = y_i+\alpha u_{i+1}+\beta u_{i+2} .$$

The total amount of work required, discounting the application of the Newton-Raphson (or any alternative root finding) algorithm to determine $\alpha$(and $\beta$ obtained from $\alpha$) is of the order of: $13n$ multiplications and $11n$ additions.

Although the algorithm involves more work than the Conte and Dames algorithm (4.3), nevertheless, the Quindiagonal (ReTriFE) algorithm requires only 4 vectors and 10 variables which represents a worthwile storage gain of 2 n-component vectors over the former method.

The strategy of algorithm (4.6) can be generalised to obtain a solution of the more general (n×n) symmetric quindiagonal system,

$$A\underline{u} = \underline{h} \qquad (4.3.71a)$$

where A is of the form,

$$A = \begin{bmatrix} d_1 & e_1 & f_1 & & & & \\ e_1 & d_2 & e_2 & f_2 & & 0 & \\ f_1 & e_2 & d_3 & e_3 & f_3 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & f_{n-1} \\ & 0 & & \ddots & \ddots & \ddots & e_{n-1} \\ & & & & f_{n-2} & e_{n-1} & d_n \end{bmatrix} \qquad (4.3.71b)$$

However, the transformations necessary to convert A into an easily factorisable form leads to the solution of n quartic equations. This makes the rectangular factorisation and expansion method unattractive for the solution of the system (4.3.71b).

It is also possible to apply the ReTriFE strategy to the solution of wider banded constant element linear systems but it was observed that as the bandwidth becomes larger, the analysis becomes more complicated and the derived algorithm less attractive or competitive. The recursive point partitioning approach which we shall introduce in the next section offers a simpler and more efficient method for such wider banded systems.

Algorithm 4.7

Further, pursuing the ReTriFE strategy, we consider the special skew-symmetric constant element (n×n) tridiagonal system,

$$A\underline{u} = \underline{d} \qquad (4.3.72)$$

where A is of the form,

$$\begin{bmatrix} b & c & & & & \\ -c & b & c & & 0 & \\ & -c & b & c & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & c \\ & & & & -c & b \end{bmatrix} \tag{4.3.73}$$

The matrix system (4.3.73) arises, for example, in the solution of parabolic partial differential equations by boundary value techniques (see section (5.4)).

By applying the following transformations,

$$\left.\begin{aligned} \frac{c}{b} = \frac{\alpha}{1-\alpha^2}\ , \quad \text{i.e.,} \quad \alpha = \frac{2c}{b+\sqrt{b^2+4c^2}}\ , \\ \text{and} \qquad g_i = (1-\alpha^2)\frac{d_i}{b}\ , \quad i=1,2,\ldots,n \end{aligned}\right\} \tag{4.3.74}$$

we can rewrite the system (4.3.72), as,

$$\hat{A}\underline{u} = \underline{g} \tag{4.3.75}$$

where $\hat{A}$ is of the form,

$$\begin{bmatrix} 1-\alpha^2 & \alpha & & & & \\ -\alpha & 1-\alpha^2 & \alpha & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & \alpha \\ & & & & -\alpha & 1-\alpha^2 \end{bmatrix}$$

The matrix $\hat{A}$ can now be factorised uniquely into the form,

$$\hat{A} = \hat{P}\hat{Q}$$

where $\hat{P}$ and $\hat{Q}$ are $(n\times(n+1))$ and $((n+1)\times n)$ rectangular matrices given by,

$$\hat{P} = \begin{bmatrix} 1 & \alpha & & & \\ & 1 & \alpha & & 0 \\ & & \ddots & \ddots & \\ & 0 & & 1 & \alpha \\ & & & & 1 \quad \alpha \end{bmatrix}_{n\times(n+1)} \quad \text{and} \quad \hat{Q} = \begin{bmatrix} 1 & & & \\ -\alpha & 1 & & 0 \\ & \ddots & \ddots & \\ & 0 & -\alpha & 1 \\ & & & -\alpha \end{bmatrix}_{(n+1)\times n} \tag{4.3.76}$$

Thus, as usual, the system (4.3.75) is solved easily by considering the system,

$$\hat{P}\hat{Q}\underline{u} = \underline{g} \tag{4.3.77}$$

which gives the two alternative forms,

$$\hat{P}\underline{y} = \underline{g} \tag{4.3.78a}$$

and

$$\hat{Q}\underline{u} = \underline{y} \tag{4.3.78b}$$

where $y_i (i=1,2,\ldots,n+1)$ is an intermediate vector.

It is immediately obtained from (4.3.74) that for $b,c>0$,

$$|\alpha| < 1$$

and for $b=0$, $c\neq 0$,

$$|\alpha| = 1 .$$

Thus, similar to algorithm (4.4), we can use $\alpha$ as a multiplier in an elimination process without incurring numerical instability resulting from the growth of rounding errors.

Hence, by applying the elimination procedure similar to that used in algorithm (4.4) to the two matrix systems (4.3.78a) and (4.3.78b), with $\alpha$ used as a multiplier in the elimination process, it can be shown that the expression for $u_n$ is given by,

$$\begin{aligned}
&u_n[1-\alpha^2+\alpha^4-\ldots+(-1)^i\alpha^{2i}+\ldots+(-1)^n\alpha^{2n}] = \\
&g_n[1-\alpha^2+\alpha^4-\ldots+(-1)^i\alpha^{2i}+\ldots+(-1)^{n-1}\alpha^{2(n-1)}] + \\
&\alpha g_{n-1}[1-\alpha^2+\alpha^4-\ldots+(-1)^i\alpha^{2i}+\ldots+(-1)^{n-2}\alpha^{2(n-2)}] \\
&+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&+\alpha^i g_{n-i}[1-\alpha^2+\alpha^4-\ldots\qquad\qquad+(-1)^{n-i-1}\alpha^{2(n-i-1)}] + \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + \\
&\alpha^{n-2}g_2[1-\alpha^2]+\alpha^{n-1}g_1 ,
\end{aligned} \tag{4.3.79}$$

together with the connecting equation,

$$-\alpha u_n = y_{n+1} . \tag{4.3.80}$$

Then, equation (4.3.79) is simplified, for $|\alpha|<1$, to give,

$$\begin{aligned}
[1-(-\alpha^2)^{n+1}]u_n = {} & g_n[1-(-\alpha^2)^n]+\alpha g_{n-1}[1-(-\alpha^2)^{n-1}] + \ldots \\
& +\alpha^{n-1}g_2[1-(-\alpha^2)^2] + \alpha^{n-1}g_1[1-(-\alpha^2)] ,
\end{aligned}$$

i.e.,

$$[1-(-\alpha^2)^{n+1}]u_n = g_n+\alpha g_{n-1}+\ldots+\alpha^{n-2}g_2+\alpha^{n-1}g_1$$
$$-\alpha^{n+1}[-g_1+\alpha g_2-\alpha^2 g_3+\ldots+(-1)^i\alpha^{i-1}g_i+\ldots+(-1)^n\alpha^{n-1}g_n] \quad (4.3.81)$$

from which, by using a nesting technique, $u_n$ can be computed in 2n multiplications and 2n additions.

A back-substitution process, using (4.3.78a), yields the components of the auxiliary vector as,

$$y_i = g_i-\alpha y_{i+1} \quad , \quad i=n,n-1,\ldots 1 \quad , \quad (4.3.82)$$

where $y_{n+1} = -\alpha u_n$ .

Finally, the solution vector $\underline{u}$ is given by a forward substitution process obtained from (4.3.78b) to give,

$$u_1 = y_1 \quad ,$$

and

$$u_i = y_i+\alpha u_{i-1} \quad , \quad i=2,3,\ldots,n-1. \quad (4.3.83)$$

The algorithmic solution (4.3.81)-(4.3.83) together with the connecting equation (4.3.80) requires 5n multiplications, and 4n additions provided that (4.3.81) is evaluated by a nesting technique.

Special Cases

When b=0, c≠0 then from (4.3.74),

$$|\alpha| = 1$$

and thus the above algorithmic process provides a method for the solution of the special (n×n) singular matrix system,

$$\begin{bmatrix} 0 & c & & & \\ -c & 0 & c & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c \\ & 0 & & -c & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \quad , \quad (4.3.84a)$$

which can easily be transformed to the form,

$$\begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & \ddots & 1 \\ & 0 & & -1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \qquad (4.3.84b)$$

or $$\tilde{A}\underline{u} = \underline{g} \qquad (4.3.85)$$

where $$g_i = d_i/c \quad , \qquad i=1,2,\ldots,n. \qquad (4.3.86)$$

The matrix $\tilde{A}$ has the unique factorisation,

$$\tilde{A} = \tilde{P}\tilde{Q} \qquad (4.3.87)$$

where $\tilde{P}$ and $\tilde{Q}$ are $n\times(n+1)$ and $(n+1)\times n$ rectangular matrices respectively of the form,

$$\tilde{P} = \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & 0 \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & 0 & & & 1 \end{bmatrix}_{n\times(n+1)} \qquad (4.3.88a)$$

and

$$\tilde{Q} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & 0 \\ & \ddots & \ddots & & \\ & & & \ddots & 1 \\ & 0 & & & -1 \end{bmatrix}_{(n+1)\times n} \qquad (4.3.88b)$$

Case 1

When $\alpha=1$, and n=even, then from (4.3.79), we have

$$u_n = g_1+g_3+g_5+\ldots+g_{n-1} \qquad (4.3.89)$$

together with

$$y_{n+1} = -u_n$$

and $$y_i = g_i - y_{i+1} \quad , \qquad i=n,n-1,\ldots,1, \qquad (4.3.90)$$

and finally the solution vector $\underline{u}$ becomes,

$$u_1 = g_1, \quad u_i = y_i+u_{i-1}, \quad i=2,3,\ldots,n-1. \qquad (4.3.91)$$

Case 2

When $\alpha=1$, and n is even then we have,

$$u_n = g_1+g_3+g_5+\ldots+g_{n-1} \tag{4.3.92}$$

together with

$$y_{n+1} = u_n$$
$$y_i = g_i+y_{i+1}, \quad i=n,n-1,\ldots,1 \tag{4.3.93}$$

and finally, $$u_1 = y_1,\ u_i = y_i-u_{i-1},\quad i=2,3,\ldots,n-1. \tag{4.3.94}$$

Thus, the solution of the special forms of the skew-symmetric singular linear system (4.3.84a) with zero diagonal entries can be obtained, for n even, in only $5n/2$ additions which is remarkably fast. The solution of (4.3.84) is however indeterminate using this method, for an odd value of n.

A Fortran implementation of algorithm (4.7) is given as Program 9 of Appendix I.

## 4.4 RECURSIVE POINT PARTITIONING (R.P.P.) METHODS FOR THE FAST SOLUTION OF BANDED MATRIX EQUATIONS ASSOCIATED WITH DIRICHLET'S AND NEUMANN'S BOUNDARY CONDITIONS

### Algorithm 4.8

Next, we introduce the method of recursive point partitioning (R.P.P.) in the solution of sparse banded matrix systems.

First, we consider the general (n×n) tridiagonal matrix equation,

$$Au = \underline{d} \ , \tag{4.4.1}$$

i.e.,

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_1 & b_2 & c_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & c_{n-2} & \\ & 0 & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{4.4.2}$$

which can be rewritten in the following block partitioned form as

$$\begin{bmatrix} \alpha_1^{(1)} & \underline{v}_1^T & 0 \\ \underline{w}_1 & \tilde{Q}_{2\to n-1} & \underline{z}_1 \\ 0 & \underline{r}_1^T & \alpha_n^{(1)} \end{bmatrix} \begin{bmatrix} u_1 \\ \tilde{\underline{u}} \\ u_n \end{bmatrix} = \begin{bmatrix} d_1^{(1)} \\ \tilde{\underline{d}}^{(1)} \\ d_n^{(1)} \end{bmatrix} \tag{4.4.3}$$

where $\underline{v}_1, \underline{w}_1, \underline{z}_1, \underline{r}_1, \tilde{\underline{u}}$ and $\tilde{\underline{d}}^{(1)}$ are (n-2) component vectors given by,

$$\underline{v}_1^T = (c_1, 0, 0, \ldots, 0) \ ,$$

$$\underline{w}_1 = (a_1, 0, 0, \ldots, 0)^T \ ,$$

$$\underline{z}_1 = (0, 0, 0 \ldots, 0, c_{n-1})^T \ ,$$

$$\underline{r}_1^T = (0, 0, 0, .., 0, a_{n-1}) \ ,$$

$$\tilde{\underline{u}} = (u_2, u_3, \ldots, u_{n-1}) \ ,$$

and $$\tilde{\underline{d}}^{(1)} = (d_2, d_3, \ldots, d_{n-1})^T \ ;$$

and the quantities $\alpha_1^{(1)}, \alpha_n^{(1)}$, $d_1^{(1)}$ and $d_n^{(1)}$ are given by,

$$\alpha_1^{(1)} = b_1 \ , \qquad \alpha_n^{(1)} = b_n \ ,$$
$$d_1^{(1)} = d_1 \quad \text{and} \quad d_n^{(1)} = d_n \ . \tag{4.4.4}$$

$\tilde{Q}_{2\to n-1}$ is a submatrix of order n-2 obtained from the 2nd to the $(n-1)^{th}$ rows and columns of the original matrix.

From the block form (4.4.3) we obtain the equations,

$$\left.\begin{aligned} \alpha_1^{(1)} u_1 + \underline{v}_1^T \underline{\tilde{u}} &= d_1^{(1)} , \\ \underline{w}_1 u_1 + \tilde{Q}_{2\to n-1} \underline{\tilde{u}} + \underline{z}_1 u_n &= \underline{\tilde{d}}^{(1)} , \\ \underline{r}_1^T \underline{\tilde{u}} + \alpha_n^{(1)} u_n &= d_n^{(1)} \end{aligned}\right\} \qquad (4.4.5)$$

from which we obtain the scalar unknowns,

$$\left.\begin{aligned} u_1 &= \frac{d_1^{(1)}}{\alpha_1^{(1)}} - \frac{\underline{v}_1^T \underline{\tilde{u}}}{\alpha_1^{(1)}} , \\ \text{and} \qquad u_n &= \frac{d_n^{(1)}}{\alpha_n^{(1)}} - \frac{\underline{r}_1^T \underline{\tilde{u}}}{\alpha_n^{(1)}} ; \end{aligned}\right\} \qquad (4.4.6)$$

and also the reduced matrix system of order (n-2) given by,

$$\left(Q_{2\to n-1} - \frac{\underline{w}_1 \underline{v}_1^T}{\alpha_1^{(1)}} - \frac{\underline{z}_1 \underline{r}_1^T}{\alpha_n^{(1)}}\right) \underline{\tilde{u}} = \underline{\tilde{d}}^{(1)} - \frac{\underline{w}_1 d_1^{(1)}}{\alpha_1^{(1)}} - \frac{\underline{z}_1 d_n^{(1)}}{\alpha_n^{(1)}} . \qquad (4.4.7)$$

It is easy to verify by a simple substitution process that the system (4.4.7), when expressed in matrix notation, is configured in a similar manner as the original system (4.4.2) but of order 2 less and is given by,

$$\begin{bmatrix} \alpha_2^{(2)} & c_2 & & & \\ a_2 & b_3 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-3} & b_{n-2} & c_{n-2} \\ & & & a_{n-2} & \alpha_{n-1}^{(2)} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} d_2^{(2)} \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1}^{(2)} \end{bmatrix} \qquad (4.4.8)$$

where

$$\left.\begin{aligned} \alpha_2^{(2)} &= b_2 - \frac{a_1 c_1}{\alpha_1^{(1)}} , \quad d_2^{(2)} = d_2^{(1)} - \frac{a_1 d_1^{(1)}}{\alpha_1^{(1)}} , \\ \alpha_{n-1}^{(2)} &= b_{n-1} - \frac{a_{n-1} c_{n-1}}{\alpha_n^{(1)}} \quad \text{and} \quad d_{n-1}^{(2)} = d_{n-1}^{(1)} - \frac{c_{n-1} d_n^{(1)}}{\alpha_n^{(1)}} . \end{aligned}\right\} \qquad (4.4.9)$$

Further, it is easily observed that the reduced system (4.4.8) can be similarly partitioned and rewritten in the form,

$$\begin{bmatrix} \alpha_2^{(2)} & \underline{v}_2^T & 0 \\ \underline{w}_2 & \tilde{Q}_{3\to n-2} & \underline{z}_2 \\ 0 & \underline{r}_2^T & \alpha_{n-1}^{(2)} \end{bmatrix} \begin{bmatrix} u_2 \\ \underline{\tilde{u}} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} d_2^{(2)} \\ \underline{d}^{(2)} \\ d_{n-1}^{(2)} \end{bmatrix} \qquad (4.4.10)$$

from which a further reduced system similar in structure to (4.4.7) is derived.

Thus, we can set up a recursive procedure in which the above reduction process is continued such that at the $i^{th}$ partitioning stage we have, identical to (4.4.4) and (4.4.9), the following recurrence relations,

$$\left.\begin{aligned} \alpha_i^{(i)} &= b_i - \frac{a_{i-1}c_{i-1}}{\alpha_{i-1}^{(i-1)}}, \quad d_i^{(i)} = d_i^{(i-1)} - \frac{a_{i-1}d_{i-1}^{(i)}}{\alpha_{i-1}^{(i-1)}} \\ \text{and} \\ \alpha_{n-i+1}^{(i)} &= b_{n-i+1} - \frac{a_{n-i+1}}{\alpha_{n-i+2}^{(i-1)}} c_{n-i+1}, \quad d_{n-i+1}^{(i)} = d_{n-i+1}^{(i-1)} - \frac{c_{n-i+1}}{\alpha_{n-i+2}^{(i-1)}} d_{n-i+2}^{(i-1)} \end{aligned}\right\} \qquad (4.4.11)$$

The recursive procedure is then continued until the reduced system is of order 1 for n odd; and of order 2 for n even.

Case 1: n odd

For n odd, the final reduced system is given, for q=(n+1)/2, by the system,

$$\left(\tilde{Q}_{q\to n-q+1} - \frac{\underline{w}_{q-1}\underline{v}_{q-1}^T}{\alpha_{q-1}^{(q-1)}} - \frac{\underline{z}_{q-1}\underline{r}_{q-1}^T}{\alpha_{n-q+2}^{(q-1)}}\right) u_q = \alpha_q^{(q-1)} - \frac{\underline{w}_{q-1}d_q^{(q-1)}}{\alpha_{q-1}^{(q-1)}} - \frac{\underline{z}_{q-1}d_{n-q+2}^{(q-1)}}{\alpha_{n-q+2}^{(q-1)}} \qquad (4.4.12)$$

which can be verified to be the single equation,

$$\alpha_q^{(q)} u_q = d_q^{(q)}, \qquad (4.4.13)$$

where

$$\left.\begin{aligned} \alpha_q^{(q)} &= b_q - \left(\frac{a_{q-1}c_{q-1}}{\alpha_{q-1}^{(q-1)}} + \frac{a_{q+1}c_{n-q+1}}{\alpha_{n-q+2}^{(q-1)}}\right) \\ \text{and} \\ d_q^{(q)} &= d_q^{(q-1)} - \left(\frac{d_{q-1}^{(q-1)}a_{q-1}}{\alpha_{q-1}^{(q-1)}} + \frac{d_{n-q+2}^{(q-1)}c_{q+1}}{\alpha_{n-q+2}^{(q-1)}}\right) \end{aligned}\right\} \qquad (4.4.14)$$

Case 2: n even

For n even, the final reduced system is given by (4.4.12) (for q=n/2) and can be verified to be the same as the (2×2) system,

$$\begin{bmatrix} \alpha_q^{(q)} & c_q \\ a_{n-q} & \alpha_{n-q+1}^{(q)} \end{bmatrix} \begin{bmatrix} u_q \\ u_{q+1} \end{bmatrix} = \begin{bmatrix} d_q^{(q)} \\ d_{n-q+1}^{(q)} \end{bmatrix} \tag{4.4.15}$$

which is solved to give,

$$\left.\begin{aligned} u_q &= (d_q^{(q)}\alpha_{n-q+1}^{(q)} - d_{n-q+1}^{(q)}c_q)/(\alpha_{n-q+1}^{(q)}\alpha_q^{(q)} - a_{n-q}c_q) \\ \text{and} \quad u_{n-q+1} &= (\alpha_q^{(q)}d_{n-q+1}^{(q)} - d_q^{(q)}a_{n-q})/(\alpha_{n-q+1}^{(q)}\alpha_q^{(q)} - a_{n-q}c_q) \end{aligned}\right\} \tag{4.4.16}$$

where $\alpha_q^{(q)}$, $\alpha_{n-q+1}^{(q)}$, $d_q^{(q)}$ and $d_{n-q+1}^{(q)}$ are as defined in (4.4.11).

Finally after obtaining the central values of $\underline{u}$ by using (4.4.13), for n odd, or (4.4.16), for n even, we proceed to obtain the remaining elements of the solution vector by a back-substitution process. By using the equations of the scalar unknown obtained at each stage of the reduction process (e.g. (4.4.6) for the 1st reduction) we derive the other elements of the solution vector as,

$$\left.\begin{aligned} u_i &= (d_i^{(i)} - c_i u_{i+1})/\alpha_i^{(i)} \\ \text{and} \quad u_{n-i+1} &= (d_{n-i+1}^{(i)} - a_{n-i}u_{n-i})/\alpha_{n-i+1}^{(i)}, \end{aligned}\right\} \quad i=q-1,q-2,\ldots,1. \tag{4.4.17}$$

We denote the above scheme as the tridiagonal recursive point partitioning (TRPP) algorithm which is given as Program 10 of Appendix I and may be summarised as follows:

TRPP Algorithm (4.8)

Step 1 Set up the following quantities,

$$\begin{aligned} \alpha_1^{(1)} &= b_1, \qquad d_1^{(1)} = d_1, \\ \alpha_n^{(1)} &= b_n, \qquad d_n^{(1)} = d_n \end{aligned} \tag{4.4.18}$$

and $q = n/2$ (n even) or $(n+1)/2$ (n odd).

Step 2 Compute the following quantities in the partitioning process:

$$t_{i-1} = \frac{a_{i-1}}{\alpha_{i-1}^{(i-1)}}, \quad s_{i-1} = \frac{c_{n-i+1}}{\alpha_{n-i+2}^{(i-1)}}, \quad i=2,3,\ldots,q, \tag{4.4.19a}$$

$$\left.\begin{aligned} \alpha_i^{(i)} &= b_i - t_{i-1}c_{i-1}, \quad d_i^{(i)} = d_i^{(i-1)} - t_{i-1}d_{i-1}^{(i-1)} \\ \alpha_{n-i+1}^{(i)} &= b_{n-i+1} - s_{i-1}a_{n-i+1}, \quad d_{n-i+1}^{(i)} = d_{n-i+1}^{(i-1)} - s_{i-1}d_{n-i+2}^{(i-1)} \end{aligned}\right\} \begin{array}{l} i=2,3,\ldots,q-1 \\ \text{(n odd)} \\ \\ i=2,3,\ldots,q \\ \text{(n even)} \end{array} \quad (4.4.19b)$$

(4.4.19c)

and (for n odd only), compute the quantities,

$$\alpha_q^{(q)} = b_q - (t_{q-1}c_{q-1} + s_{q-1}a_{q+1}) \quad ,$$

and

$$d_q^{(q)} = d_q^{(q-1)} - (t_{q-1}d_{q-1}^{(q-1)} + s_{q-1}d_{n-q+2}^{(q-1)}).$$

Step 3 Finally, compute the solution vector $\underline{u}$ by using the formulae,

$$u_q = d_q^{(q)}/\alpha_q^{(q)} \qquad \text{(if n is odd)} \qquad (4.4.20a)$$

or

$$\left.\begin{aligned} u_q &= (d_q^{(q)}\alpha_{n-q+1}^{(q)} - d_{n-q+1}^{(q)}c_q)/(\alpha_{n-q+1}^{(q)}\alpha_q^{(q)} - a_{n-q}c_q) \\ u_{n-q+1} &= (\alpha_q^{(q)}d_{n-q+1}^{(q)} - d_q^{(q)}a_{n-q})/(\alpha_{n-q+1}^{(q)}\alpha_q^{(q)} - a_{n-q}c_q) \end{aligned}\right\} \text{ if n is even} \quad (4.4.20b)$$

and then $u_i = (d_i^{(i)} - c_i u_{i+1})/\alpha_i^{(i)}$ ,

and $u_{n-i+1} = (d_{n-i+1}^{(i)} - a_{n-i}u_{n-i})/\alpha_{n-i+1}^{(i)}$ , $\left.\right\} i=q-1,q-2,\ldots,1.$ (4.4.20c)

The TRPP algorithm requires less than 3n multiplications, 2n divisions and 3n additions (a slight improvement over the Thomas algorithm (4.1)). For repeated solutions with the coefficient matrix unchanged, only the terms $d_i^{(i)}$ and $u_i$ need to be recalculated in 2n multiplications, n divisions and 2n additions. Furthermore, the $\alpha_i^{(i)}$ and $d_i^{(i)}$ terms can overwrite the diagonal elements, $b_i$ and the right-hand side elements, $d_i$ respectively, thus giving a gain of two n-component vector storage over the Thomas algorithm.

The TRPP algorithm is simply a systematic modification of the diagonal and right-hand side vectors from both the top and the bottom of the respective vectors towards the centre where it then becomes a simple outward process to calculate the solution vector.

Algorithm 4.9

The recursive point partitioning strategy introduced in algorithm (4.8) offers further computational gains when applied to the solution of the simple

constant element tridiagonal system,

$$A\underline{u} = \underline{d} \tag{4.4.21}$$

where A is of the form,

$$A = \begin{bmatrix} \lambda & -1 & & & \\ -1 & \lambda & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -1 & \lambda \end{bmatrix} \tag{4.4.22}$$

A variant of algorithm (4.8) which gives an efficient fast solution of the above matrix system is given below and is denoted as the CTRPP algorithm.

CTRPP Algorithm (4.9)

Step 1 First, we set up the quantities,

$$\alpha_1^{(1)} = \alpha_n^{(1)} = \lambda \quad , \tag{4.4.23a}$$

$$d_1^{(1)} = d_1 \, , \quad d_n^{(1)} = d_n \, , \tag{4.4.23b}$$

and q=n/2 (for n even) or (n+1)/2 (for n odd).

Step 2 Then, we compute the following recursive relations,

$$t_{i-1} = \frac{1}{\alpha_{i-1}^{(i-1)}} \quad , \quad i=2,3,\ldots,q \tag{4.4.24a}$$

$$\left.\begin{aligned} \alpha_i^{(i-1)} &= \alpha_{n-i+1}^{(i)} = \lambda - t_{i-1} \\ d_i^{(i)} &= d_i^{(i-1)} + t_{i-1} d_{i-1}^{(i-1)} \\ d_{n-i+1}^{(i)} &= d_{n-i+1}^{(i-1)} + t_i d_{n-i+2} \end{aligned}\right\} \begin{array}{l} i=2,\ldots,q-1, \text{ (n odd)} \\ \quad \text{or} \\ i=2,\ldots,q, \text{ (n even)} \end{array} \tag{4.4.24b}$$

and for n odd only,

$$\alpha_q^{(q)} = \lambda - 2t_{q-1} \, , \quad d_q^{(q)} = d_q^{(q-1)} + (d_{q-1}^{(q-1)} t_{q-1} + d_{n-q+2}^{(q-1)} t_{q-1}) . \tag{4.4.24c}$$

Step 3 Next, we obtain the central elements of the solution vector by using the formula,

$$u_q = d_q^{(q)} / \alpha_q^{(q)} \quad \text{(if n is odd)} \tag{4.4.25a}$$

$$\left.\begin{aligned} \text{or} \qquad u_q &= (d^{(q)}_{n-q+1}+\alpha_q d^{(q)}_q)/(\alpha^{(q)}_q \alpha^{(q)}_{n-q+1}-1) \\ \text{and} \qquad u_{n-q+1} &= (d^{(q)}_q+\alpha^{(q)}_q d^{(q)}_{n-q+1})/(\alpha^{(q)}_q \alpha^{(q)}_{n-q+1}-1) \end{aligned}\right\} \text{(if n is even).} \quad (4.4.25b)$$

Finally, the remaining components of the solution vector are obtained from,

$$\left.\begin{aligned} u_i &= (d^{(i)}_i+u_{i+1})t_i \quad , \\ \text{and} \qquad u_{n-i+1} &= (d^{(i)}_{n-i+1}+u_{n-i})t_i \quad , \end{aligned}\right\} \; i=q-1,q-2,\ldots 1. \quad (4.4.25c)$$

This special variant of the TRPP algorithm (4.8) requires less than 2n multiplications n/2 divisions and 5n/2 additions; and for repeated solutions, only 2n multiplications and 2n additions. Furthermore, in a few special applications (e.g. in the solution of Laplace equation in a square region with Dirichlet's boundary condition, the right-hand side vector is entirely dependent on the boundary conditions, and hence such right hand side vector may be such that it is symmetric about its centre. In such a situation, the $d^{(i)}_i$ and $d^{(i)}_{n-i+1}$ terms are equal; thus enabling algorithms (4.8) and its variant (4.9) to be implemented with a further saving of n/2 multiplications and n/2 additions in the operational count.

Special cases for Neumann's boundary conditions

In the solution of two point boundary value problem with Neumann's boundary conditions, we are often required to solve the system,

$$A\underline{u} = \underline{d} \quad , \quad (4.4.26)$$

where A is of the form,

$$\begin{bmatrix} \lambda & -2 & & & \\ -1 & \lambda & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -2 & \lambda \end{bmatrix} \quad (4.4.27a)$$

The first super-diagonal and the last sub-diagonal elements are modified by the Neumann's boundary condition when the derivatives, $\frac{\partial U}{\partial x}$, $\frac{\partial U}{\partial y}$ (which specify the boundary conditions) are approximated by central differences.

In this case, the CTRPP algorithm (4.9) is immediately applicable; with only a minor modification in the initialisation stage (i.e., $t_1 = \frac{2}{\alpha(1)}$).

Further, when the Neumann's boundary condition, $\frac{\partial U}{\partial n}$ is approximated by one sided differences (instead of the central differences) then we are often required to solve the system (4.4.26) where the coefficient matrix A is now of the form,

$$A = \begin{bmatrix} \lambda' & -1 & & & \\ -1 & \lambda & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ 0 & & & \lambda & -1 \\ & & & -1 & \lambda' \end{bmatrix} \qquad (4.4.27b)$$

with the first and last diagonal elements modified by the boundary conditions. In this case, algorithm (4.9) is also immediately applicable.

Solution of Singular Systems

It is, however, well known that for the Neumann boundary condition the coefficient matrix can become singular; e.g., if $\lambda'=1$, then the smallest eigenvalue of the matrix A in (4.4.27b) is zero and hence A becomes singular. In such a case, additional constraints have to be imposed on the right-hand side vector (i.e., the source function in physical terms) in order to guarantee the solvability of the matrix system. This can be achieved (Machuk (1975)) by subtracting a common factor ($\bar{g}$ say) from the components of the right-hand side vector $\underline{d}$ such that,

$$\bar{g} = \sum_{i=1}^{n} (d_i/n) \; . \qquad (4.4.28)$$

In the actual realisation of the algorithmic solution of the subsequent modified problem by the method of algorithm (4.9), it is possible to have in the calculation process, operations of the type "0/0" which are to be replaced by any arbitrary constant.

Algorithm 4.10

We further apply the recursive point partitioning (RPP) strategy to derive an algorithmic solution for the constant element symmetric quindiagonal system,

$$A\underline{u} = \underline{d} \tag{4.4.29}$$

where,

$$A = \begin{bmatrix} b & a & c & & & & & \\ a & b & a & c & & & 0 & \\ c & a & b & a & c & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & b & a & c & \\ & 0 & & & a & b & a & \\ & & & & c & a & b & \end{bmatrix} . \tag{4.4.30}$$

The above system occurs, for example, in the finite difference approximation of Biharmonic and $4^{th}$ order boundary value problems (Bauer et al (1972)).

By rewriting the system (4.4.29) in the form,

$$\begin{bmatrix} b_1^{(1)} & \underline{v}_1^T & 0 \\ \underline{v}_1 & \tilde{Q}_{2\to n-1} & \underline{z}_1 \\ 0 & \underline{z}_1^T & b_1^{(1)} \end{bmatrix} \begin{bmatrix} u_1 \\ \tilde{\underline{u}} \\ u_n \end{bmatrix} = \begin{bmatrix} d_1^{(1)} \\ \tilde{\underline{d}}^{(1)} \\ d_n^{(1)} \end{bmatrix} \tag{4.4.32}$$

where $\underline{v}_1$, $\underline{z}_1$, $\tilde{\underline{u}}, \tilde{\underline{d}}^{(1)}$ are the (n-2)-component vectors given by,

$$\underline{v}_1^T = (a_1^{(1)}, c, 0, \ldots, 0, 0) \quad ,$$

$$\underline{z}_1^T = (0, 0, \ldots c, a_1^{(1)}) \quad ,$$

$$\tilde{\underline{u}}^T = (u_2, u_3, \ldots, u_{n-1}) \quad ,$$

$$\tilde{\underline{d}}^T = (d_2, d_3, \ldots, d_{n-1}) \quad ,$$

$$b_1^{(1)} = b \quad , \qquad a_1^{(1)} = a \quad , \qquad \text{and} \quad d_i^{(1)} = d_i \quad ,$$

we can derive the following scalar unknowns,

$$u_1 = \frac{d_1^{(1)}}{b_1^{(1)}} - \frac{\underline{v}_1^T \tilde{\underline{u}}}{b_1^{(1)}} \quad ,$$

$$u_n = \frac{d_n^{(1)}}{b_1^{(1)}} - \frac{\underline{z}_1^T \tilde{\underline{u}}}{b_1^{(1)}} \quad ,$$

and also the reduced system of order (n-2) given by,

$$\left(Q_{2\to n-1} - \frac{\underline{v}_1\underline{v}_1^T}{b_1^{(1)}} - \frac{\underline{z}_1\underline{z}_1^T}{b^{(1)}}\right)\underline{\tilde{u}} = d_1^{(1)} - \frac{\underline{v}_1 d_1^{(1)}}{b_1^{(1)}} - \frac{\underline{z}_1 d_n^{(1)}}{b_1^{(1)}} \quad . \qquad (4.4.33)$$

Since $\underline{v}_1\underline{v}_1^T$, $\underline{z}_1\underline{z}_1^T$ are matrices of the form,

$$\underline{v}_1\underline{v}_1^T = \begin{bmatrix} (a_1^{(1)})^2 & a_1^{(1)}c & & & \\ a_1^{(1)}c & c^2 & & 0 & \\ & & 0 & & \\ & 0 & & \ddots & \\ & & & & 0 \end{bmatrix}_{(n-2)\times(n-2)}$$

and

$$\underline{z}_1\underline{z}_1^T = \begin{bmatrix} 0 & & & & \\ & \ddots & & 0 & \\ & & 0 & & \\ & 0 & & c^2 & a_1^{(1)}c \\ & & & a_1^{(1)}c & (a_1^{(1)})^2 \end{bmatrix}_{(n-2)\times(n-2)}$$

it is easily verified by substitution that (4.4.33) is the same as the following (n-2)×(n-2) matrix system,

$$\begin{bmatrix} b_2^{(2)} & a_2^{(2)} & c & & & & \\ a_2^{(2)} & b_3^{(2)} & a & c & & & \\ c & a & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & a & c & \\ & & c & a & b_3^{(2)} & a_2^{(2)} & \\ & & & c & a_2^{(2)} & b_2^{(2)} & \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} d_2^{(2)} \\ d_3^{(2)} \\ d_4 \\ \vdots \\ d_{n-3} \\ d_{n-2}^{(2)} \\ d_{n-1}^{(2)} \end{bmatrix} \qquad (4.4.33)$$

where

$$\left.\begin{aligned} b_2^{(2)} &= b_2^{(1)} - \frac{(a_1^{(1)})^2}{b_1^{(1)}}, \quad b_3^{(2)} = b_3^{(1)} - \frac{c^2}{b_1^{(1)}}, \\ a_2^{(2)} &= a_2^{(1)} - a_1^{(1)}c, \\ d_2^{(2)} &= d_2^{(1)} - \frac{a_1^{(1)}}{b_1^{(1)}} d_1^{(1)}, \\ d_3^{(2)} &= d_3^{(1)} - \frac{c d_1^{(1)}}{b_1^{(1)}}, \end{aligned}\right\} \qquad (4.4.34)$$

$$d_{n-1}^{(2)} = d_{n-1}^{(1)} - \frac{a_1^{(1)} d_n^{(1)}}{b_1^{(1)}} \quad ,$$

and

$$d_{n-2}^{(2)} = d_{n-2}^{(1)} - \frac{c d_n^{(1)}}{b_1^{(1)}} \quad .$$

Again, the reduced system (4.4.33) has the same structure as the original system (4.4.29) with modification only in the first and last two equations. Hence, we can set up a recursive reduction procedure such that at the $i^{th}$ stage of the process, where $i=1,2,\ldots,q$, ($q=n/2$ ($n$ even) and $q=(n-1)/2$ ($n$ odd)), we have the recurrence relations,

$$\left.\begin{aligned}
&b_i^{(i)} = b_i^{(i-1)} - \frac{(a_{i-1}^{(i-1)})^2}{b_{i-1}^{(i-1)}} \quad , \quad b_{i+1}^{(i)} = b_{i+1}^{(i-1)} - \frac{c^2}{b_{i-1}^{(i-1)}} \quad , \\
&a_i^{(i)} = a_i^{(i-1)} - \frac{a_{i-1}^{(i-1)} c}{b_{i-1}^{(i-1)}} \quad , \\
&d_i^{(i)} = d_i^{(i-1)} - \frac{a_{i-1}^{(i-1)}}{b_{i-1}^{(i-1)}} d_{i-1}^{(i-1)} \quad , \quad d_{i+1}^{(i)} = d_{i+1}^{(i-1)} - \frac{c d_{i-1}^{(i-1)}}{b_{i-1}^{(i-1)}} \\
\text{and} \quad &d_{n-i+1}^{(i)} = d_{n-i+1}^{(i-1)} - \frac{a_{i-1}^{(i-1)}}{b_{i-1}^{(i-1)}} d_{n-i+2}^{(i-1)} \quad , \quad d_{n-i}^{(i)} = d_{n-i}^{(i-1)} - \frac{c d_{n-i+2}^{(i-1)}}{b_{i-1}^{(i-1)}} \quad .
\end{aligned}\right\} \tag{4.4.35}$$

The quantities $\frac{a_{i-1}^{(i-1)}}{b_{i-1}^{(i-1)}}$ , $\frac{c}{b_{i-1}^{(i-1)}}$ appear more than once in the recurrence relation and so should be computed first and then used where needed to compute the quantities as in (4.4.35).

After the $q^{th}$ level, for $n$ even, the reduced system is of order 2 given by,

$$\begin{bmatrix} b_q^{(q)} & a_q^{(q)} \\ a_q^{(q)} & b_{n-q+1}^{(q)} \end{bmatrix} \begin{bmatrix} u_q \\ u_{n-q+1} \end{bmatrix} = \begin{bmatrix} d_q^{(q)} \\ d_{n-q+1}^{(q)} \end{bmatrix} \quad , \tag{4.4.36}$$

from which the central values of the solution vector, $u_q$, $u_{n-q+1}$ are obtained as,

$$u_q = (d_q^{(q)} b_{n-q+1}^{(q)} - d_{n-q+1}^{(q)} a_q^{(q)}) / (b_{n-q+1}^{(q)} b_q^{(q)} - (a_q^{(q)})^2)$$

and $$u_{n-q+1} = (b_q^{(q)} d_{n-q+1}^{(q)} - a_q^{(q)} d_q^{(q)}) / (b_{n-q+1}^{(q)} b_q^{(q)} - (a_q^{(q)})^2). \qquad (4.4.37)$$

For n odd, q=(n-1)/2, and the final reduced system is then of order 3 and is given by,

$$\begin{bmatrix} b_q^{(q)} & a_q^{(q)} & c \\ a_q^{(q)} & b_{q+1}^{(q)} & a_q^{(q)} \\ c & a_q^{(q)} & b_q^{(q)} \end{bmatrix} \begin{bmatrix} u_q \\ u_{q+1} \\ u_{n-q+1} \end{bmatrix} = \begin{bmatrix} d_q^{(q)} \\ d_{q+1}^{(q)} \\ d_{n-q+1}^{(q)} \end{bmatrix} \qquad (4.4.38)$$

which is solved, omitting pivoting, to give,

$$\left.\begin{aligned} u_{n-q+1} &= \frac{(d_{n-q+1}^{(2)} - m_2 d_q^{(q)} - m_3 (d_{q+1}^{(q)} - m_1 d_q^{(q)}))}{b_q^{(q)} - m_2 c - m_3 (a_q^{(q)} - m_1 c)} \quad , \\ u_{q+1} &= \frac{d_{q+1}^{(q)} - m_1 d_q^{(q)} - u_{n-q+1}(a_q^{(q)} - m_1 c)}{b_{q+1}^{(q)} - m_1 a_q^{(q)}} \quad , \\ \text{and} \quad u_q &= (d_q^{(q)} - a_q^{(q)} u_{q+1} - c u_{n-q+1}) / b_q^{(q)} \quad , \end{aligned}\right\} \qquad (4.4.39)$$

where

$$m_1 = a_q^{(q)} / b_q^{(q)} \quad ,$$

$$m_2 = c_{n-q} / b_q^{(q)} \quad ,$$

and $$m_3 = (a_q^{(q)} - m_2 a_q^{(q)}) / (b_{q+1}^{(q)} - m_1 a_q^{(q)}) \quad .$$

Finally, the back substitution formulae which give the other elements of the solution vector $\underline{u}$ are given by,

$$\left.\begin{aligned} u_i &= (d_i^{(i)} - a_i^{(i)} u_{i+1} - c u_{i+2}) / b_i^{(i)} \\ \text{and} \quad u_{n-i+1} &= (d_{n-i+1}^{(i)} - a_i^{(i)} u_{n-i} - c u_{n-i+1}) / b_i^{(i)} \end{aligned}\right\} i=q-1, q-2, \ldots, 1. \qquad (4.4.40)$$

The above outlined constant quindiagonal recursive point partitioning (CQRPP) algorithm may be summarised below as follows:

CQRPP Algorithm (4.10)

Step 1 Set up the following initial quantities,

$$q = n/2 \text{ (n even)}, \text{ or } (n-1)/2 \text{ (n odd)} , \qquad (4.4.41a)$$

$$\left.\begin{aligned} b_i^{(1)} &= b \quad , \\ a_i^{(1)} &= a \quad , \\ d_i^{(1)} &= d_i \quad , \\ \text{and} \quad d_{n-i+1}^{(1)} &= d_{n-i+1} \end{aligned}\right\} \quad i=1,2,\ldots,q. \qquad (4.4.41b)$$

Step 2 Compute the recurrence relations,

$$\left.\begin{aligned} & t_{i-1} = \frac{a_{i-1}^{(i-1)}}{b_{i-1}^{(i-1)}} \quad , \qquad s_{i-1} = \frac{c}{b_{i-1}^{(i-1)}} \quad , \\ & b_i^{(i)} = b_i^{(i-1)} - t_{i-1} a_{i-1}^{(i-1)} , \quad b_{i+1}^{(i)} = b_{i+1}^{(i-1)} - s_{i-1} c \quad , \\ & a_i^{(i)} = a_i^{(i-1)} - c t_{i-1} \quad , \\ & d_i^{(i)} = d_i^{(i-1)} - t_{i-1} d_{i-1}^{(i-1)} \quad , \quad d_{i+1}^{(i)} = d_{i+1}^{(i-1)} - s_{i-1} d_{i-1}^{(i-1)} \\ \text{and} \quad & d_{n-i+1}^{(i)} = d_{n-i+1}^{(i-1)} - t_{i-1} d_{n-i+2}^{(i-1)} , \quad d_{n-i}^{(i)} = d_{n-i}^{(i-1)} - s_{i-1} d_{n-i+2}^{(i-1)} \end{aligned}\right\} \quad i=2,3,\ldots,q. \qquad (4.4.42)$$

Step 3 Compute the central values of the solution vector by using (4.4.37) for n even or (4.4.39) for n odd.

Step 4 Finally, we obtain the remaining components of the solution vector by using the recurrence relations,

$$\left.\begin{aligned} u_i &= d_i^{(i)} - t_i u_{i+1} - s_i u_{i+2} \\ \text{and} \quad u_{n-i+1} &= d_{n-i+1}^{(i)} - t_i u_{n-i} - s_i u_{n-i-1} \end{aligned}\right\} \quad i=q-1,q-2,\ldots,1. \qquad (4.4.43)$$

This algorithm requires the order of 11n/2 additions which gives a significant improvement over the Conte and Dames algorithm (4.3). For repeated solutions with unchanged coefficient matrix the order of only 4n multiplications and 4n additions is required.

Algorithm 4.11

The method of algorithm (4.10) is easily generalised to apply to the more general symmetric quindiagonal matrix equation,

$$A\underline{u} = \underline{d} \qquad (4.4.44)$$

where

$$A = \begin{bmatrix} b_1 & a_1 & c_1 & & & & \\ a_1 & b_2 & a_2 & c_2 & & 0 & \\ c_1 & a_1 & b_3 & a_3 & c_3 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & c_{n-2} \\ & & 0 & & \ddots & b_{n-1} & a_{n-1} \\ & & & & c_{n-1} & a_{n-1} & b_n \end{bmatrix} \quad . \qquad (4.4.45)$$

The derived algorithmic solution which we may denote as the General Quindiagonal Recursive Point Partitioning (GQRPP) method is summarised below and given as Program 11 of Appendix I.

GQRPP Algorithm (4.11)

Step 1 Set up the following,

$$q = n/2 \text{ (n even)} \quad \text{or} \quad (n-1)/2 \text{ (n odd)} \ ;$$

$$\left.\begin{aligned} b_i^{(1)} &= b_i \ , \\ a_i^{(1)} &= a_i \ , \\ \text{and} \quad d_i^{(1)} &= d_i \ , \end{aligned}\right\} \quad i=1,2,\ldots,n-1,n. \qquad (4.4.46)$$

Step 2 Compute the following recurrence reduction formulae, for i=2,3,...,q:

$$\left.\begin{aligned} t_{i-1} &= \frac{a_i^{(i-1)}}{b_{i-1}^{(i-1)}} \ , \quad v_{i-1} = \frac{a_{n-i+1}^{(i-1)}}{b_{n-i+2}^{(i-1)}} \\ r_{i-1} &= \frac{c_{i-1}}{b_{i-1}^{(i-1)}} \ , \quad s_{i-1} = \frac{c_{n-i}}{b_{n-i+2}^{(i-1)}} \end{aligned}\right\} \qquad (4.4.47a)$$

and then obtain

$$\left.\begin{aligned} & b_i^{(i)} = b_i^{(i-1)} - a_{i-1}^{(i-1)} t_{i-1} \ , \ b_{i+1}^{(i)} = b_{i+1}^{(i-1)} - r_{i-1} c_{i-1} \ , \\ & b_{n-i+1}^{(i)} = b_{n-i+1}^{(i-1)} - v_{i-1} a_{n-i+1}^{(i-1)} \ , \ b_{n-i}^{(i)} = b_{n-i}^{(i-1)} - s_{i-1} c_{n-i} \ , \\ & a_i^{(i)} = a_i - t_{i-1} c_{i-1} \ , \ a_{n-i}^{(i)} = a_{n-i}^{(i-1)} - s_{i-1} a_{n-i+1} \ , \\ & d_i^{(i)} = d_i^{(i-1)} - t_{i-1} d_{i-1}^{(i-1)} \ , \ d_{i+1}^{(i)} = d_{i+1}^{(i-1)} - t_{i-1} d_{i-1}^{(i-1)} \\ & d_{n-i+1}^{(i)} = d_{n-i+1}^{(i-1)} - v_{i-1} d_{n-i+2} \quad \text{and} \quad d_{n-i}^{(i)} = d_{n-i}^{(i-1)} - s_{i-1} d_{n-i+2}^{(i-1)} . \end{aligned}\right\} \qquad (4.4.47b)$$

Step 3 The central values of the solution vector are given, for n even, by

$$u_q = (d_q^{(q)} b_{n-q+1}^{(q)} - d_{n-q+1}^{(q)} a_q^{(q)}) / (b_q^{(q)} b_{n-q+1}^{(q)} - a_q^{(q)} a_{n-q}^{(q)})$$

and $$u_{n-q+1} = (b_q^{(q)} d_{n-q+1}^{(q)} - a_{n-q}^{(q)} d_q^{(q)}) / (b_q^{(q)} b_{n-q+1}^{(q)} - a_q^{(q)} a_{n-q}^{(q)}) \quad ; \qquad (4.4.48a)$$

and for n odd, by solving the (3×3) linear system,

$$\begin{pmatrix} b_q^{(q)} & a_q^{(q)} & c_q \\ a_q^{(q)} & b_{q+1}^{(q)} & a_{q+1}^{(q)} \\ c_{n-q} & a_{n-q}^{(q)} & b_{n-q+1}^{(q)} \end{pmatrix} \begin{pmatrix} u_q \\ u_{q+1} \\ u_{n-q+1} \end{pmatrix} = \begin{pmatrix} d_q^{(q)} \\ d_{q+1}^{(q)} \\ d_{n-q+1}^{(q)} \end{pmatrix} \qquad (4.4.48b)$$

Step 4 Finally, the other elements of the solution vector are obtained from the recurrence relations,

$$\left.\begin{aligned} u_i &= d_i^{(i)} - t_i u_{i+1} - r_i u_{i+2} \\ \text{and} \quad u_{n-i+1} &= d_{n-i+1}^{(i)} - v_i u_{n-i} - s_i u_{n-i-1} \end{aligned}\right\} \; i=q-1,q-2,\ldots,1. \qquad (4.4.49)$$

The reduction step (2) of this algorithm requires the order of 5n multiplications, 5n additions and 2n divisions; and a further 2n multiplications, 2n additions are required for the back substitution, giving altogether the operational count of 7n multiplications, 2n divisions and 7n additions; and for subsequent solutions, the order of only 4n multiplications and 4n additions.

This is again an improvement over the Conte and Dames algorithm (4.3). Moreover in the reduction process the $b^{(i)}, a_i^{(i)}$ and $d_i^{(i)}$ terms overwrite the diagonal, subdiagonal and right hand side vectors. There is therefore a further gain of two n-component vectors of storage over the Conte and Dames algorithm.

## 4.5 PARALLEL IMPLEMENTATION OF THE RECURSIVE POINT PARTITIONING (R.P.P.) ALGORITHM

Most conventional tridiagonal solvers (e.g. the Thomas algorithms (4.1) would not normally run faster on a parallel machine which has more than one processor because of the highly serial nature of their computation. However, it is easily observed that each of the recursive point partitioning algorithms discussed in section (4.4) has a considerable measure of implicit parallelism which arises from the parallel nature of the applied recursive reduction process at both ends of the matrix system. Thus, by exploiting the parallelism of the R.P.P. algorithms in a parallel program run on a parallel machine, a substantial improvement in the running time of these algorithms can be achieved.

First, we mention briefly some properties common to all parallel programs. In general, a typical segment of a parallel program has a graphical representation of the form shown in Figure (4.1) where 'left before right' precedence holds, so that the completion of the execution of the segment of

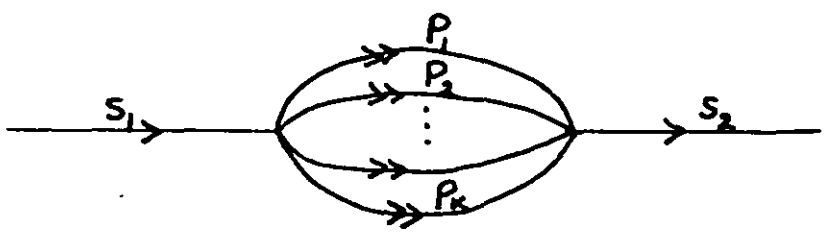



FIGURE 4.1

code $S_1$ must preceed the start of $P_i$ (i=1,2,...,k); and the completion of all the $P_i$ must precede the start of $S_2$. The $P_i$ themselves have no precedence relationship and so can be executed concurrently. The following points are important in the implementation of this structure.

(i) Each code path segment ($S_1$,$P_1$,$P_2$,...) must be executed by only one processor so that when one processor takes up a path all others must be 'informed' and locked out of this path. Similarly, on completion of the path all processors must be informed so that the next set of

available paths are freed to them,

(ii) Only after the preceding paths have been completed can a given path be started, i.e., $P_1,P_2,\ldots,P_k$ must be completed before starting $S_2$.

(iii) Variables used by $P_1,P_2,\ldots,P_k$ but defined in $S_1$ must be made available to all processors and all values set by $P_1,P_2,\ldots,P_n$ and used in $S_2$ must be made available to the processor that executes $S_2$. Likewise for variables used by $S_2$ but defined in $S_1$.

We now consider specifically a parallel program which is to be run on the Loughborough University Interdata parallel computer with two model 70 processors sharing a 32kb block of common core memory, with each also having a 32kb of private memory. Special software (Barlow et al (1977)) locally developed is available which enables the programmer to effect the necessary control, allocation and lock-out of resources as outlined in (i),(ii) and (iii) above during the execution of segments of the parallel program. For example, the 'FORK' and 'JOIN' constructs inserted in a standard Fortran program enables the two Interdata processors to 'fork', i.e., work on separate sections of code independently and in parallel; and then 'join' when only one processor is necessary to work on the program after collecting results from the section done in parallel. 'DOPAR' and 'PAREND' macros also perform a similar function. The "GETRES(I)" subroutine offers lock-out mechanism by permitting an exclusive use of a resource I and the "PUTRES(I)" relinquishes ownership of the resource I.

In order to program, for example, the TRPP algorithm (4.8) (for the solution of the general tridiagonal matrix system) in parallel, we arrange the order of calculation of the intermediate quantities and solution vector components in the form given in Figure (4.2) in order to take advantage of the implicit parallelism of the algorithm. A similar arrangement is also applicable to the parallel programming of the GQRPP algorithm (4.11) for the general symmetric quindiagonal matrix system.

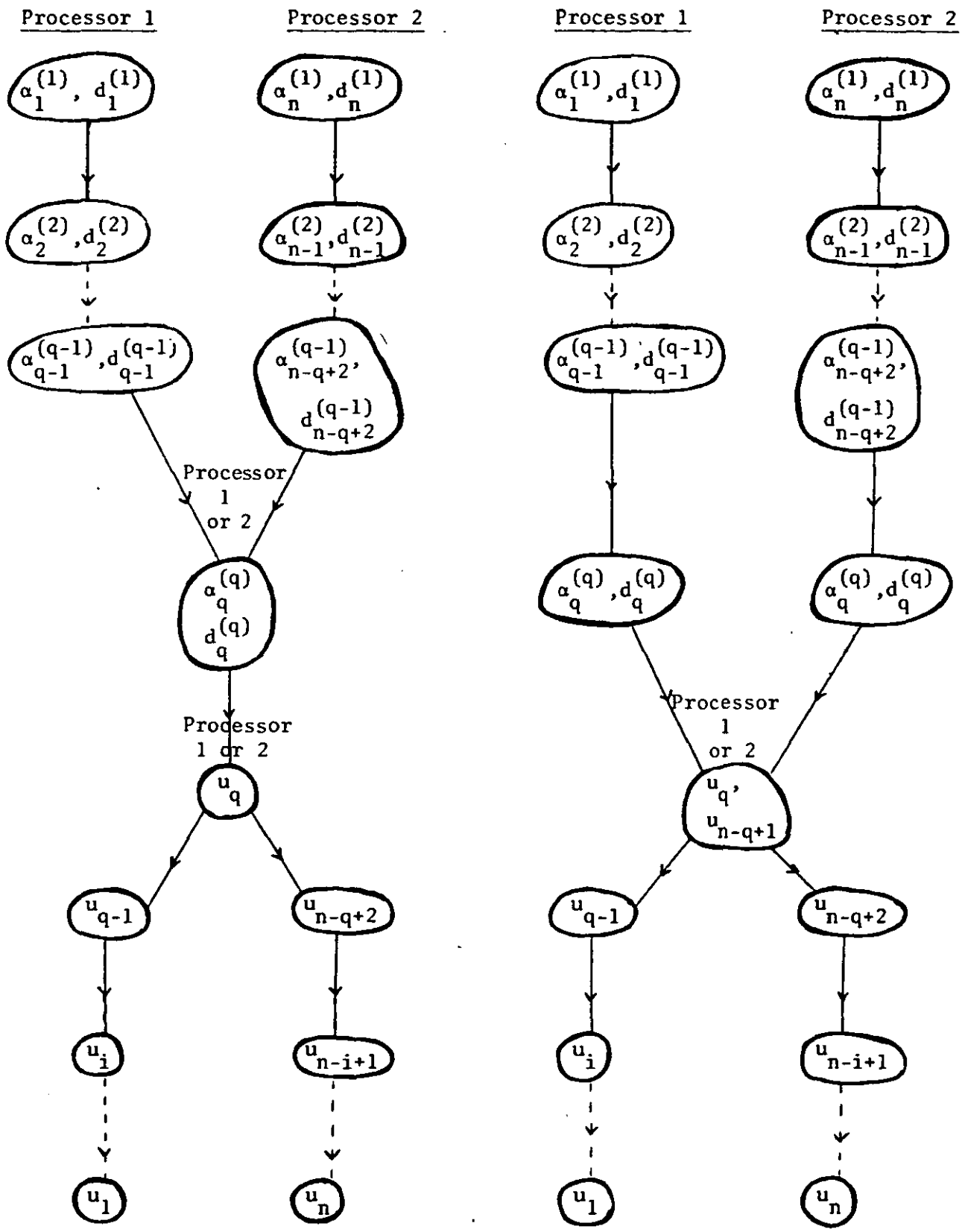



FIGURE 4.2

Order of the Parallel Implementation of the R.P.P. Algorithm (4.8)

The parallel programs for the TRPP and GQRPP algorithms are given in Appendix I as Program 12 and Program 13 respectively. They are written in standard Fortran (except that the parallel processing constructs, FORK and JOIN, GETRES(I) and PUTRES(I), used for parallel controls are inserted) and then run on the Loughborough University Interdata parallel computer.

The two parallel programs were test run on matrix systems of varying order. In order to assess the performance of the parallel programs, timing results of runs on matrix systems of order 650 were taken. For example, for each program, the times for the programs to run sequentially without the parallel constructs were noted. Then, the times for each program to run in parallel mode with the insertion of parallel constructs were also noted. These and other relevant timing results taken are shown in Table (A2.1) of Appendix II. From these experimental results, we compute the estimates of the slow down or efficiency losses (due to memory clashing, parallel control contentions, etc.) and the speed-up (i.e., the amount by which a program runs faster in parallel mode than it does in serial mode) of the two parallel programs. The details of the calculation of the speed up and the various overheads for the two parallel R.P.P. programs (for the solution of the tridiagonal and quindiagonal matrix systems of order 650) are given in Appendix II. Below, we summarise the results of the calculations in Table (4.1), from which it can be seen that the speed up of the parallel program (12) is 175% and that of the parallel program (13) is 182%.

The Overheads and Speed-up of the R.P.P. Tridiagonal and Quindiagonal Parallel Programs for Systems of Order n=650

| Matrix Systems and Program | Shared Memory (Static) Overhead | Parallel Control Overheads | | Shared Memory Access Overhead | Others (e.g. time in scheduler etc.) | Speed-up* of parallel program |
|---|---|---|---|---|---|---|
| | | (Static) | (Dynamic) | (Dynamic) | | |
| 1. Tridiagonal System (Program 12) | 3.1% | 3.1% | 1.2% | ~ 0.0% | 17.5% | 175% |
| 2. Quindiagonal System (Program 13) | 3.2% | 2.2% | 1.7% | ~ 0.0% | 10.9% | 182% |

TABLE 4.1

*The time taken by the program to run in a parallel mode with two processors as a % of the time taken by the program to run in a serial mode with only one processor.*

## 4.6 ROUNDING ERROR ANALYSIS FOR THE RECURSIVE POINT PARTITIONING METHOD

In the formulation of an algorithmic process for the solution of a matrix system on a computer (which does not do an exact arithmetic), it is essential to assess the cumulative effect of rounding errors that are made during the course of the implementation of such a process in order to guarantee that the solution we derive is always stable.

Here we consider the rounding error analysis for the newly introduced point partitioning methods. The analysis of the algorithm for the tridiagonal case only will be discussed since a similar approach would apply for the quin-diagonal and other matrix systems of wider bands.

In our analysis, we shall assume a floating-point computer arithmetic in which each number x (say) is represented internally in the form,

$$x = m.2^{E} \quad , \qquad 0.5 \leq |m| \leq 1 \ ,$$

where E, the exponent, is bounded by the binary word length of the given computer registers.

Further, in our analysis, we shall follow the backward analysis scheme developed by Wilkinson (1963), and hence use the notation $fl(x.y)$ to denote the computed result of multiplying together two floating point numbers x and y; and so on. Then in general we have the following exact mathematical relationships,

$$\left.\begin{aligned} fl(x\pm y) &= (x\pm y)(1+\varepsilon_1) \quad , \\ fl(x.y) &= xy(1+\varepsilon_2) \quad , \\ \text{and} \qquad fl(x/y) &= x/y(1+\varepsilon_3) \quad . \end{aligned}\right\} \tag{4.6.1}$$

Each $\varepsilon_i$, which represents the rounding error associated with the respective arithmetic operation, is some value of $\varepsilon$ where $|\varepsilon| < 2^{-k}$ and k (=24 for the I.C.L. 1904S computer) is the number of binary digits allocated to the mantissa of the floating point number in the computer.

Now, we consider the TRPP algorithm (4.8). First, we examine the recurrence reduction formula (4.4.11) i.e.,

$$\alpha_i^{(i)} = b_i - \frac{a_{i-1}c_{i-1}}{\alpha_i^{(i-1)}}\ ,\ i=2,3,\ldots,q\ . \qquad (4.6.2)$$

Since $\alpha_i^{(1)} = b_i$ $(i=1,2,\ldots,n)$, there is no rounding error in obtaining $\alpha_i^{(1)}$.

Next, by considering $\alpha_2^{(2)}$ we have,

$$fl(\alpha_2^{(2)}) = fl(b_2 - \frac{a_1c_1}{\alpha_1^{(1)}}) = [b_2 - \frac{a_1c_1}{\alpha_1^{(1)}}(1+\epsilon_2)(1+\epsilon_3)](1+\epsilon_1)$$

and hence on neglecting second order quantities in $\epsilon$, we have

$$|fl(\alpha_2^{(2)})-\alpha_2^{(2)}| = b_2\epsilon_1 + \frac{a_1c_1}{\alpha_1^{(1)}}(\epsilon_1+\epsilon_2+\epsilon_3)$$

$$\leq (b_2+\frac{3a_1c_1}{\alpha_1^{(1)}})\epsilon = g_2\epsilon = w_2 \qquad (4.6.3)$$

where $\epsilon > |\epsilon_1|, |\epsilon_2|, |\epsilon_3|$;

$$g_2 = (b_2+\frac{3a_1c_1}{b_1^{(1)}})$$

and $w_2$ denotes the upper bound of the rounding error associated with the determination of $\alpha_2^{(2)}$.

Further, we consider $\alpha_3^{(3)} = b_3 - \frac{a_2c_2}{\alpha_2^{(2)}}$ and hence

$$fl(\alpha_3^{(3)}) = \left[b_3 - \frac{a_2c_2(1+\epsilon_2)(1+\epsilon_3)}{\alpha_2^{(2)}(1+w_2)}\right](1+\epsilon_1)\ ,$$

from which we obtain,

$$fl(\alpha_3^{(3)})-\alpha_3^{(3)} = [b_3\epsilon_1 + \frac{a_2c_2}{\alpha_2^{(2)}}(\epsilon_1+\epsilon_2+\epsilon_3-w_2)]$$

and hence,

$$|fl(\alpha_3^{(3)})-\alpha_3^{(3)}| \leq b_3\epsilon_1 + \frac{a_2c_2}{\alpha_2^{(2)}}(\epsilon_1+\epsilon_2+\epsilon_3) + \frac{a_2c_2}{\alpha_2^{(2)}}w_2\ .$$

On substituting for $w_2$ we have the result,

$$|fl(\alpha_3^{(3)})-\alpha_3^{(3)}| \leq (b_3 + \frac{3a_2c_2}{\alpha_2^{(2)}})\epsilon + \frac{a_2c_2}{\alpha_2^{(2)}}g_2\epsilon$$

$$= (g_3 + \frac{a_2c_2g_2}{\alpha_2^{(2)}})\epsilon = w_3 \qquad (4.6.5)$$

where $g_3 = b_3 + \frac{3a_2c_2}{\alpha_2^{(2)}}$

and $w_3$ denotes the upper bound of the rounding error associated with $\alpha_3^{(3)}$.

A continuation of this analysis gives,

$$|fl(\alpha_4^{(4)})-\alpha_4^{(4)}| \leq [g_4 + \frac{a_3c_3}{\alpha_3^{(3)}}(g_3 + \frac{a_2c_2}{\alpha_2^{(2)}} g_2)|\ \varepsilon$$

$$= [g_4 + \frac{a_3c_3}{\alpha_3^{(3)}} g_3 + \frac{a_3c_3}{\alpha_3^{(3)}} \cdot \frac{a_2c_2}{\alpha_2^{(2)}} g_2]\varepsilon = w_4$$

where $g_4 = b_4 - \frac{3a_3c_3}{\alpha_3^{(3)}}$ .

Similarly,

$$|fl(\alpha_5^{(5)}) - \alpha_5^{(5)}| \leq [g_5 + \frac{a_4c_4}{\alpha_4^{(4)}} g_4 + (\frac{a_4c_4}{\alpha_4^{(4)}} \cdot \frac{a_3c_3}{\alpha_3^{(3)}})g_3 +$$

$$\frac{a_4c_4}{\alpha_4^{(4)}} \cdot \frac{a_3c_3}{\alpha_3^{(3)}} \cdot \frac{a_2c_2}{\alpha_2^{(2)}} g_2]\varepsilon = w_5$$

In general, if we define the following quantities,

$$\left.\begin{aligned} g = \max_i g_i &= \max_i (b_i + \frac{3a_{i-1}c_{i-1}}{\alpha_{i-1}^{(i-1)}}) \\ \text{and} \quad t = \max_i t_i &= \max_i (\frac{c_ia_i}{\alpha_i^{(i)}}) \ , \end{aligned}\right\} \qquad (4.6.6a)$$

then,

$$|fl(\alpha_i^{(i)})-\alpha_i^{(i)}| \leq [g_i + t_{i-1}g_{i-1} + t_{i-1}t_{i-2}g_{i-2} + \cdots$$

$$+ (t_{i-1}t_{i-2}\cdots t_2)g_2] \ ,$$

$$\leq [1+t+t^2\ldots+t^{i-2}]g\varepsilon \ . \qquad (4.6.6b)$$

Hence, $|fl(\alpha_i^{(i)})-\alpha_i^{(i)}| \leq \frac{g\varepsilon}{1-t}$ $(=w_i)$ if $t<1$ (4.6.7)

where $w_i$ denotes the upper bound of the rounding error associated with each $\alpha_i^{(i)}$, for $i=2,3,\ldots,q$ $(=n/2$ n even; $(n+1)/2$, n odd).

By a similar analysis, and commencing from the rear of the system with $\alpha_n^{(1)}$ having no rounding error, it can be easily shown that the upper bound of the rounding error associated with $\alpha_{n-i+1}^{(i)} = b_{n-i+1} - \frac{a_{n-i+1}c_{n-i+1}}{\alpha_{n-i+2}^{(i)}}$, $i=2,3,\ldots,q$ is also bounded. Further by using a similar method of analysis as above, it can be shown that the recurrence relations for the right hand side vector, i.e.,

$$\left.\begin{aligned} d_i^{(i)} &= d_i^{(i-1)} - \frac{a_{i-1}d_{i-1}^{(i-1)}}{\alpha_{i-1}^{(i-1)}} \\ \text{and} \quad d_{n-i+1}^{(i)} &= d_{n-i+1}^{(i-1)} - \frac{c_{n-i+1}d_{n-i+2}^{(i-1)}}{\alpha_{n-i+2}^{(i)}} \end{aligned}\right\} \quad i=2,3,\ldots,q,$$

have bounded rounding errors $\rho_i$ (say) associated with each $d_i^{(i)}$.

Next, we investigate the rounding error in the backward substition phase of the algorithm and consider first the central values of the solution vector, for n odd, i.e.,

$$u_q = d_q^{(q)}/\alpha_q^{(q)} \quad , \quad q=(n+1)/2 \qquad (n \text{ odd})$$

Then,

$$fl(u_q) = \frac{d_q^{(q)}(1+\rho_q)(1+\varepsilon_3)}{\alpha_q^{(q)}(1+w_q)} \simeq \frac{d_q^{(q)}}{\alpha_q^{(q)}}(1+\rho_q+\varepsilon_3-w_q)$$

where $w_q,\rho_q$ denote the rounding error bounds associated with $\alpha_q^{(q)}$ and $d_q^{(q)}$ respectively.

Hence, if we define,

$$\bar{\varepsilon} = \max(w_i,\rho_i,\varepsilon_1,\varepsilon_2,\varepsilon_3) \qquad (4.6.8)$$

then, $|fl(u_q)-u_q| \lesssim u_q(\varepsilon_3+\rho_q+w_q)$

$$\lesssim u_q(3\bar{\varepsilon}) \; (= u_q\sigma_q) \qquad (4.6.9)$$

where $\sigma_q=3\bar{\varepsilon}$ is the bound on the rounding error associated with $u_q$.

For n even, it can be similarly shown that the rounding error $\sigma_q$ associated with the central value $u_q$ of the solution vector, i.e., $u_q=(d_q^{(q)}\alpha_{n-q+1}^{(q)}-d_{n-q+1}^{(q)}c_q)/(\alpha_{n-q+1}^{(q)}\alpha_q^{(q)}-a_{n-q}c_q)$, is also bounded.

Next, we consider the back substitution formula for the remaining elements of the solution vector, i.e.,

$$u_i = (d_i^{(i)}-c_iu_{i+1})/\alpha_i^{(i)}, \quad i=q-1,q-2,\ldots 1.$$

The quantity $fl(u_{q-1})$ is given by,

$$fl(u_{q-1}) = [d_{q-1}^{(q-1)}(1+\rho_{q-1})-c_{q-1}u_q(1+\sigma_q)(1+\varepsilon_2)]\,\frac{(1+\varepsilon_1)(1+\varepsilon_3)}{\alpha_{q-1}^{(q-1)}(1+w_{q-1})},$$

and hence on neglecting the second order quantities of the error terms, we have,

$$fl(u_{q-1})-u_{q-1} = \frac{d_q^{(q-1)}}{\alpha_{q-1}^{(q-1)}}(\varepsilon_1+\varepsilon_3-w_{q-1}) - \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}}(1+\varepsilon_1+\varepsilon_2+\varepsilon_3-w_{q-1}+\sigma_q)$$

which, on using the relation in (4.6.8) and substituting for $u_{q-1}$ gives,

$$|fl(u_{q-1})-u_{q-1}| \leq (\frac{d_{q-1}^{(q-1)}}{\alpha_q^{(q-1)}} + \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}})4\bar{\varepsilon} + \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}}\sigma_q$$

and on substituting $\sigma_q=3\bar{\varepsilon}$ from (4.6.9) gives,

$$|fl(u_{q-1})-u_{q-1}| \leq [g_{q-1} + g_q\frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}}]4\bar{\varepsilon} \quad (=\sigma_{q-1}) , \tag{4.6.10}$$

where

$$g_q \equiv 1, \qquad g_{q-1} = \frac{d_{q-1}^{(q-1)}}{\alpha_q^{(q-1)}} + \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}}$$

and $\sigma_{q-1}$ denotes the upper bound of the rounding error associated with $u_{q-1}$.

Further, we consider $fl(u_{q-2})$ which is given by,

$$fl(u_{q-2}) = [d_{q-2}^{(q-2)}(1+\rho_{q-2})-c_{q-2}u_{q-1}(1+\sigma_{q-1})(1+\varepsilon_2)]\frac{(1+\varepsilon_1)(1+\varepsilon_3)}{\alpha_{q-2}^{(q-2)}(1+w_{q-1})}$$

from which, after some simplifications, we obtain the result,

$$|fl(u_{q-2})-u_{q-2}| \leq \frac{d_{q-2}^{(q-2)}}{\alpha_{q-2}^{(q-2)}}(\varepsilon_1+\varepsilon_3+w_{q-1}+\rho_{q-2}) + \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}}(\varepsilon_1+\varepsilon_2+\varepsilon_3+w_{q-1})$$

$$+ \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}}\sigma_{q-1}$$

which, on using (4.6.8) and substituting for $\sigma_{q-1}$ from (4.6.10) gives,

$$|fl(u_{q-2})-u_{q-2}| \leq 4\bar{\varepsilon}g_{q-2} + \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}}(g_{q-1} + \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}})4\bar{\varepsilon}$$

$$\leq \sigma_{q-2} , \tag{4.6.11a}$$

where

$$g_{q-2} = (\frac{d_{q-2}^{(q-2)}}{\alpha_{q-2}^{(q-2)}} + \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}}) . \tag{4.6.11b}$$

Further, by considering $u_{q-3}$, we obtain,

$$|fl(u_{q-3})-u_{q-3}| = [g_{q-3}+(\frac{c_{q-3}u_{q-2}}{\alpha_{q-3}^{(q-3)}})g_{q-2} + (\frac{c_{q-3}u_{q-2}}{\alpha_{q-3}^{(q-3)}} \cdot \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}})g_{q-1}$$

$$+(\frac{c_{q-3}u_{q-2}}{\alpha_{q-3}^{(q-3)}} \cdot \frac{c_{q-2}u_{q-1}}{\alpha_{q-2}^{(q-2)}} \cdot \frac{c_{q-1}u_q}{\alpha_{q-1}^{(q-1)}})\, g_q]4\bar{\varepsilon} \tag{4.6.12a}$$

where $$g_{q-3} = \left(\frac{d_{q-3}}{\alpha_{q-3}^{(q-3)}} + \frac{c_{q-3}u_{q-2}}{\alpha_{q-3}^{(q-3)}}\right) \quad . \tag{4.6.12b}$$

If we now define $z_i = \frac{c_i u_{i+1}}{\alpha_i^{(i)}}$ and use the definitions in (4.6.6b) then,

$$|fl(u_{q-3})-u_{q-3}| = [g_{q-3}+z_{q-3}g_{q-2}+z_{q-3}z_{q-2}g_{q-1}+z_{q-3}z_{q-2}z_{q-1}g_q]4\bar{\varepsilon} \tag{4.6.13}$$

In general, if the above analysis is pursued, we obtain the final result for $u_1$ as,

$$|fl(u_1)-u_1| \leq [g_1+z_1g_2+z_1z_2g_3+z_1z_2z_3g_4+\ldots+z_1z_2\ldots z_{q-1}g_q]4\bar{\varepsilon}. \tag{4.6.14}$$

On defining,

$$\bar{g} = \max_i(g_i) \quad \text{and} \quad \bar{z} = \max_i(z_i) \tag{4.6.15}$$

where $g_i = \frac{d_i}{\alpha_i^{(i)}} + \frac{c_i u_{i+1}}{\alpha_i^{(i)}}$ and $z_i = \frac{c_i u_i}{\alpha_i^{(i)}}$

then, we obtain,

$$\begin{aligned} |fl(u_1)-u_1| &\leq (1+z+z^2+\ldots+z^{q-1})4\bar{\varepsilon}\bar{g} \\ &\leq \frac{4\bar{\varepsilon}\bar{g}(1-\bar{z}^{q-1})}{1-\bar{z}} \end{aligned} \tag{4.6.16}$$

which is finite for a finite value of q.

In a similar way, the rounding errors which occur in the evaluation of the lower half of the solution vector, $u_{n-i+1}$ (i=q,q-1,...1), can also be shown to be bounded.

Thus, it has been shown that the rounding errors incurred in the solution of a tridiagonal matrix system using the recursive point partitioning algorithmic scheme is bounded for a finite value of q (and hence n).

## 4.7 THE RECURSIVE POINT DECOUPLING (R.P.D.) ALGORITHM

In the development of the recursive point partitioning algorithms of section (4.4) the 'one-line at a time' partitioning scheme was adopted. It is possible, however, to vary the size of the block structure (by using 'two-lines at a time', 'three-lines at a time' etc.) in the partitioning process. The 'two-line' partitioning scheme was investigated for both the tridiagonal and quindiagonal matrix systems. It was found that the algorithmic solution derived by adopting such a partitioning strategy involves almost twice as much arithmetic work (under a sequential processing system) as does the corresponding 'one-line' partitioning method. However, there is an increased implicit parallelism of the algorithm derived compared to that of the 'one-line' method.

A similar result was also obtained on investigating the '3-line' partitioning scheme. Thus, it was necessary to examine the extreme case in which for an n order system, the 'n/2-line' partitioning strategy is employed in the solution of the given tridiagonal matrix system.

We consider, for simplicity, the symmetric tridiagonal matrix system of order n ($=2^m$, where m is any positive integer) given by,

$$\underline{Au} = \underline{d} \tag{4.7.1}$$

or in matrix notation as,

$$\begin{bmatrix} \lambda-\beta_1 & 1 & & & & & & \\ 1 & \lambda & 1 & & & & & \\ & 1 & \lambda & 1 & & & & \\ & & & & & & & \\ & & & 1 & \lambda & 1 & & \\ & & & & 1 & \lambda & 1 & \\ & & & & & 1 & \lambda-\beta_n & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n/2} \\ u_{\frac{n}{2}+1} \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} d_1-\phi_1 \\ d_2 \\ \vdots \\ d_{n/2} \\ d_{\frac{n}{2}+1} \\ \vdots \\ d_{n-1} \\ d_n-\phi_n \end{bmatrix} \tag{4.7.2}$$

where

$$\beta_1 = \beta_n = \phi_1 = \phi_n = 0$$

are introduced for convenience, and reference as to their purpose will be evident in what follows. We also assume that the matrix A is diagonally dominant.

The system (4.7.2) can be partitioned into two separate halves, as indicated, to give the block form,

$$\begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix} \begin{pmatrix} \underline{\tilde{u}}_1 \\ \underline{\tilde{u}}_2 \end{pmatrix} = \begin{pmatrix} \underline{\tilde{d}}_1 \\ \underline{\tilde{d}}_2 \end{pmatrix} \tag{4.7.3}$$

from which we obtain the following:

$$(A_1 - B_1 A_2^{-1} B_2)\underline{\tilde{u}}_1 = \underline{\tilde{d}}_1 - B_1 A_2^{-1} \underline{\tilde{d}}_2 \tag{4.7.4}$$

and

$$(A_2 - B_2 A_1^{-1} B_1)\underline{\tilde{u}}_2 = \underline{\tilde{d}}_2 - B_2 A_1^{-1} \underline{\tilde{d}}_1 \ . \tag{4.7.5}$$

Both (4.7.4) and (4.7.5) form a decoupled matrix system of the form,

$$\left(\begin{array}{c|c} A_1^{(1)} & 0 \\ \hline 0 & A_2^{(1)} \end{array}\right) \begin{pmatrix} \underline{\tilde{u}}_1 \\ \underline{\tilde{u}}_2 \end{pmatrix} = \begin{pmatrix} \underline{\tilde{d}}_1^{(1)} \\ \underline{\tilde{d}}_2^{(1)} \end{pmatrix} \tag{4.7.6}$$

where

$$\left.\begin{aligned} A_1^{(1)} &= A_1 - B_1 A_2^{-1} B_2 \ , \\ A_2^{(1)} &= A_2 - B_2 A_1^{-1} B_1 \ , \\ \underline{\tilde{d}}_1^{(1)} &= \underline{\tilde{d}}_1 - B_1 A_2^{-1} \underline{\tilde{d}}_2 \ , \\ \text{and} \quad \underline{\tilde{d}}_2^{(1)} &= \underline{\tilde{d}}_2 - B_2 A_1^{-1} \underline{\tilde{d}}_1 \ . \end{aligned}\right\} \tag{4.7.7}$$

We now proceed to show that the decoupled submatrices $A_1^{(1)}$ and $A_2^{(1)}$ are tridiagonals and of the same structure as the original matrix A.

The submatrices $A_1$ and $A_2$ are tridiagonals of order n/2, but their inverses are full.

By definition,

$$(A_1^{-1})_{i,j} = \frac{\gamma_{j,i}}{|A_1|} \ , \quad i,j=1,2,\ldots,n/2 \ , \tag{4.7.8}$$

where $\gamma_{j,i}$ denotes the (i,j) cofactor (signed minor) of $A_1$ and $|A_1|$ is the determinant of $A_1$.

Similarly

$$(A_2^{-1})_{i,j} = \frac{\alpha_{j,i}}{|A_2|} \ , \quad i,j=1,2,\ldots,n/2 \ , \tag{4.7.9}$$

where $\alpha_{j,i}$ denotes the (i,j) cofactor of $A_2$.

It is therefore easily shown by simple multiplication that the matrices $B_1 A_2^{-1} B_2$ and $B_2 A_1^{-1} B_1$ are matrices of order n/2 given by,

$$B_1A_2^{-1}B_2 = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & \ddots & \\ & & & \beta_{n/2} \end{bmatrix} \quad \text{and} \quad B_2A_1^{-1}B_1 = \begin{bmatrix} \beta_{\frac{n}{2}+1} & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} \tag{4.7.10}$$

where,

$$\beta_{n/2} = \frac{\alpha_{1,1}}{|A_2|} \quad \text{and} \quad \beta_{\frac{n}{2}+1} = \frac{\gamma_{n/2,n/2}}{|A_1|} \; . \tag{4.7.11}$$

It can be similarly shown that $B_1A_2^{-1}\tilde{d}_2$ and $B_2A_1^{-1}\tilde{d}_2$ are vectors of length n/2 given by,

$$B_1A_2^{-1}\tilde{d}_2 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \phi_{n/2} \end{bmatrix} \quad , \text{ and } B_2A_1^{-1}\tilde{d}_2 = \begin{bmatrix} \phi_{\frac{n}{2}+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{4.7.12}$$

where

$$\phi_{n/2} = \frac{1}{|A_2|} \sum_{i=1}^{n/2} \alpha_{i,1} d_{i+n/2} \; , \tag{4.7.13a}$$

and

$$\phi_{\frac{n}{2}+1} = \frac{1}{|A_1|} \sum_{i=1}^{n/2} \gamma_{i,n/2} d_i \; . \tag{4.7.13b}$$

We shall take advantage of the special form of the matrices $A_1$ and $A_2$ to derive alternative expressions for the quantities $\alpha_{i,1}$ and $\gamma_{i,n/2}$ in terms of the Sturm sequence of polynomials $\{P_i\}$, (i=0,1,2,...) obtained by a simple Laplace expansion of the leading principal minors of the tridiagonal matrix of the form,

$$C = \begin{bmatrix} \lambda & 1 & & & \\ 1 & \lambda & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & 1 & \lambda \end{bmatrix}_{(n/2 \times n/2)}$$

Thus,

$$\left.\begin{aligned} P_0 &= 1 \\ P_1 &= \lambda \\ P_i &= \lambda P_{i-1} - P_{i-2} \quad (i=2,3,\ldots) \end{aligned}\right\} \tag{4.7.14}$$

By making use of the structure of $A_1$ and $A_2$ as shown in (4.7.2), it is now possible to express the determinants $|A_1|, |A_2|$ as a function of $P_i$ $(i=0,1,\ldots,n/2-1)$,

$$\left.\begin{aligned} \text{i.e.,} \qquad |A_1| &= (\lambda-\beta_1)P_{n/2} - P_{\frac{n}{2}-2} \quad , \\ \text{and} \qquad |A_2| &= (\lambda-\beta_n)P_{\frac{n}{2}-1} - P_{\frac{n}{2}-2} \quad . \end{aligned}\right\} \tag{4.7.15}$$

Also, it is easily verified that the set of cofactors of $A_1$, i.e., $\gamma_{i,n/2}$ $(i=1,2,\ldots,n/2)$ is given by the relationships,

$$\left.\begin{aligned} \gamma_{1,n/2} &= -1 \\ \gamma_{2,n/2} &= (-1)^2(\lambda-\beta_1) \\ \text{and} \qquad \gamma_{i,n/2} &= (-1)^i[P_{i-2}(\lambda-\beta_1)-P_{i-3}] \quad , \quad i=3,4,\ldots,n/2 \quad . \end{aligned}\right\} \tag{4.7.16}$$

Similarly, the cofactors of $A_2$, i.e., $\alpha_{i,1}$ can be expressed as,

$$\left.\begin{aligned} \alpha_{1,1} &= (\lambda-\beta_n)P_{\frac{n}{2}-2} - P_{\frac{n}{2}-3} \quad , \\ \alpha_{2,1} &= (-1)^{2+1}[(\lambda-\beta_n)P_{\frac{n}{2}-3} - P_{\frac{n}{2}-4}] \quad , \\ \alpha_{i,1} &= (-1)^{1+i}[(\lambda-\beta_n)P_{\frac{n}{2}-i-1} - P_{\frac{n}{2}-i-2}] \quad , \quad i=3,4,\ldots,\tfrac{n}{2}-2 \\ \alpha_{\frac{n}{2}-1,1} &= (\lambda-\beta_n) \\ \text{and} \qquad \alpha_{\frac{n}{2},1} &= -1. \end{aligned}\right\} \tag{4.7.17}$$

Hence, the quantities $\beta_{n/2}, \beta_{\frac{n}{2}+1}, \phi_{n/2}, \phi_{\frac{n}{2}+1}$ in (4.7.11)-(4.7.14) can be expressed, on substituting for $|A_1|, |A_2|$, $\gamma_{n/2,n/2}$ and $\alpha_{n/2,1}$ from (4.7.15)-(4.7.17), as follows:

$$\left.\begin{aligned} \beta_{n/2} &= \frac{(\lambda-\beta_n)P_{\frac{n}{2}-2} - P_{\frac{n}{2}-3}}{(\lambda-\beta_n)P_{\frac{n}{2}-1} - P_{\frac{n}{2}-2}} \\ \beta_{\frac{n}{2}+1} &= \frac{(\lambda-\beta_1)P_{\frac{n}{2}-2} - P_{\frac{n}{2}-3}}{(\lambda-\beta_1)P_{\frac{n}{2}-1} - P_{\frac{n}{2}-2}} \end{aligned}\right\}$$

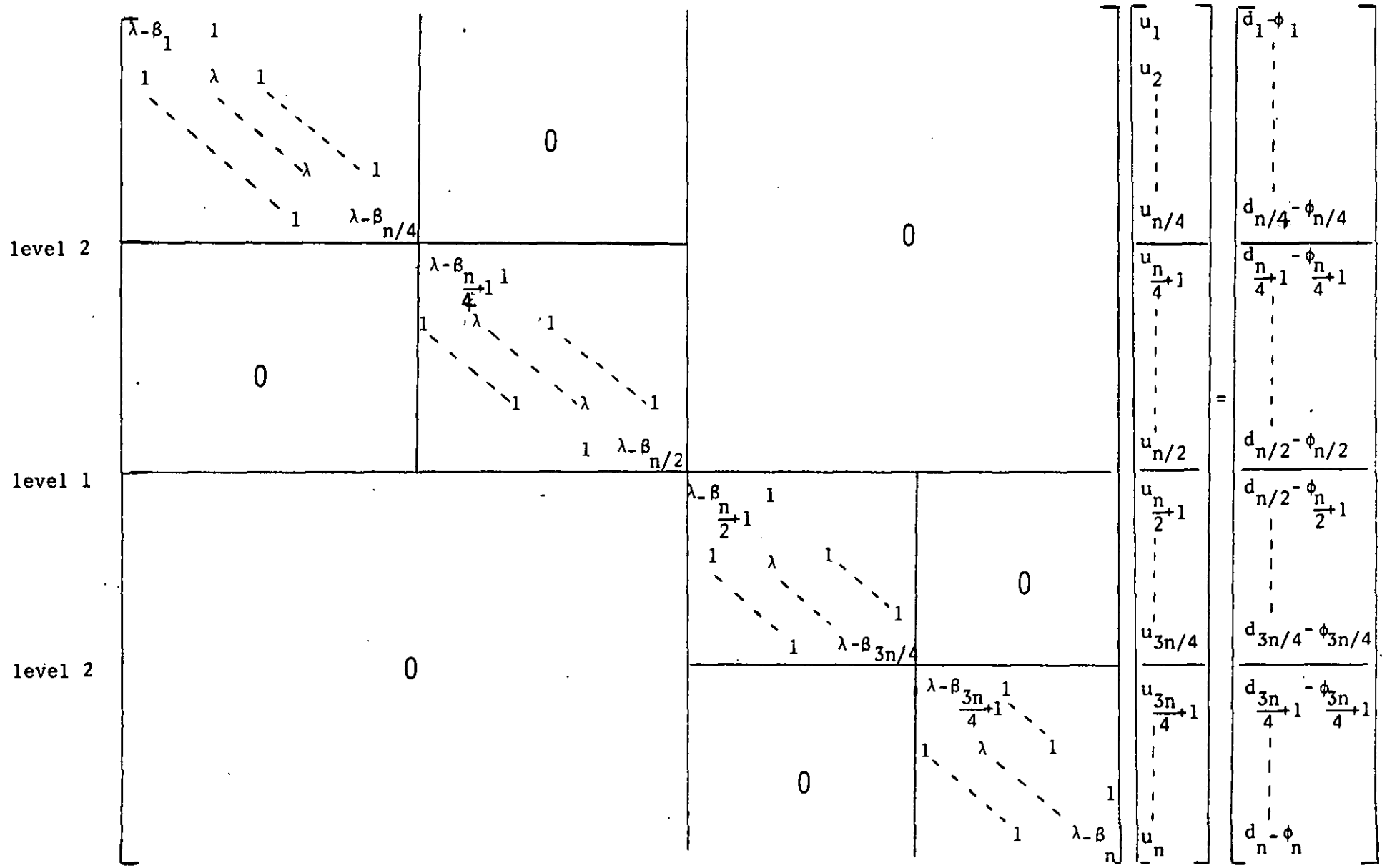
$$\left.\begin{aligned} \phi_{n/2} &= \frac{\sum_{i=1}^{n/2} \alpha_{i,1} d_{i+n/2}}{(\lambda-\beta_n)P_{\frac{n}{2}-1} - P_{\frac{n}{2}-2}} \\ \text{and} \quad \phi_{\frac{n}{2}+1} &= \frac{\sum_{i}^{n/2} \gamma_{i,n/2} d_i}{(\lambda-\beta_1)P_{\frac{n}{2}-1} - P_{\frac{n}{2}-2}} \end{aligned}\right\} \qquad (4.7.18)$$

By symmetry of (4.7.2), if $\beta_1=\beta_n$, then $\beta_{n/2}=\beta_{\frac{n}{2}+1}$ which agrees with the formula obtained in (4.7.18).

Now, by a substitution of $B_1A_2^{-1}B_2$, $B_2A_1^{-1}B_1$ from (4.7.10) and $B_1A_2^{-1}\tilde{d}_2$, $B_2A_1^{-1}\tilde{d}_2$ from (4.7.12) into (4.7.7) we find that the decoupled system (4.7.6) is the same system as,

$$\left[\begin{array}{ccc|ccc} \lambda-\beta_1 & 1 & & & & \\ 1 & \lambda & 1 & & & \\ & 1 & \lambda & 1 & & 0 \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & \lambda-\beta_{\frac{n}{2}} & \\ \hline & & & & \lambda-\beta_{\frac{n}{2}+1} & 1 \\ & & & & 1 & \lambda \quad 1 \\ & 0 & & & & \ddots \quad \ddots \\ & & & & 1 & \lambda-\beta_n \end{array}\right] \left[\begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_{n/2} \\ \hline u_{\frac{n}{2}+1} \\ \vdots \\ u_n \end{array}\right] = \left[\begin{array}{c} d_1-\phi_1 \\ d_2 \\ \vdots \\ d_{n/2}-\phi_{n/2} \\ \hline d_{\frac{n}{2}+1}-\phi_{\frac{n}{2}+1} \\ \vdots \\ d_{n-1} \\ d_n+\phi_n \end{array}\right] \qquad (4.7.19)$$

where each half of the decoupled system has the same form as the original system (4.7.2) but of order n/2. In computational terms the decoupling process involves only a simple modification of the two central diagonal elements and the corresponding right-hand side vector elements at the zone of decoupling. We refer to the process of transforming the system (4.7.2) into the form (4.7.19) as level-1 decoupling. Thus, a similar transformation applied to both halves of the already decoupled sub-systems in (4.7.19) produces a level-2 decoupling of the form,


level 2
level 1
level 2
(4.7.20)

where

$$\left.\begin{aligned}
\beta_{n/4} &= \frac{(\lambda-\beta_{n/2})P_{\frac{n}{4}-2} - P_{\frac{n}{4}-3}}{(\lambda-\beta_{n/2})P_{\frac{n}{4}-1} - P_{\frac{n}{4}-2}}\,, \\
\beta_{\frac{n}{4}+1} &= \frac{(\lambda-\beta_{1})P_{\frac{n}{4}-2} - P_{\frac{n}{4}-3}}{(\lambda-\beta_{1})P_{\frac{n}{4}-1} - P_{\frac{n}{4}-2}}\,, \\
\beta_{\frac{3n}{4}} &= \frac{(\lambda-\beta_{n})P_{\frac{n}{4}-2} - P_{\frac{n}{4}-3}}{(\lambda-\beta_{n})P_{\frac{n}{4}-1} - P_{\frac{n}{4}-2}}\,, \\
\text{and}\quad \beta_{\frac{3n}{4}+1} &= \frac{(\lambda-\beta_{\frac{n}{2}+1})P_{\frac{n}{4}-2} - P_{\frac{n}{4}-3}}{(\lambda-\beta_{\frac{n}{2}+1})P_{\frac{n}{4}-1} - P_{\frac{n}{4}-2}}\,.
\end{aligned}\right\} \qquad (4.7.21)$$

Similarly, the corresponding right-hand side modifier quantities are given by,

$$\left.\begin{aligned}
\phi_{n/4} &= \frac{\sum_{i=1}^{n/4} \alpha_{i,1} d_{i+n/4}}{P_{\frac{n}{4}-1}(\lambda-\beta_{n/2}) - P_{\frac{n}{4}-2}}\,, \\
\phi_{\frac{n}{4}+1} &= \frac{\sum_{i=1}^{n/4} \gamma_{i,n/4} d_{i}}{P_{\frac{n}{4}-1}(\lambda-\beta_{1}) - P_{\frac{n}{4}-2}}\,, \\
\phi_{\frac{3n}{4}} &= \frac{\sum_{i=1}^{n/4} \alpha_{i,1} d_{\frac{3n}{4}+i}}{P_{\frac{n}{4}-1}(\lambda-\beta_{n}) - P_{\frac{n}{4}-2}}\,, \\
\text{and}\quad \phi_{\frac{3n}{4}+1} &= \frac{\sum_{i=1}^{n/4} \gamma_{i,n/4} d_{\frac{n}{2}+i}}{P_{\frac{n}{4}-1}(\lambda-\beta_{\frac{n}{2}+1}) - P_{\frac{n}{4}-2}}\,.
\end{aligned}\right\} \qquad (4.7.22)$$

In general, at the end of the $t^{th}$ ($t=1,2,\ldots,\log_2 n-1$) level of the decoupling process we are left with $2^t$ decoupled sub-systems, each of order $n/2^t$. Hence, given a system of dimension $n=2^m$, we require $(m-1)$ $(=\log_2 n-1)$ levels of decoupling to reduce the system into sub-blocks of order 2.

Example: If $n=16=2^4$, then after the $3^{rd}$ level of decoupling we have the resulting decoupled subsystems each reduced to order 2 as shown diagramatically in Figure (4.3).

We shall now assume that the $t^{th}$ level of decoupling has been obtained and hence the process of obtaining the $(t+1)^{th}$ level of decoupling can be generalised in the following manner:

Recursive Point Decoupled Structure of a Tridiagonal System

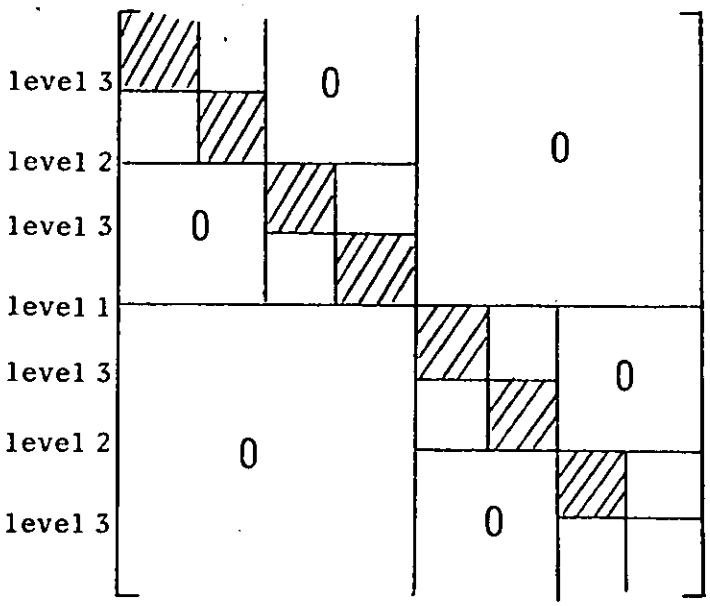



FIGURE 4.3

After the $t^{th}$ level, the first and the last of the $2^t$ decoupled subsystems (each of order $n/2^t$) are of the form,

$$A_1^{(t)} = \begin{bmatrix} \lambda-\beta_1 & & & \\ 1 & \lambda & 1 & \\ & \ddots & \ddots & \ddots \\ & & \lambda & 1 \\ & & 1 & \lambda-\beta_{\frac{n}{2^t}} \end{bmatrix}_{\left(\frac{n}{2^t}\times\frac{n}{2^t}\right)}, \qquad (4.7.23)$$

$$A_{2^t}^{(t)} = \begin{bmatrix} \lambda-\beta_{(2^t-1)(\frac{n}{2^t})+1} & 1 & & & \\ 1 & \lambda & \ddots & & \\ & \ddots & \ddots & 1 & 0 \\ & & \lambda & 1 & \\ 0 & & 1 & \lambda-\beta_n & \end{bmatrix}_{(\frac{n}{2^t}\times\frac{n}{2^t})}, \qquad (4.7.24)$$

and the corresponding right-hand side decoupled sub-vectors of the form,

$$\underline{\tilde{d}}_1^{(t)} = \begin{bmatrix} d_1-\phi_1 \\ d_2 \\ \vdots \\ d_{\frac{n}{2^t}-1} \\ d_{\frac{n}{2^t}}-\phi_{\frac{n}{2^t}} \end{bmatrix}_{(\frac{n}{2^t}\times 1)} \quad \text{and} \quad \underline{\tilde{d}}_{2^t}^{(t)} = \begin{bmatrix} d_{(2^t-1)(\frac{n}{2^t})+1} - \phi_{(2^t-1)(\frac{n}{2^t})+1} \\ d_{(2^t-1)(\frac{n}{2^t})+2} \\ \vdots \\ d_{n-1} - \phi_{n-1} \\ d_n - \phi_n \end{bmatrix}_{(\frac{n}{2^t}\times 1)}. \qquad (4.7.24)$$

The $k^{th}$ $(k=2,3,\ldots,2^t-1)$ other intermediate decoupled sub-systems are of the form,

$$A_k^{(t)} = \begin{bmatrix} \lambda-\beta_{(k-1)(\frac{n}{2^t})+1} & 1 & & & \\ 1 & \lambda & \ddots & & \\ & \ddots & \ddots & 1 & 0 \\ & & \lambda & 1 & \\ 0 & & 1 & \lambda-\beta_{k(\frac{n}{2^t})} & \end{bmatrix}_{(\frac{n}{2^t}\times\frac{n}{2^t})}, \qquad (4.7.25)$$

with the corresponding right-hand side decoupled sub-vectors of the form,

$$\underline{\gamma}_k^{(t)} = \begin{bmatrix} d_{(k-1)(\frac{n}{2^t})+1} - \phi_{(k-1)(\frac{n}{2^t})+1} \\ d_{(k-1)(\frac{n}{2^t})+2} \\ \vdots \\ d_{k(\frac{n}{2^t})-1} \\ d_{k(\frac{n}{2^t})} - \phi_{k(\frac{n}{2^t})} \end{bmatrix} . \qquad (4.7.26)$$

Since all the $2^t$ sub-systems resulting from the t-level of decoupling are independent of each other, then for the $(t+1)^{th}$ level of decoupling, each of the $2^t$ subsystems can now be further decoupled independently and in parallel by a process which amounts to subtracting the modifier quantities, $\beta_{s+2sj}, \beta_{s+2sj+1}$ and $\phi_{s+2sj}, \phi_{s+2sj+1}$ from the corresponding $(s+2sj)^{th}$, $(s+2sj+1)^{th}$ diagonal and right-hand side elements respectively.

Generally, for,

$$\left.\begin{aligned} & t=0,1,\ldots,\log_2 n-1, \quad s=n/2^{t+1} \\ \text{and} \quad & j=0,1,\ldots,t \end{aligned}\right\} \qquad (4.7.27a)$$

these modifier elements, expressed in terms of the sequence $P_i$, are given by,

$$\left.\begin{aligned} \beta_{s+2sj} &= \frac{[\lambda-\beta_{(j+1)(2s)}]P_{s-2}-P_{s-3}}{[\lambda-\beta_{(j+1)(2s)}]P_{s-1}-P_{s-2}} \\ \beta_{s+2sj+1} &= \frac{[\lambda-\beta_{(j+1)(2s)}]P_{s-1}-P_{s-3}}{[\lambda-\beta_{2js+1}]P_{s-1}-P_{s-2}} \\ \phi_{s+2sj} &= \frac{\sum_{i=1}^{s} \alpha_{i,1}^{(t,j+1)} d_{s+2sj+i}}{[\lambda-\beta_{(j+1)2s}]P_{s-1}-P_{s-2}} \\ \text{and} \quad \phi_{s+2sj+1} &= \frac{\sum_{i=1}^{s} \gamma_{i,s}^{(t,j+1)} d_{2sj+i}}{[\lambda-\beta_{2sj+1}]P_{s-1}-P_{s-2}} . \end{aligned}\right\} \qquad (4.7.27b)$$

The quantities $\alpha_{i,1}^{(t,j+1)}$ and $\gamma_{i,s}^{(t,j+1)}$ are the $(1,i)^{th}$ and $(s,i)^{th}$ cofactors of the upper and lower halves of the $(j+1)^{th}$ submatrix which is obtained at

the end of the $t^{th}$ level of decoupling, where

$$\left.\begin{aligned} \gamma_{i,s}^{(t,j+1)} &= \gamma_{i,s}^{(t,1)} + (\beta_1 - \beta_{2sj+1})P_{i-2} \\ \alpha_{i,1}^{(t,j+1)} &= \alpha_{i,1}^{(t,1)} + [\beta_{2s} - \beta_{2s(j+1)}]P_{s-i+1} \end{aligned}\right\} \qquad (4.7.28)$$

with

$$\left.\begin{aligned} \gamma_{1,s}^{(t,1)} &= -1 \quad , \\ \gamma_{2,s}^{(t,1)} &= \lambda - \beta_1 \quad , \\ \gamma_{i,s}^{(t,1)} &= (-1)^i[(\gamma - \beta_1)P_{i-2} - P_{i-3}] \quad , \; i=3,\ldots,s \; , \end{aligned}\right\} \qquad (4.7.29)$$

and

$$\left.\begin{aligned} \alpha_{s,1}^{(t,1)} &= 1 \quad , \\ \alpha_{s-1,1}^{(t,1)} &= \lambda - \beta_{2s} \quad , \\ \alpha_{i,1}^{(t,1)} &= (-1)^{i+1}[(\lambda - \beta_s)P_{s-i+1} - P_{s-i+2}], \; i=s-2,s-3,\ldots,1. \end{aligned}\right\} \qquad (4.7.30)$$

with the terms P defined as in (4.7.14).

The order of calculation of the modifier elements, $\beta_i, \phi_i$, (i=1,2,...,n) can be represented in a tree-structured form as shown in Figure (4.4), where all the 'tree-branching' operations at any given level can be done simultaneously and in parallel.

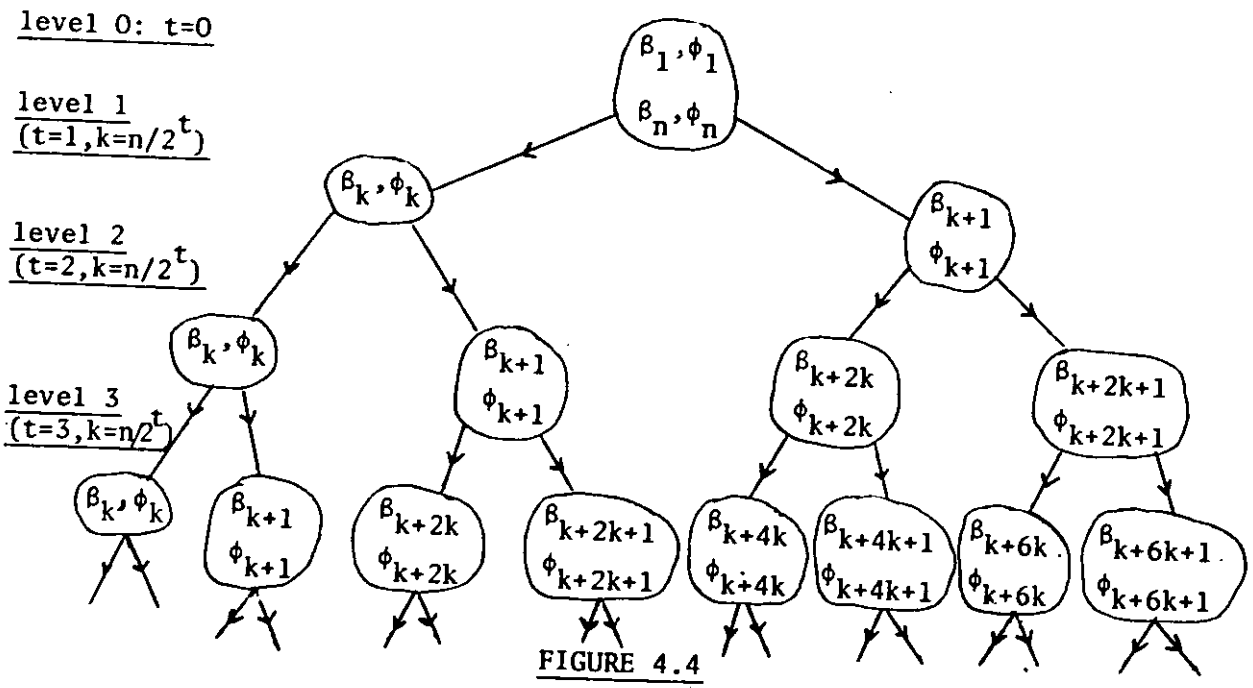



FIGURE 4.4

The Order of Calculation of the Modifier Elements in the Recursive Point Decoupling Process

Finally, at the end of the $(m-1)^{th}$ level of decoupling all the $2^{m-1}$ subsystems are each (2×2) matrix systems which, for $j=2^t$, $t=1,2,\ldots,m-1$, are given by,

$$\begin{pmatrix} \lambda-\beta_{j-1} & 1 \\ 1 & \lambda-\beta_j \end{pmatrix} \begin{pmatrix} u_{j-1} \\ u_j \end{pmatrix} = \begin{pmatrix} d_{j-1} - \phi_{j-1} \\ d_j - \phi_j \end{pmatrix} . \quad (4.7.31)$$

The solution of the system (4.7.31) is given, on omitting interchanges, by,

$$\left.\begin{aligned} u_j &= [(d_j-\phi_j)-(d_{j-1}-\phi_{j-1})(d_j-\phi_j)]/[1-(\lambda-\beta_j)(\lambda-\beta_{j-1})] \\ u_{j-1} &= [(d_j-\phi_j)-u_j]/(\lambda-\beta_j). \end{aligned}\right\} \quad (4.7.32)$$

On the introduction of interchanges the solution of (4.7.31) becomes,

$$\left.\begin{aligned} u_j &= [(\lambda-\beta_{j-1})d_j-(d_{j-1}-\phi_{j-1})]/[(\lambda-\beta_{j-1})(\lambda-\beta_j)-1] \\ \text{and} \quad u_{j-1} &= (d_j-\phi_j)-(\lambda-\beta_j)u_j. \end{aligned}\right\} \quad (4.7.33)$$

The R.P.D. algorithm given above is rather cumbersome and not very competitive for solving a tridiagonal system if the scheme is programmed serially for running on a serial machine. However, since at any level t of the decoupling process, all the intermediate subsystems can all be decoupled further in parallel, the RPD algorithm has its merit in its highly parallel structure. For parallel computers, it is an attractive method to use since it requires only the order of $\log_2 n$ parallel operations, for a system of order n, on a parallel computer having $\log_2 n$ processors.

# CHAPTER 5

# THE SOLUTION OF PERIODIC BOUNDARY VALUE PROBLEMS BY FAST METHODS

## 5.1 INTRODUCTION

A well known source of sparse periodic matrix problems is the system of linear algebraic equations which arise when we solve, by finite difference methods, partial differential equations of parabolic and elliptic type with periodic boundary conditions; such problems are of frequent occurrence in the analysis of Mathematical Physics and Engineering problems.

A number of fast methods for the solution of the one dimensional periodic tridiagonal matrices was presented in Chapter 3. Our interest for presenting those algorithms in point form was partly motivated by the fact that the spectral decomposition technique (Buzbee et al (1970)) discussed in section (5.2), when applied to block separable matrix problems leads to the solution of a set of one-dimensional matrix equations, for which the fast point form algorithms become readily applicable. Moreover, it is often possible to generalise the method of point-form algorithms in order to obtain their block equivalent.

In this chapter, we consider a number of model problems with periodic conditions for which new direct methods are developed for their solution. The method of discrete separation of variables which forms the basis of our comparison with the newly introduced methods is considered in section (5.2). A block factorisation method for the solution of a periodic elliptic partial difference equation in a rectangular region is introduced in section (5.3). In section (5.4), the use of elliptic boundary value techniques in the solution of a periodic parabolic problem is considered; and by adopting the otherwise unstable Richardson's finite difference scheme, a periodic skew symmetric block matrix equation is derived, and an algorithmic solution of this block form proposed and compared with the equivalent spectral resolution method. The use of the classical implicit scheme in the steady state solution of the one space-dimensional parabolic heat conduction equation produces a cyclic lower triangular block matrix. Fast methods for the

solution of such matrix forms are considered in section (5.5). The more usual marching technique for parabolic problems is considered in section (5.6) for the self-adjoint, periodic boundary value parabolic problem for which a generalised factorisation method is employed in the solution of the resulting periodic tridiagonal matrix systems. Finally, in section (5.7), the alternating direction implicit (A.D.I.) methods for the two-space dimensional heat equation is discussed for which fast periodic tridiagonal matrix solver algorithms presented in Chapter 3 are particularly applicable.

## 5.2 THE SOLUTION OF THE PERIODIC MATRIX DIFFERENCE EQUATION BY THE SPECTRAL RESOLUTION METHOD

Algorithm 5.1

We consider the matrix equation,

$$A\underline{u} = \underline{d} \quad , \tag{5.2.1}$$

where A is the (m×m) block matrix of the form,

$$A = \begin{bmatrix} B & C & & & C \\ C & B & C & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & C \\ C & & & C & B \end{bmatrix}_{(m\times m)} \tag{5.2.2}$$

and the submatrices, B and C are (n×n) symmetric real matrices.

The vectors $\underline{u}$ and $\underline{d}$ are written in partitioned form so as to conform with the structure of A and hence have the form,

$$\underline{u} = \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \vdots \\ \underline{u}_m \end{bmatrix} \quad , \qquad \underline{d} = \begin{bmatrix} \underline{d}_1 \\ \underline{d}_2 \\ \vdots \\ \underline{d}_m \end{bmatrix} \tag{5.2.3}$$

where,

$$\underline{u}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix} \quad \text{and} \quad \underline{d}_j = \begin{bmatrix} d_{1,j} \\ d_{2,j} \\ \vdots \\ d_{n,j} \end{bmatrix} \quad , \; j=1,2,\ldots,m. \tag{5.2.4}$$

We assume that B and C commute, (i.e., BC=CB, and have a common basis of eigenvectors.)

Then, by the well known theorem of Frobenius (see, for example, Varga (1962)) there exists an orthogonal matrix Q (i.e., $Q^T=Q^{-1}$) whose columns are the set of eigenvectors of B and C such that

$$\left.\begin{aligned} Q^T B Q &= \Lambda \\ Q^T C Q &= \Omega, \end{aligned}\right\} \qquad (5.2.5)$$

where $\Lambda$ and $\Omega$ are the diagonal matrices whose elements $\lambda_i, \omega_i$ $(i=1,2,\ldots,n)$ are the eigenvalues of B and C respectively.

The system (5.2.1) together with (5.2.2) and (5.2.3), may be written as,

$$B\underline{u}_1 + C\underline{u}_2 + C\underline{u}_m = \underline{d}_1 , \qquad (5.2.6a)$$

$$C\underline{u}_{j-1} + B\underline{u}_j + C\underline{u}_{j+1} = \underline{d}_j , \quad j=2,3,\ldots,m-1, \qquad (5.2.6b)$$

and

$$C\underline{u}_1 + C\underline{u}_{m-1} + B\underline{u}_m = \underline{d}_m . \qquad (5.2.6c)$$

By using equation (2.5.2) we have,

$$B = Q \Lambda Q^T$$

and

$$C = Q \Omega Q^T$$

which, when substituted into (5.2.6), give the following equations

$$\Lambda\bar{\underline{u}}_1 + \Omega\bar{\underline{u}}_2 + \Omega\bar{\underline{u}}_m = \bar{\underline{d}}_1 , \qquad (5.2.7a)$$

$$\Omega\bar{\underline{u}}_{j-1} + \Lambda\bar{\underline{u}}_j + \Omega\bar{\underline{u}}_{j+1} = \bar{\underline{d}}_j , \quad j=2,3,\ldots,m-1, \qquad (5.2.7b)$$

$$\Omega\bar{\underline{u}}_1 + \Omega\bar{\underline{u}}_{m-1} + \Lambda\bar{\underline{u}}_m = \bar{\underline{d}}_m , \qquad (5.2.7c)$$

where

$$\left.\begin{aligned} \bar{\underline{u}}_j &= Q^T \underline{u}_j \\ \bar{\underline{d}}_j &= Q^T \underline{d}_j \end{aligned}\right\} \quad j=1,2,\ldots,m, \qquad (5.2.8)$$

and $\bar{\underline{u}}_j, \bar{\underline{d}}_j$ are labelled as in (5.2.4).

Further, we now resolve the equations in (5.2.7) by rewriting them, for $i=1,2,\ldots,n$, as

$$\lambda_i \bar{u}_{i,1} + \omega_i \bar{u}_{i,2} + \omega_i \bar{u}_{i,m} = \bar{d}_{i,1} \qquad (5.2.9a)$$

$$\omega_i u_{i,j-1} + \lambda_i \bar{u}_{i,j} + \omega_i u_{i,j+1} = \bar{d}_{i,j} , \quad j=2,3,\ldots,m-1, \qquad (5.2.9b)$$

$$\omega_i u_{i,1} + \omega_i \bar{u}_{i,m-1} + \lambda_i u_{i,m} = \bar{d}_{i,m} . \qquad (5.2.9c)$$

Now, if we write

$$\Gamma_i = \begin{bmatrix} \lambda_i & \omega_i & & & \omega_i \\ \omega_i & \lambda_i & \omega_i & 0 & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \omega_i \\ \omega_i & & 0 & \omega_i & \lambda_i \end{bmatrix}_{(m\times m)} , \qquad (5.2.10)$$

$$\hat{\underline{u}}_i = \begin{bmatrix} \bar{u}_{i,1} \\ \bar{u}_{i,2} \\ \vdots \\ \bar{u}_{i,m} \end{bmatrix} \quad \text{and} \quad \hat{\underline{d}}_i = \begin{bmatrix} \bar{d}_{i,1} \\ \bar{d}_{i,2} \\ \vdots \\ \bar{d}_{i,m} \end{bmatrix} \tag{5.2.11}$$

then the equations in (5.2.9) are equivalent to the system

$$\Gamma_i \hat{\underline{u}}_i = \hat{\underline{d}}_i, \qquad i=1,2,\ldots,n. \tag{5.2.12}$$

Thus, the vector $\hat{\underline{u}}_i$ satisfies a symmetric tridiagonal matrix system of equations that has a constant diagonal, super- and sub-diagonal elements as in (5.2.10) which can be solved in an efficient manner by using either the PQFACT1 algorithm (3.6) or algorithm (3.10) in 5m multiplications and 4m additions per system.

After solving (5.2.12), it is then possible to solve for

$$\underline{u}_j = Q\bar{\underline{u}}_j \quad , \qquad j=1,2,\ldots,m. \tag{5.2.13}$$

The above matrix decomposition algorithm is due to Buzbee et al (1970). If we regard the block vectors $\underline{u}$ and $\underline{d}$ as 2-dimensional arrays, then the above algorithm may be summarised as follows:

Step 1

Compute or determine the eigenvalues of matrices B and C and the eigenvectors of B. These eigensystems are often given by known analytical formulae for certain representations of B and C; e.g. in the case of solving a Poisson equation in a sequence, B is tridiagonal and C is diagonal.

Step 2

Compute the vectors,

$$\bar{\underline{d}}_j = Q^T \underline{d}_j \quad , \qquad j=1,2,\ldots,m \tag{5.2.14}$$

which is equivalent to multiplying each row of $\underline{d}$ by $Q^T$.

Step 3

Next, we re-order the array $\bar{\underline{d}}$ by vertical lines instead of horizontal

lines to generate the array $\hat{\underline{d}}$ and then solve the tridiagonal systems,

$$\Gamma_i \hat{\underline{u}}_i = \hat{\underline{d}}_i \quad , \qquad i=1,2,\ldots,n. \tag{5.2.15}$$

Step 4

Finally, we re-order the array $\underline{u}$ by horizontal lines instead of vertical lines to generate the array $\bar{\underline{u}}$ and then compute the solution vector,

$$\underline{u}_j = Q\bar{\underline{u}}_j \quad , \qquad j=1,2,\ldots,m. \tag{5.2.16}$$

If we neglect the computation of the eigensystem (step 1) then, the operation count for the spectral resolution method is given as shown in Table (5.1). A reduction in the arithmetic operation count is possible if the fast Fourier transform (Cooley and Tukey (1965)) is used to perform steps (2) and (4).

Summary of Arithmetic Operation Count

| Steps | Multiplications(×) | Additions(+) |
|---|---|---|
| 2 | $n^2m$ | $n^2m$ |
| 3 | $5nm$ | $4nm$ |
| 4 | $n^2m$ | $n^2m$ |
| TOTAL | $2n^2m+5nm$ | $2n^2m+4mn$ |

TABLE 5.1

## 5.3 A BLOCK FACTORISATION METHOD FOR THE SOLUTION OF AN ELLIPTIC P.D.E. WITH PERIODIC BOUNDARY CONDITIONS IN A RECTANGULAR REGION

We consider here the block matrix factorisation method for the solution of an elliptic partial differential equation with periodic boundary conditions in a rectangle.

Problem Definition

Consider the elliptic (Helmholtz's) equation in two space dimensions given by,

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \tau U(x,y) + q(x,y) \tag{5.3.1}$$

on a rectangular region,

$$R : \begin{cases} 0 \leq x \leq a \\ 0 \leq y \leq b \end{cases} \tag{5.3.2}$$

enclosed by the boundary region $\partial R$, and with the periodic boundary conditions in both the x- and the y-directions given by,

$$\begin{aligned} U(0,y) &= U(a,y), \quad q(0,y) = q(a,y) , \\ U(x,0) &= U(x,b), \quad q(x,0) = q(x,b) , \end{aligned} \tag{5.3.3}$$

where $q(x,y)$ is a known function in x and y, and $\tau$ is any positive constant.

We define the mesh spacings $\Delta x = a/m$ and $\Delta y = b/n$ (m,n are integers) and then super-impose mesh-points $(x_i, y_j) = (i\Delta x, j\Delta y)$ over the interior region $R_h$ and the discrete boundary region, $\partial R_h$ where

$$R_h = \{(x_i, y_j) \mid 1 \leq i \leq m-1, \ 1 \leq j \leq n-1\}$$

and

$$\partial R_h = \partial R \cap \{(x_i, y_j) \mid 0 \leq i \leq m, \ 0 \leq j \leq n\}.$$

Further, by using the notation $U_{i,j} = U(x_i, y_j)$ and then applying the five point finite difference approximation (see section (2.2)), i.e.,

$$\nabla^2 U_{i,j} = \frac{1}{(\Delta x)^2}(U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) + \frac{1}{(\Delta y)^2}(U_{i-1,j-1} - 2U_{i,j} + U_{i,j+1}), \tag{5.3.4}$$

the given elliptic equation (5.3.1) at the point (i,j) becomes,

$$U_{i-1,j} + \sigma U_{i,j-1} - 2(1+\sigma+\tau)U_{i,j} + \sigma U_{i,j+1} + U_{i+1,j} = d_{i,j} \tag{5.3.5}$$

where

$$\sigma = \left(\frac{\Delta x}{\Delta y}\right)^2 \quad \text{and} \quad d_{i,j} = (\Delta x)^2 q_{i,j} .$$

We shall here adopt a column-wise ordering of the mesh points as illustrated in Figure (5.1).

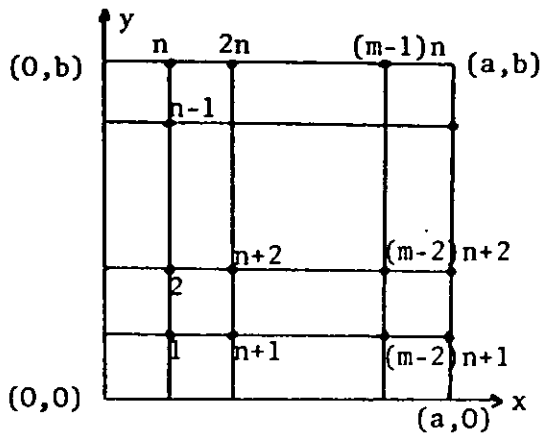



FIGURE 5.1

Hence, by applying equation (5.3.5) over the enclosed rectangular region together with the periodic boundary conditions (5.3.3), we obtain the block matrix system,

$$A\underline{u} = \underline{d} \tag{5.3.6}$$

where

$$A = A[I,B,I] = \begin{bmatrix} B & I & & & I \\ I & B & I & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & & I & \\ I & & & I & B \end{bmatrix}_{(m\times m)}, \tag{5.3.7}$$

$\underline{u}$, the approximate finite difference solution of (5.3.1) is given by,

$$\underline{u} = \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \vdots \\ \underline{u}_m \end{bmatrix}, \quad \underline{u}_i = \begin{bmatrix} u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,n} \end{bmatrix}, \quad i=1,2,\ldots,m,$$

and the vector $\underline{d}$ is similarly partitioned as $\underline{u}$.

The matrix B is an $(n\times n)$ periodic tridiagonal matrix of the form,

$$B=B[\sigma,-2(1+\sigma+\tau),\sigma]=\begin{bmatrix} -2(1+\sigma+\tau) & \sigma & & & \sigma \\ \sigma & -2(1+\sigma+\tau) & \sigma & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \sigma \\ \sigma & & & \sigma & -2(1+\sigma+\tau) \end{bmatrix} \qquad (5.38)$$

and I is the (n×n) identity matrix.

Cyclic Block Factorisation Method

In Chapter 3, we introduced the general cyclic factorisation method for the solution of the general periodic tridiagonal matrix equation and then derived the various simplified variants of that general algorithm. Although each of the periodic matrix solver algorithms ((3.2), and (3.6)-(3.9)) given is applicable to the periodic matrix systems in point form, the analogous block equivalent exists, provided the norm of the matrix B is greater than 2 (see (3.4.6) for the equivalent condition for the point case).

Here we give the block form of the PQFACT1 algorithm (3.6) which requires the replacement of the matrix scalar elements 'b' and 'a' by the submatrix B and the identity matrix I respectively in order to obtain a direct solution of the block matrix equation (5.3.6) by the following procedure:

By defining a submatrix N of order n, (cf. 3.4.7), i.e.,

$$N = .5[B-(B^2-4I)^{\frac{1}{2}}], \quad ||B||_\infty > 2 \qquad (5.3.9)$$

where $W^{1/2}$ is defined as the square root of matrix W (Späth, 1967)

then, the right-hand side vector $\underline{d}$ is modified to be

$$\left.\begin{aligned} \underline{d}'_1 &= \underline{d}_1 \\ \underline{d}'_i &= \underline{d}_i - N\underline{d}'_{i-1}, \quad i=2,3,\ldots,m. \end{aligned}\right\} \qquad (5.3.10)$$

Next, we calculate the intermediate solution sub-vectors $\underline{y}_i$ (i=1,...,m) as

$$\left.\begin{aligned} \underline{y}_m &= (I+\theta N^m)^{-1}\underline{d}'_m \\ \text{and} \quad \underline{y}_i &= \underline{d}'_i - \theta N^i \underline{y}_m, \quad i=m-1,m-2,\ldots,1, \end{aligned}\right\} \qquad (5.3.11)$$

where $\Theta=1$ (i odd) and $\Theta=-1$, (i even).

Further, by defining the intermediate vectors,

$$\left.\begin{aligned} \underline{g}_m &= N\underline{y}_m \\ \text{and} \quad \underline{g}_i &= N(\underline{y}_i - \underline{g}_{i+1}), \quad i=m-1,m-2,\ldots,1 \end{aligned}\right\} \tag{5.3.12}$$

we then obtain the final solution vector $\underline{u}$ from the expressions,

$$\left.\begin{aligned} \underline{u}_1 &= (I+\Theta N^m)^{-1}\underline{g}_1 \\ \text{and} \quad \underline{u}_i &= \underline{g}_i - \Theta N^{m-i+1}\underline{u}_1, \quad i=2,3,\ldots,m. \end{aligned}\right\} \tag{5.3.13}$$

This block factorisation method (5.3.9)-(5.3.13) which we denote as the BKFACT1 algorithm gives the desired direct solution of equation (5.3.6) provided that the matrix $(I+\Theta N^m)$ is non-singular. It is also necessary for stability considerations, to have the norm of the matrix N (since N is used as a multiplier in an elimination process) to be less than unity.

Implementation of the BKFACT1 Algorithm

The determination of the sub-matrix N in the form given in (5.3.9) involves the evaluation of the square root of a positive definite matrix $(B^2-4I)$. Various iterative methods, based on the Raphson-Newton scheme or its variants, are known to exist for the determination of the square root of such matrices. References for this include Laasonen (1958), Spath (1967), Babuska et al (1966) and Schofield (1973). Also the BKFACT1 algorithm involves the calculation of the multiple powers of the matrix N which can be achieved by repeated multiplication. However, both the iterative method for the evaluation of the square root of a matrix and the repeated matrix multiplication required to obtain the higher order powers of the matrix N would lead to a grossly inefficient method requiring excessive computing effort and storage. Thus, in the BKFACT1 algorithmic solution these computational difficulties must be resolved before the algorithm can become competitive with alternative methods such as the spectral resolution method introduced in section (5.2).

The matrix $B[\sigma,-2(1+\sigma+\tau),\sigma]$ defined in (5.3,8) is a symmetric cyclic tridiagonal matrix. Thus, if we introduce

$$\Lambda_B = \mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$$

as the matrix of eigenvalues of B, then it is well known (Polozhii (1974)) that $\Lambda_B$ is given as

$$\Lambda_B = \begin{bmatrix} b+2\sigma & & & & & & \\ & b+2\sigma\cos\alpha & & & & & \\ & & b+2\sigma\cos\alpha & & & 0 & \\ & & & b+2\sigma\cos 2\alpha & & & \\ & & & & b+2\sigma\cos 2\alpha & & \\ & 0 & & & & \ddots & \\ & & & & & & b+2\sigma\cos(\frac{n-1}{2})\alpha \\ & & & & & & & b+2\sigma\cos(\frac{n-1}{2})\alpha \end{bmatrix}, \quad n \text{ odd} \qquad (5.3.14)$$

or

$$\Lambda_B = \begin{bmatrix} b+2\sigma & & & & & & \\ & b+2\sigma\cos\alpha & & & & & \\ & & b+2\sigma\cos\alpha & & & 0 & \\ & & & b+2\sigma\cos 2\alpha & & & \\ & & & & b+2\sigma\cos 2\alpha & & \\ & 0 & & & & \ddots & \\ & & & & & & b+2\sigma\cos(\frac{n}{2}-1)\alpha \\ & & & & & & & b+2\sigma\cos(\frac{n}{2}-1)\alpha \\ & & & & & & & & b-2\sigma \end{bmatrix}, \quad n \text{ even} \qquad (5.3.15)$$

where $b=-2(1+\sigma+\tau)$ and $\alpha=\frac{2\pi}{n}$ .

It is also known that the $(n\times n)$ fundamental matrix Q whose columns are the normalised eigenvectors of B corresponding to the eigenvalues $\Lambda_B$ may be obtained in the following form:

$$Q=\sqrt{\frac{2}{n}}\begin{bmatrix} \sqrt{\frac{1}{2}}, & \cos\alpha, & \sin\alpha, & \cos 2\alpha, & \sin 2\alpha, \ldots\ldots & \cos\frac{(n-1)\alpha}{2}, & \sin\frac{(n-1)\alpha}{2} \\ \sqrt{\frac{1}{2}}, & \cos 2\alpha, & \sin 2\alpha, & \cos 4\alpha, & \sin 4\alpha, \ldots\ldots & \cos\frac{2(n-1)\alpha}{2}, & \sin\frac{2(n-1)\alpha}{2} \\ \sqrt{\frac{1}{2}}, & \cos 3\alpha, & \sin 3\alpha, & \cos 6\alpha, & \sin 6\alpha, \ldots\ldots & \cos\frac{3(n-1)\alpha}{2}, & \sin\frac{3(n-1)\alpha}{2} \\ \ldots\ldots & & & & & & \\ \sqrt{\frac{1}{2}}, & \cos n\alpha, & \sin n\alpha, & \cos 2n\alpha, & \sin 2n\alpha, \ldots\ldots & \cos\frac{n(n-1)\alpha}{2}, & \sin\frac{n(n-1)\alpha}{2} \end{bmatrix} \qquad (5.3.16)$$

when n is odd,

and

$$Q=\sqrt{\frac{2}{n}}\begin{bmatrix} \sqrt{\frac{1}{2}}, & \cos\alpha, & \cos 2\alpha, & \sin 2\alpha, \ldots\ldots & \cos\frac{(n-2)\alpha}{2}, & \sin\frac{(n-2)\alpha}{2}, & -\sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}}, & \cos 2\alpha, & \cos 4\alpha, & \sin 4\alpha, \ldots\ldots & \cos\frac{2(n-2)\alpha}{2}, & \sin\frac{2(n-2)\alpha}{2}, & \sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}}, & \cos 3\alpha, & \cos 6\alpha, & \sin 6\alpha, \ldots\ldots & \cos\frac{3(n-2)\alpha}{2}, & \sin\frac{3(n-2)\alpha}{2}, & -\sqrt{\frac{1}{2}} \\ \ldots\ldots & & & & & & \\ \sqrt{\frac{1}{2}}, & \cos n\alpha, & \cos 2n\alpha, & \sin 2n\alpha, \ldots\ldots & \cos\frac{n(n-1)\alpha}{2}, & \sin\frac{n(n-1)\alpha}{2}, & \sqrt{\frac{1}{2}} \end{bmatrix} \qquad (5.3.17)$$

when n is even.

By theorems (2.7) and (2.8) the orthogonal matrix Q and the tri-diagonal matrix B are related in the form $B=Q\Lambda_B Q^T$. Thus, using this relation we now state and prove the following lemma:

Lemma 5.1

If the orthogonal matrix Q of eigenvectors of the matrix B which has the matrix of eigenvalues $\Lambda_B$ exist such that $B=Q\Lambda_B Q^T$, then for any integer k

$$B^k = Q\Lambda_B^k Q^T \quad . \qquad (5.3.18)$$

Proof

$$B^k = (Q\Lambda_B Q^T)(Q\Lambda_B Q^T)(Q\Lambda_B Q^T) \ \ldots \text{ repeated k times.}$$

The associative law allows us to remove the brackets and since $QQ^T=I$ from the orthogonality property, the product gives the required result.

Lemma 5.2

Let $f(B)=a_0B^k+a_1B^{k-1}+\ldots a_{k-1}B+a_kI$ . (5.3.19)

denote a polynomial of degree k in the matrix B, then

$$f(B) = Qf(\Lambda_B)Q^T$$

where $f(\Lambda_B)$ is a polynomial of degree k of the diagonal matrix $\Lambda_B$.

Proof

This proof is given as in Noble (1969), P. 348.

We require the result of Lemma 5.1.

By substituting (5.3.18) into (5.3.19) we have,

$$f(B) = a_0 Q\Lambda_B^k Q^T + a_1 Q\Lambda_B^{k-1} Q^T + \ldots a_{k-1} Q\Lambda_B Q^T + a_k QQ^T$$

$$= Q(a_0\Lambda_B^k + a_1\Lambda_B^{k-1} + \ldots a_{k-1}\Lambda_B + a_k I)Q^T$$

$$= Qf(\Lambda_B)Q^T$$

and the result follows.

By using Lemma (5.2), it is now possible to express the (n×n) submatrix $N(=0.5[B-(B^2-4I)^{\frac{1}{2}}])$ and the $i^{th}$ power of N, for any integer i, in the forms,

$$\left.\begin{aligned} N &= Q\Lambda_N Q^T \\ N^i &= Q\Lambda_N^i Q^T \end{aligned}\right\} \qquad (5.3.20)$$

where $\Lambda_N = f(\Lambda_B) = \text{diag}(\mu_1, \mu_2, \ldots, \mu_n)$,

$\Lambda_N^i = f(\Lambda_B^i) = \text{diag}(\mu_1^i, \mu_2^i, \ldots, \mu_n^i)$

and $u_i = \frac{1}{2}[\lambda_i - (\lambda_i^2 - 4)^{\frac{1}{2}}]$ . (5.3.21)

Similarly, the matrix inverse $(I+N^m)^{-1}$ can be expressed in the form,

$$(I+N^m)^{-1} = Q\bar{\Lambda}Q^T \qquad (5.3.22)$$

where

$$\bar{\Lambda} = [\text{diag}(1+\mu_1^m, 1+\mu_2^m, \ldots, 1+\mu_n^m)]^{-1} .$$

It is now necessary to demonstrate the extent to which the introduction of the forms (5.3.20) and (5.3.22) into the BKFACT1 algorithm can greatly reduce the amount of arithmetic calculation required for the implementation of that algorithm.

We consider, for example, the determination of the intermediate vector, $\underline{y}_i$ (i=m-1,m-2,...,1) given in (5.3.11), i.e.,

$$\underline{y}_i = \underline{d}_i' \pm N^i \underline{y}_m , \quad i=m-1,m-2,\ldots 1.$$

On using (5.3.20), this becomes,

$$\underline{y}_i = \underline{d}_i' \pm Q\Lambda_N^i Q^T \underline{y}_m , \qquad i=m-1,m-2,\ldots 1$$

i.e., $$\underline{y}_i = \underline{d}_i' \pm W_i \underline{z} , \qquad i=m-1,m-2,\ldots 1 \qquad (5.3.23a)$$

where $W_i = Q\Lambda_N^i$, $\underline{z} = Q^T \underline{y}_m$ . (5.3.23b)

Thus, if the vector $\underline{z}$ is computed first, then the subsequent computation of each $\underline{y}_i$ in (5.2.23a) requires only one matrix-vector multiplication and a vector addition.

Thus, by introducing the modified forms of $N^i$ and $(I+N^m)^{-1}$ given by (5.3.20) and (5.3.22) respectively, and the simplifications introduced in (5.3.23), into the BKFACT1 algorithm (5.3.9-5.3.13), then the arithmetic operation count of the algorithm becomes $O(4n^2m)$ (if we exclude the evaluation of the eigensystem of $\mathcal{B}$). Table (5.2) gives a summary of sections of the operational count of the BKFACT1 algorithm.

Summary of Operation Counts

| Quantities and Equation No. | Nature of Computation (Each matrix/vector is of order n) | Multiplications × | Additions + |
|---|---|---|---|
| 1. $\underline{d}_i'$, i=2,3,...m (5.3.10) | vector-matrix×vector | $n^2(m-1)$ | $n^2(m-1)+n(m-1)$ |
| 2. $\underline{y}_i$, i=1,2,...m (5.3.11) | vector-(matrix×vector) | $n^2m$ | $n^2m+nm$ |
| 3. $\underline{g}_i$, i=1,2,...m (5.3.12) | matrix*(vector+vector) | $n^2m$ | $n^2m+nm$ |
| 4. $\underline{u}_i$, i=1,2,...m (5.3.13) | vector+(matrix×vector) | $n^2m$ | $n^2m+nm$ |
| TOTAL | $4n^2m$ | $4n^2m$ | $4n^2m+4nm$ |

TABLE 5.2

Numerical Experiments

A Fortran program (written and run in single precision arithmetic) to implement the BKFACT1 algorithm is given as Program 14 in Appendix I. This program was tested on a variety of problems of the form (5.3.1) using the Loughborough University I.C.L. 1904S computer. We consider a specific case in which we adopt a uniform mesh size, $\Delta x=\Delta y=h=1/n$ within a square region $\{0\leq x,y\leq 1\}$, i.e., a=b=1. By choosing an exact solution vector $\underline{u}$ to be the unit

vector, the right-hand side vector was then determined by multiplying $\underline{u}$ by the coefficient matrix A given in (5.3.7). The approximate solution was then recomputed by imputing both the coefficient matrix and the pre-determined right hand side vector into the BKFACT1 subroutine. The relevance of this example is that it gives an indication of the round-off error that can be expected in the BKFACT1 algorithm.

In order to assess the performance of the BKFACT1 algorithm, both the maximum error of the computed solution (i.e., the maximum absolute difference between the known and computed solution) and the computation time were determined, and comparisons made between these results and those of the spectral resolution algorithm (5.1). The maximum errors and execution times (in mill units) obtained from both algorithms for different mesh sizes h, are summarised in Table (5.3).

It can be seen from this table that the spectral resolution method is about 25% faster than the BKFACT1 algorithm, but there is very little to choose between them in terms of accuracy.

Mean Max Error and Execution Times (in Mill-Units) for the Solution of the Finite Difference Equation (5.3.7) of order $n^2$ where $n=h^{-1}$

| $h^{-1}=n$ | Algorithms | Max.Error ($10^{-10}$) | Execution Time (Mill-Units) | Scaled Execution Time |
|---|---|---|---|---|
| 10 | BKFACT1<br>Spectral Resolution | 9.2<br>8.1 | 7<br>6 | 1.16<br>1.0 |
| 15 | BKFACT1<br>Spectral Resolution | 16.3<br>15.8 | 18<br>16 | 1.13<br>1.0 |
| 20 | BKFACT1<br>Spectral Resolution | 20.6<br>20.8 | 43<br>34 | 1.26<br>1.0 |
| 30 | BKFACT1<br>Spectral Resolution | 31.6<br>32.8 | 140<br>104 | 1.34<br>1.0 |
| 40 | BKFACT1<br>Spectral Resolution | 43.8<br>47.2 | 296<br>234 | 1.24<br>1.0 |

TABLE 5.3

## 5.4 THE SOLUTION OF PERIODIC PARABOLIC PROBLEMS BY BOUNDARY-VALUE TECHNIQUES

Finite-difference methods for parabolic initial boundary problems are usually treated as marching procedures in which we seek a point (or line) step-by-step solution on a chosen network of lines over a given semi-infinite region (e.g., $R:\{0\leq x\leq a,\ t\geq 0\}$. In such a step-ahead technique, results on one row are used to generate results on the next row in a recursive fashion. There is, therefore, inherent in such a method, an accumulation of the effects of round-off and truncation errors, as we proceed from row to row.

A method which has been developed (Carasso and Parter (1970) and Greenspan (1974)) to remove this row-to-row error accumulation is by the use of elliptic boundary-value techniques for the numerical computation of such parabolic problems. It is based on the assumption that the solution of the parabolic p.d.e. reaches a known steady state value as time, $t\to\infty$ and hence one is able to provide or determine solutions on the line $t=T$ (for a sufficiently large, preselected value T). With this extra data, it then becomes feasible to solve the parabolic problem by the use of an elliptic-boundary value method within the truncated region, $R:\{0\leq x\leq a,\ 0\leq t\leq T\}$. The mathematical theory which supports the validity and viability of this method is given in Carasso and Parter (1970).

Here we consider a periodic parabolic problem in which, instead of the usual initial boundary condition, we are given periodic conditions along the infinite t-axis (i.e., the solution is periodically repeated at some interval of time T (e.g. $T=2\pi$)). Such a periodic condition thus provides the extra data (equivalent to the steady state solution of a non-periodic problem) required for a complete specification of the reformulated elliptic problem.

### Problem Definition

We now consider, for example, the linear one-dimensional heat-conduction equation,

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial U}{\partial t} = F(x,t) \tag{5.4.1}$$

in the semi-infinite strip,

$$R:\{0\leq x\leq a,\ t\geq 0\} \tag{5.4.2}$$

subject to the Dirichlet's boundary conditions,

$$\begin{aligned} U(0,t) &= f(t), \quad t\geq 0 \\ U(a,t) &= g(t), \quad t\geq 0\ , \end{aligned} \tag{5.4.3}$$

and the periodic condition,

$$U(x,t+T) = U(x,t), \quad 0\leq x\leq a,\ t\geq 0\ . \tag{5.4.4}$$

This problem reformulated as an elliptic-boundary value problem is equivalent to (5.4.1) subject to the boundary conditions (5.4.3) over the enclosed truncated rectangular region, $R=\{(x,t)\,|\,0\leq x\leq a,\ 0\leq t\leq T\}$ where the solution is required to satisfy the periodicity condition (5.4.4) as illustrated in Figure (5.2).

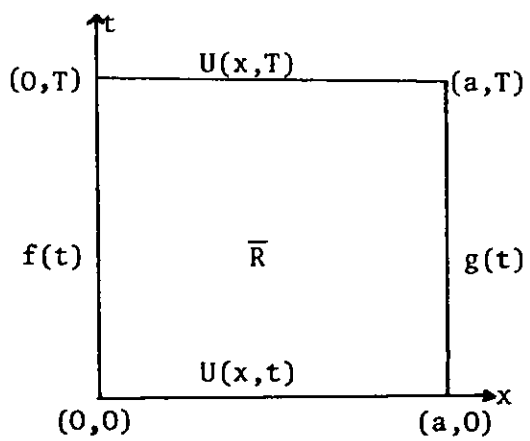



FIGURE 5.2

Next, we divide the range (0,T) of t into m uniform mesh sizes each of length k=T/m; and the range (0,a) of x into (n+1) equal steps of size h=a/(n+1) such that

$$t_j = jk \quad , j=0,1,\ldots,m$$

and

$$x_i = ih \quad , i=0,1,\ldots,n+1.$$

Further, we write,

$$U_{i,j} = U(x_i,t_j) = U(ih,jk),$$

$$F_{i,j} = F(x_i,t_j) = F(ih,jk),$$

$$f_j = f(t_j) ,$$

and

$$g_j = g(t_j) .$$

and then consider the following finite difference approximation,

$$\frac{\partial^2 U}{\partial x^2} = (U_{i-1,j} - 2U_{i,j} + U_{i+1,j})/h^2 + O(h^2)$$

and

$$\frac{\partial U}{\partial t} = (U_{i,j+1} - U_{i,j-1})/2k + O(k^2).$$

Hence, equation (5.4.1) can be represented, on neglecting the truncation error, at the point (i,j) in R by the five point central difference operator,

$$-U_{i,j-1} - 2rU_{i-1,j} + 4rU_{i,j} - 2rU_{i+1,j} + U_{i,j+1} = -2kF_{i,j} \qquad (5.4.5)$$

$$i=1,2,\ldots,n;\ j=1,2,\ldots,m,$$

where $r=k/h^2$.

The scheme (5.4.5) is the so-called 'leap frog' or Richardson's scheme and it is well-known that when used in a marching procedure with parabolic p.d.e., it leads to an improperly posed numerical problem as the data on the line t=k must be supplied, in addition to the usual initial data, in order to start the calculation. It is also known that the Richardson's scheme used as a marching procedure is unconditionally unstable (Forsythe and Wasow (1960)). It has however been established in a proof given by Carasso and Parter (1970) that as a boundary-value procedure, for linear and mildly non-linear problems, the Richardson's scheme is unconditionally stable.

Hence, we consider a row-wise ordering of the mesh points in the truncated region R (Figure 5.2) and apply the difference equation (5.4.5) at every point in R to obtain the finite difference approximation of (5.4.1)-(5.4.4) as,

$$A\underline{u} = \underline{d} \qquad (5.4.6)$$

where A is the (m×m) skew-symmetric block tridiagonal matrix,

$$A[-I,B,I] = \begin{bmatrix} B & I & & & -I \\ -I & B & I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & B & I \\ I & & & -I & B \end{bmatrix}_{(m\times m)} \qquad (5.4.7)$$

I is the (n×n) identity matrix and B is a tridiagonal matrix given by,

$$B[-2r,4r,-2r] = \begin{bmatrix} 4r & -2r & & & \\ -2r & 4r & -2r & & \\ & \ddots & \ddots & \ddots & \\ & & -2r & 4r & -2r \\ & & & -2r & 4r \end{bmatrix}_{(n\times n)} \qquad (5.4.8)$$

The solution vector $\underline{u}$ (the approximation of $\underline{U}$) and the right-hand side vector $\underline{d}$, each of which can be thought of as a two dimensional array, are written as,

$$\underline{u} = \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \vdots \\ \underline{u}_m \end{bmatrix}, \quad \underline{u}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix}, \quad j=1,2,\ldots,m, \qquad (5.4.9a)$$

and

$$\underline{d} = \begin{bmatrix} \underline{d}_1 \\ \underline{d}_2 \\ \vdots \\ \underline{d}_m \end{bmatrix}, \quad \underline{d}_j = \begin{bmatrix} -2kF_{1,j}+2rf_j \\ -2kF_{2,j} \\ \vdots \\ -2kF_{n-1,j} \\ -2kF_{n,j}+2rg_j \end{bmatrix}, \quad j=1,2,\ldots,m. \qquad (5.4.9b)$$

A block factorisation method for the solution of the skew symmetric matrix A[-1,B,I]

The skew symmetric block matrix A given in (5.4.7) can be factorised into the product of P and Q, i.e.,

$$A = PQ \qquad (5.4.10a)$$

where

$$P = \begin{bmatrix} I & & & -N \\ -N & I & & \\ & \ddots & \ddots & \\ 0 & & -N & I \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} M & I & & \\ & M & I & \\ & & \ddots & \ddots \\ I & 0 & & I \\ & & & M \end{bmatrix} \qquad (5.4.10b)$$

Both N and M are assumed to be square matrices. If we then multiply P and Q together and equate to A as in (5.4.10a) we obtain immediately the relations,

$$\left.\begin{aligned} NM &= I \\ M-N &= B \end{aligned}\right\} \qquad (5.4.11)$$

and

Either by using the continued fraction concept (see (3.3.4)-(3.3.6) or by a simple direct substitution, the value of N is derived from (5.4.11) as,

$$N = 0.5[-B+(B^2+4I)^{\frac{1}{2}}] \ .$$

Since the matrix A in (5.4.7) is the block form of the matrix (3.4.21), it is feasible to adopt the block form of the PQFACT4 algorithm (3.9) in the solution of equation (5.4.6). Hence, by assuming that the vectors $\underline{u}$ and $\underline{d}$ are partitioned relative to A, we derive immediately, as a block form generalisation of the PQFACT4 algorithm (3.9), the following direct method for the solution of the block skew symmetric matrix (5.4.7):

We define the submatrix N of order n as,

$$N = 0.5[-B+(B^2+4I)^{\frac{1}{2}}] \ , \qquad (5.4.12)$$

and then, the vector $\underline{d}$ is transformed to the form,

$$\begin{aligned} \underline{d}'_1 &= \underline{d}_1 \\ \underline{d}'_j &= \underline{d}_j + N\underline{d}'_{j-1}, \quad j=2,3,\ldots,m. \end{aligned} \qquad (5.4.13)$$

Next, we obtain the intermediate solution sub-vectors $\underline{y}_j$ as,

$$\begin{aligned} \underline{y}_m &= (I-N^m)^{-1}\underline{d}'_m \\ \underline{y}_j &= \underline{d}'_j + N^j\underline{y}_m \ , \quad j=1,2,\ldots,m-1. \end{aligned} \qquad (5.4.14)$$

Further, by defining the intermediate vectors,

$$\begin{aligned} \underline{g}_m &= N\underline{y}_m \\ \underline{g}_j &= N(\underline{y}_j - \underline{g}_{j+1}) \ , \quad j=m-1,m-2,\ldots,1, \end{aligned} \qquad (5.4.15)$$

and

we then obtain the final solution vector $\underline{u}$ as,

$$\underline{u}_1 = (I-\tau N^m)^{-1}\underline{g}_1 \quad ,$$

and
$$\underline{u}_j = \underline{g}_j+\tau N^{m-j+1}\underline{u}_1 \quad , \quad j=2,3,\ldots,m \qquad (5.4.16)$$

where $\tau=-1$ (if m and j are both even or both odd)

$= 1$ otherwise.

We denote algorithm (5.4.12)-(5.4.16) as the BKFACT4 algorithm which gives a fast direct solution of the difference equation (5.4.6) provided that $(I-\tau N^m)$ is non-singular and that $||N||_\infty<1$.

Implementation of BKFACT4 Algorithm

The matrix $B[-2r,4r,-2r]$ given by (5.4.8) is a constant tridiagonal matrix of order n. It is well known (Polozhii (1974)) that its matrix of eigenvalues, $\Lambda_B$, is given by

$$\Lambda_B = \text{diag}(\lambda_1,\lambda_2,\ldots\lambda_n)$$

where
$$\lambda_i = 4r(1-\cos(\frac{i\pi}{n+1})) \quad , \quad i=1,\ldots,n \qquad (5.4.17)$$

and the orthonormalised matrix $Q=(q_{i,j})$ whose columns are the eigenvectors of B is given by,

$$(Q)_{i,j}=q_{i,j}=\sqrt{2/(n+1)}\,\sin(\frac{ij\pi}{n+1}), \quad i,j=1,2,\ldots,n. \qquad (5.4.18)$$

The matrix Q is symmetric.

With the eigenvalues and eigenvectors of B determined by (5.4.17) and (5.4.18) respectively, it is then possible to employ the relations (5.3.20) and (5.3.22) in a further simplification of the BKFACT4 algorithm just as was shown for the BKFACT1 algorithm, in order to improve the competitiveness of the latter method. Thus instead of the expression for N, for example, in (5.4.12), we can write,

$$N = Q\Lambda_N Q^T$$

where $\Lambda_N = \text{diag}(\mu_1,\mu_2,\ldots,\mu_n)$,

and
$$\mu_i = \tfrac{1}{2}[-\lambda_i+(\lambda_i^2+4)^{\frac{1}{2}}], \quad i=1,2,\ldots,n. \qquad (5.4.19)$$

By using this and other similar modifications, it can be shown, as was done in Table (5.2), that the BKFACT4 algorithm requires $O(4n^2m)$

arithmetic operations.

Program 16 of Appendix I gives the Fortran implementation of the BKFACT4 algorithm.

The Spectral Resolution Approach

An alternative method of solving the skew symmetric block periodic tridiagonal matrix system (5.4.7) is by the matrix decomposition approach (Buzbee et al (1970)) outlined in algorithm (5.1). The algorithm proceeds as follows:

(1) Determine the eigenvectors Q of B and the associated eigenvalues of B using the formula in (5.4.18) and (5.4.17) respectively.

(2) Compute the modified right-hand side vector $\bar{\underline{d}}_j$ by multiplying each row of $\underline{d}$ by $Q^T$ to obtain,

$$\bar{\underline{d}}_j = Q^T \underline{d}_j \quad , \; j=1,2,\ldots,m. \tag{5.4.20}$$

(3) Re-order the sub-vectors $\bar{\underline{d}}_j$ by vertical lines instead of horizontal lines to generate the sub-vectors $\hat{\underline{d}}_i$ and then solve the n set of tridiagonal matrix systems,

$$\Gamma_i \hat{\underline{u}}_i = \hat{\underline{d}}_i \quad , \; i=1,2,\ldots,n \tag{5.4.21}$$

where $\Gamma_i$ is given by,

$$\Gamma_i = \begin{bmatrix} \lambda_i & 1 & & & -1 \\ -1 & \lambda_i & 1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & 1 \\ 1 & & & -1 & \lambda_i \end{bmatrix} \tag{5.4.22}$$

and $\hat{\underline{u}}_i$, $\hat{\underline{d}}_i$ are respectively of the form

$$\hat{\underline{u}}_i = \begin{bmatrix} u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,m} \end{bmatrix} \quad \text{and} \quad \hat{\underline{d}}_i = \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ \vdots \\ d_{i,m} \end{bmatrix} \tag{5.4.23}$$

(4) Next, we reorder the array $\hat{\underline{u}}_i$ by horizontal lines instead of vertical lines to generate $\bar{\underline{u}}_j$ and then compute the final solution vector,

$$\underline{u}_j = Q\underline{\bar{u}}_j \ , \quad j=1,2,\ldots,m. \tag{5.4.24}$$

Each of the n sets of tridiagonal linear systems (5.4.22) can be solved by using, for example, the fast PQFACT4 algorithm (3.9).

Numerical Results

We consider the heat conduction equation,

$$\frac{\partial^2 U(x,t)}{\partial x^2} - \frac{\partial U(x,t)}{\partial t} = -\sin x(\sin t+\cos t) \tag{5.4.25}$$

in the semi-infinite region $R=\{0\leq x\leq \pi,\ t>0\}$ (5.4.26)

satisfying the boundary condition,

$$U(0,t) = U(\pi,t) = 0 \ , \quad t>0 \tag{5.4.27}$$

and the periodic condition,

$$U(x,2\pi) = U(x,0) \ , \quad 0\leq x\leq \pi \ . \tag{5.5.28}$$

Because of symmetry with respect to $x=\pi/2$, the solution in the range $0\leq x\leq \pi/2$ is repeated in the range $\pi/2\leq x\leq \pi$.

For a mesh spacing $\Delta x=h=\pi/21$, and $\Delta t=k=2\pi/20$, the computed solution $\underline{u}$ of the periodic heat equation (5.5.25)-(5.5.28) at specified discrete values of x and t, obtained by the solution of the (400×400) matrix equation (5.4.7), using both the BKFACT4 and the spectral decomposition algorithmic methods, are given in Table (5.4). From this table, it can be seen that the results obtained from both algorithms agree up to 10 decimal places of accuracy; which suggests that both algorithms have the same order of accuracy.

The analytical solution of the model problem under discussion is not known and hence we are not able to compare the computed solution with any known exact solution.

However, by substituting the values of the computed solution at any arbitrary mesh point (i,j) and its 4 neighbouring points into the finite difference scheme (5.4.5) i.e.

$$\psi_{i,j} = -u_{i,j-1}-2ru_{i-1,j}+4ru_{i,j}-2ru_{i+1,j}+u_{i,j+1}+2kF_{i,j} = 0 \tag{5.4.29}$$

The solution of the periodic heat conduction equation (5.5.25)-(5.5.28) using boundary-value technique where $\Delta x=h=\pi/21$, $\Delta t=k=2\pi/20$

| x / t | | 2h | 4h | 6h | 8h | 10h |
|---|---|---|---|---|---|---|
| k | a | 0.17659856568 | 0.33750557379 | 0.46542373059 | 0.55772038321 | 0.59746113232 |
| | b | 0.17659856566 | 0.33750557366 | 0.46842373051 | 0.55772038315 | 0.59746113222 |
| 2k | a | 0.28356541049 | 0.54193478984 | 0.75215088487 | 0.89553507300 | 0.95934703988 |
| | b | 0.28356541045 | 0.54193478976 | 0.75215088976 | 0.89553507286 | 0.95934703973 |
| 3k | a | 0.28221990655 | 0.53936333588 | 0.74858196585 | 0.89128580305 | 0.95479498529 |
| | b | 0.28221990651 | 0.53936333581 | 0.74858196577 | 0.89128580294 | 0.95479498513 |
| 4k | a | 0.17307599062 | 0.33077341993 | 0.45908017931 | 0.54659565011 | 0.58554369869 |
| | b | 0.17307599060 | 0.33077341990 | 0.45908017926 | 0.54659565004 | 0.58554369881 |
| 5k | a | -0.00217707108 | -0.00416069985 | 0.00577463218 | -0.00687546308 | -0.00736537894 |
| | b | -0.00217707108 | -0.00416069985 | 0.00577463218 | -0.00687546308 | -0.00736537894 |
| 6k | a | 0.17659856563 | -0.33750557371 | -0.46842373045 | -0.55772038307 | -0.59746113214 |
| | b | -0.17659856561 | -0.33750557368 | -0.46842373040 | -0.55772038302 | -0.59746113208 |
| 7k | a | -0.28356541046 | -0.54193478977 | -0.75215088476 | -0.89553507286 | -0.95934703977 |
| | b | -0.28356541046 | -0.54193478968 | -0.75215088466 | -0.89553507274 | -0.95934703961 |
| 8k | a | -0.28221990648 | -0.53936333576 | -0.74858196568 | -0.89128580285 | 0.95479498509 |
| | b | -0.28221990643 | -0.53936333567 | -0.74858196557 | -0.89128580271 | 0.95479498492 |
| 9k | a | -0.17307599049 | -0.33077341970 | -0.45908017899 | -0.54659564971 | -0.58554369827 |
| | b | -0.17307599047 | -0.33077341966 | -0.45908017893 | -0.54659564965 | -0.58554369819 |
| 10k | a | 0.00217707123 | 0.00416070013 | 0.00577463258 | 0.00687546357 | 0.00736537945 |
| | b | 0.00217707123 | 0.00416070013 | 0.00577463258 | 0.00687546357 | 0.00736537945 |

TABLE 5.4

*a=BKFACT4 algorithm*

*b=Spectral Decomposition method*

where $F(x,t)=-\sin x(\sin t+\cos t)$, it is found that the computed solutions satisfy the equation (5.4.29), as shown in Table (5.5), to within $10^{-5}$, which is a good accuracy to attain with the $O(h^2+k^2)$ truncation error of the finite difference approximation.

Random checks on the solution of equation (5.5.25)-(5.5.28) at arbitrary points (i,j)

for $h=\Delta x=\pi/21$, $k=\Delta t=2\pi/20$, $r=k/h^2$

| Mesh point (i,j) | $U_{i,j-1}$ | $U_{i-1,j}$ | $U_{i,j}$ | $U_{i+1,j}$ | $U_{i,j+1}$ | $\psi_{i,j}$ (5.4.29) |
|---|---|---|---|---|---|---|
| (1,1) | 0.00110083 | 0.0000000 | 0.475247696 | 0.093987914 | 0.089296652 | 0 .00001795 |
| (4,6) | -0.568472066 | 0.415431649 | 0.539363335 | 0.651246536 | 0.457457964 | 0.000016658 |

TABLE 5.5

## 5.5 AN APPLICATION OF A CYCLIC LOWER TRIANGULAR BLOCK MATRIX IN THE SOLUTION OF A PERIODIC PARABOLIC PROBLEM

We consider here two fast methods for the solution of the cyclic lower triangular block matrix equation arising from the classical implicit finite difference representation of the steady-state solution of the parabolic one-space dimensional heat conduction equation with periodic conditions. This problem has been considered by Tee (1964) in which a method of successive over-relaxation (S.O.R.) was developed for the iterative solution of the discretised problem. Later, the problem was further considered in Osborne (1965) in which a direct method, based on a special factorisation of a cyclic lower triangular matrix, was introduced. Here, we present two alternative direct methods. The first is a block extension of the elimination strategy introduced in the development of the point-form algorithm (3.2). Mathematically, the approach is equivalent to Gaussian elimination without pivoting (since pivoting for size is shown to be unnecessary for a stable solution of the problem under discussion). The other method is an extension of the spectral resolution approach for which we introduce a new fast algorithm for the solution of the resulting decomposed simple point-form cyclic lower triangular matrix systems.

Now, we consider the periodic parabolic heat equation defined in (5.4.1)-(5.4.4), i.e.,

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial U}{\partial t} = F(x,t) \qquad (5.5.1a)$$

in the semi-infinite strip $R:\{0 \leq x \leq a,\ t \geq 0\}$

subject to the boundary conditions,

$$U(0,t) = f(t), \quad t \geq 0$$

and $$U(a,t) = g(t), \quad t \geq 0 \qquad (5.5.1b)$$

where f and g are periodic with period T, and

$$U(x,t+T) = U(x,t) \quad 0 \leq x \leq a,\ t > 0. \qquad (5.5.1c)$$

Let us approximate (5.5.1a) by the finite difference scheme,

$$\frac{(U_{i-1,j+1}-2U_{i,j+1}+U_{i+1,j+1})}{h^2} - \frac{(U_{i,j+1}-U_{i,j})}{k} + O(k+h^2) = F_{i,j}$$

which, on following the usual notations introduced in the last section and neglecting the $O(k^2+kh^2)$ truncation error term, gives the difference equation

$$-rU_{i-1,j+1}+(1+2r)U_{i,j+1}-rU_{i+1,j+1}-U_{i,j} = -kF_{i,j} \qquad (5.5.2)$$

where $r=k/h^2$.

By adopting a row-wise ordering of the mesh points in the truncated rectangular region $R=\{0\leq x\leq a, 0\leq t\leq T\}$ the totality of the difference equation (5.5.2) leads to the m-cyclic block matrix equation,

$$\begin{bmatrix} P & & & -I \\ -I & P & & \\ & \ddots & \ddots & \\ 0 & & -I & P \end{bmatrix} \begin{bmatrix} \underline{u}_0 \\ \underline{u}_1 \\ \vdots \\ \underline{u}_{m-1} \end{bmatrix} = \begin{bmatrix} \underline{h}_0 \\ \underline{h}_1 \\ \vdots \\ \underline{h}_{m-1} \end{bmatrix} \qquad (5.5.3)$$

where $\underline{u}_j$ (the approximation of $\underline{U}_j$) and $\underline{h}_j$ are written as

$$\underline{u}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix} \quad \text{and} \quad \underline{h}_j = \begin{bmatrix} rf_j-kF_{1,j} \\ -kF_{2,j} \\ \vdots \\ -kF_{n-1,j} \\ rg_j-kF_{n,j} \end{bmatrix}, \quad j=0,1,\ldots,m-1. \qquad (5.5.4)$$

In addition, P is the $(n\times n)$ tridiagonal matrix of the form,

$$P = \begin{bmatrix} 1+2r & -r & & \\ -r & 1+2r & -r & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & -r & 1+2r \end{bmatrix} \qquad (5.5.5)$$

which possesses a complete set of eigenvalues,

$$\Lambda_p = \text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$$

and the associated matrix of orthogonal eigenvectors, Q, such that, $PQ=Q\Lambda_p$.

The matrix P is non-singular (i.e. $P^{-1}$ exists) since its eigenvalues, $\lambda_i$, given by,

$$\lambda_i = 1+4r\sin^2(i\pi/2(n+1)), \quad i=1,2,\ldots,n, \tag{5.5.6}$$

are greater than unity.

Hence, the system (5.5.3) can be normalised by multiplying it by the block diagonal matrix,

$$D = \begin{bmatrix} P^{-1} & & & \\ & P^{-1} & & 0 \\ & & \ddots & \\ & 0 & & P^{-1} \end{bmatrix} \tag{5.5.7}$$

to give,

$$\begin{bmatrix} I & & & -N \\ -N & I & & \\ & \ddots & \ddots & \\ 0 & & -N & I \end{bmatrix} \begin{bmatrix} \underline{w}_1 \\ \underline{w}_2 \\ \vdots \\ \underline{w}_m \end{bmatrix} = \begin{bmatrix} \underline{d}_1 \\ \underline{d}_2 \\ \vdots \\ \underline{d}_m \end{bmatrix} . \tag{5.5.8}$$

where,

$$N = P^{-1}, \quad \underline{d}_j = P^{-1}\underline{h}_{j-1} \ , \ j=1,2,\ldots,m. \tag{5.5.9}$$

and $\underline{w}_j$ is defined as $\underline{u}_{j-1}$.

For a stable solution of the m-cyclic block matrix equation (5.5.8) by any form of Gaussian elimination method which does not include a pivoting strategy, it is necessary to have the norm of the matrix N less than unity in magnitude.

Now, since the matrix P is symmetric with its eigenvalues (see 5.5.6) greater than unity, it is then easy to show, by using Theorem (2.1), that the spectral norm of P is greater than unity,

$$\text{i.e.,} \quad ||P||_2 > 1 \ .$$

Hence, it follows immediately that

$$||N||_2 = ||P^{-1}||_2 < 1. \tag{5.5.10}$$

Thus, since the norm of N is bounded and independent of r, the mesh

ratio, one can immediately conclude that a block algorithm, for the steady state solution of the implicit finite difference equation (5.5.8), in which the matrix N is used as a multiplier in an elimination process, is guaranteed to give a stable solution without the need to apply a pivoting strategy.

A fast algorithmic solution of the m-cyclic system

We consider first the simple point form of the system (5.5.8), i.e., the m-cyclic matrix equation,

$$\begin{bmatrix} 1 & & & & -\ell \\ -\ell & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ 0 & & & -\ell & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ \vdots \\ z_m \end{bmatrix} \qquad (5.5.11)$$

where $\ell$ is assumed, for stability reasons, to be less than unity.

It can be immediately recognised that the matrix equation (5.5.11) is of the same structure as system (3.2.18) in Chapter 3, with all the sub-diagonal elements $\ell_i$ (i=1,2,...) replaced by $-\ell$. The algorithmic solution of (5.5.11) can thus be derived as a simplified variant of equations (3.2.20)-(3.2.21), to give, for $|\ell|<1$, the following stable fast method:

By defining the quantities,

$$\phi_k = -\ell^k \quad , \; k=1,2,\ldots,m, \qquad (5.5.12)$$

and then modifying the right-hand side vector $\underline{z}$ to the form,

$$z_1' = z_1$$

and

$$z_k' = z_k + \ell z_{k-1} \quad , \quad k=2,3,\ldots,m, \qquad (5.5.13)$$

we obtain the final solution vector $\underline{x}$ of (5.5.11) as,

$$x_m = z_m'/(1+\phi_m)$$

and

$$x_k = z_k' - \phi_k x_m \quad , \quad k=m-1,m-2,\ldots,1. \qquad (5.5.14)$$

Since we have shown that the norm of the matrix N is less than unity

then it becomes feasible to develop a stable compact method for the solution of the m-cyclic block matrix system (5.5.8) by generalising the above point-form algorithm (5.5.12)-(5.5.14).

We shall refer to the resulting generalised block form (which applies to a block m-cyclic matrix equation of the form (5.5.8)) as the BLOMC algorithm outlined below as follows:

The BLOMC Algorithm

First, we define the following auxiliary sub-vectors as,

$$\underline{d}'_1 = \underline{d}_1$$

$$\underline{d}'_j = \underline{d}_j + N\underline{d}'_{j-1} \quad , \quad j=2,3,\ldots,m. \tag{5.5.15}$$

Then, the solution sub-vectors $\underline{w}_j$ are obtained as,

$$\underline{w}_m = (I-N^m)^{-1}\underline{d}'_m \tag{5.5.16a}$$

and

$$\underline{w}_j = \underline{d}'_j - N^j\underline{w}_m \quad , \quad j=1,2,\ldots,m-1. \tag{5.5.16b}$$

provided that $(I-N^m)$ is non-singular; a condition which is always satisfied since $||N||_2$ has been shown in (5.5.10) to be less than unity.

The matrix P as given in (5.5.5) is a symmetric tridiagonal matrix and therefore has a complete set of theoretically known eigenvalues $\Lambda_p=\text{diag}(\lambda_1,\lambda_2,\ldots\lambda_n)$ and a corresponding fundamental matrix of eigen-vectors Q. It is therefore possible to introduce the similarity normal form of P in order to obtain a computationally simpler form of the BLOMC algorithm.

Hence, by writing N as in (5.3.20), i.e.,

$$N = Q\Lambda_N Q^T \text{ where } \Lambda_N = \Lambda_P^{-1}$$

we have, in place of (5.5.16a), for example, the alternative form,

$$\underline{w}_m = Q\bar{\Lambda}_N Q^T \underline{d}'_m \tag{5.5.17}$$

where

$$\bar{\Lambda}_N = [\text{diag}(1-1/\lambda_1^m,\ 1-1/\lambda_2^m,\ldots 1-1/\lambda_n^m)]^{-1} .$$

Similarly, instead of (5.5.16b) we can write,

$$\underline{w}_j = \underline{d}'_j - Q\Lambda_N^j Q^T \underline{w}_m \quad , \quad j=m-1,m-2,\ldots 1$$

which on pre-computing the vector $\underline{v}=Q^T\underline{w}_m$ becomes,

$$\underline{w}_j = \underline{d}'_j - Q\Lambda_N^j \underline{v} \quad , \quad j=m-1,m-2,\ldots,1. \tag{5.5.18}$$

Altogether, the BLOMC algorithm (5.5.15)-(5.5.16) can be implemented, using the normal form simplifications as indicated, in $O(2n^2m)$ arithmetic operations. A Fortran program for this algorithm is given as Program 16 of Appendix I.

The spectral resolution technique (section 5.2) can be applied as an alternative scheme in the solution of the m-cyclic block matrix system (5.5.8) in the following manner:

(1) Determine the eigenvalues $\Lambda_N=\text{diag}(\mu_1,\mu_2,\ldots,\mu_n)$ of N and the corresponding matrix of eigenvectors, Q. Since $N=P^{-1}$, both N and P are known to have a common matrix of eigenvectors Q. The set of eigenvalues of N, i.e.,

$$\Lambda_N = \Lambda_P^{-1} = \text{diag}(1/\lambda_1, 1/\lambda_2, \ldots, 1/\lambda_n) ,$$

are immediately obtained from $\lambda_i$, the theoretically known eigenvalues of P.

(2) Next, we compute the intermediate vectors,

$$\tilde{\underline{d}}_j = Q^T \underline{d}_j \; (= \Lambda_P^{-1} Q^T \underline{h}_{j-1} \text{ on using (5.5.9))}, \quad j=1,2,\ldots,m.$$

(3) Then, we solve a set of n linear m-cyclic matrix systems,

$$\Gamma_i \hat{\underline{w}}_i = \hat{\underline{d}}_i , \quad i=1,2,\ldots,n \tag{5.5.19}$$

where $\hat{\underline{w}}_i$ and $\hat{\underline{d}}_i$ are vectors structured as those defined in (5.4.23), and $\Gamma_i$ is the m-cyclic matrix given by,

$$\Gamma_i = \begin{bmatrix} 1 & & & -\mu_i \\ -\mu_i & 1 & & \\ & \ddots & \ddots & \\ 0 & & -\mu_i & 1 \end{bmatrix}_{(m\times m)} . \tag{5.5.20}$$

(4) Finally, we compute the solution subvectors,

$$\underline{w}_j = Q\tilde{\underline{w}}_j , \quad j=1,2,\ldots,m. \tag{5.5.21}$$

The formulae (5.5.12)-(5.5.14) provide a fast solution of the simple linear systems (5.5.19); and by neglecting the computation of the eigenvalues and eigenvectors the separation of variables technique as outlined requires $O(2n^2m+2nm)$ arithmetic operations for the solution of the given m-cyclic system (5.5.8).

The methods presented here provide a more compact form for the solution of the implicit periodic parabolic heat conduction problem in one space dimension than the given method of successive over-relaxation (S.O.R.) proposed for the same problem by Tee (1964), and the direct method proposed by Osborne (1965).

## 5.6 THE SOLUTION OF THE FOURTH BOUNDARY VALUE SELF-ADJOINT PROBLEM FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS

In the numerical solution (by a marching technique) of the implicit finite difference equations derived from the quasi-linear self-adjoint parabolic partial differential equations with periodic boundary conditions in the space dimension, there arises the need to solve repeatedly systems of linear equations in which the cofficient matrix is the general circulant tridiagonal matrix of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \qquad (5.6.1)$$

where A is assumed to be positive definite and diagonally dominant, i.e., $|b_i| \geq |a_i| + |c_i|$, $i=1,2,\ldots,n$.

In section (3.2) of this thesis, we discussed various generalisations of algorithmic methods for the fast solution of the general tridiagonal matrix system of the form (5.6.1). Here, we consider the solution of a model parabolic fourth boundary-value problem in which these fast methods become readily applicable.

### Problem Definition

We consider the following general diffusion equation with periodic spatial boundary condition, i.e.,

$$\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}\left(K(x)\frac{\partial U}{\partial x}\right) + \Psi(x,t) \ , \qquad (5.6.2)$$

in the semi-infinite region, $R=\{0 \leq x \leq \ell,\ t \geq 0\}$ where the function $U(x,t)$ is required to satisfy the initial boundary condition,

$$U(x,0) = f(x) \qquad (5.6.3)$$

and the periodic boundary conditions,

$$U(0,t) = U(\ell,t)$$
$$\frac{\partial U}{\partial x}(0,t) = \frac{\partial U}{\partial x}(\ell,t) \quad . \tag{5.6.4}$$

A typical physical realisation of the above problem is the cooling of a thin circular ring whose circumference is represented by $\ell$..

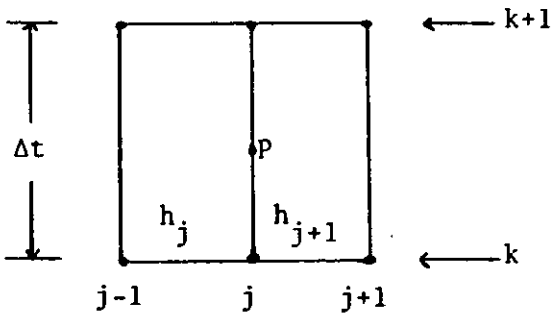
For a numerical solution of (5.6.2)-(5.6.4) we consider a set of non-uniform mesh points $x_j$ such that

$$0 = x_1 < x_2 \ldots < x_n = \ell$$

and then define the mesh sizes $h_j = x_j - x_{j-1}$ and $t_k = k\Delta t$.

Further, we denote $x_{j+\frac{1}{2}}$ for $x_j + \frac{1}{2}h_{j+1}$, $K_{j+\frac{1}{2}}$ for $K(x_{j+\frac{1}{2}})$, and $t_{k+\frac{1}{2}} = t_k + \frac{1}{2}\Delta t$.

Next we approximate the p.d.e. (5.6.2) at the point (j,k+1) by the stable Crank-Nicolson finite difference implicit formula (see 2.2.16) in which symmetry in the discretisation process is introduced by writing all the finite differences about the point $P(x_j, t_{k+\frac{1}{2}})$, half-way between the known time level k and the unknown time level (k+1) as illustrated in Figure (5.3).




The Computational Molecule for the Crank-Nicolson Scheme

FIGURE 5.3

Hence we have,

$$\left(\frac{\partial U}{\partial t}\right)_{j,k+1} = \frac{(U_{j,k+1} - U_{j,k})}{\Delta t} + O(\Delta t) \tag{5.6.5}$$

and

$$\frac{\partial}{\partial x}\left[K(x)\frac{\partial U}{\partial x}\right]_{j,k+1} = \frac{\left[K_{j+\frac{1}{2}}\left(\frac{\partial U}{\partial x}\right)_{j+\frac{1}{2},k+1} - K_{j-\frac{1}{2}}\left(\frac{\partial U}{\partial x}\right)_{j-\frac{1}{2},k+1}\right]}{(h_j + h_{j+1})/2}$$

which gives, at the point $(j,k+\frac{1}{2})$,

$$\frac{\partial}{\partial x}\left[K(x)\frac{\partial U}{\partial x}\right]_{j,k+\frac{1}{2}} = \left[\frac{1}{2}\left\{K_{j+\frac{1}{2}}\left(\frac{U_{j+1,k+1}-U_{j,k+1}}{h_{j+1}}\right) + K_{j+\frac{1}{2}}\left(\frac{U_{j+1,k}-U_{j,k}}{h_{j+1}}\right)\right\}\right.$$

$$\left. - \frac{1}{2}\left\{K_{j-\frac{1}{2}}\left(\frac{U_{j,k+1}-U_{j-1,k+1}}{h_j}\right) + K_{j-\frac{1}{2}}\left(\frac{U_{j,k}-U_{j-1,k}}{h_j}\right)\right\}\right] \Big/ \left(\frac{h_j+h_{j+1}}{2}\right) + O(h_j h_{j+1}) \tag{5.6.6}$$

On omitting the truncation error terms and substituting the difference equations (5.6.5) and (5.6.6) into (5.6.2), we obtain at the point (j,k+1) the following equation,

$$\frac{h_j+h_{j+1}}{2}\left[\frac{U_{j,k+1}-U_{j,k}}{\Delta t}\right] = K_{j+\frac{1}{2}}\left(\frac{U_{j+1,k+1}-U_{j,k+1}}{2h_{j+1}}\right) - K_{j-\frac{1}{2}}\left(\frac{U_{j,k+1}-U_{j-1,k+1}}{2h_j}\right)$$

$$+K_{j+\frac{1}{2}}\left(\frac{U_{j+1,k}-U_{j,k}}{2h_{j+1}}\right) - K_{j-\frac{1}{2}}\left(\frac{U_{j,k}-U_{j-1,k}}{2h_j}\right) + \Psi(x_j,(k+\tfrac{1}{2})\Delta t) \tag{5.6.7}$$

$$\text{for } 1 \leq j \leq n.$$

By regrouping the terms, equation (5.6.7) can be written as a series of linear equations of the form,

$$a_j U_{j-1,k+1} + b_j U_{j,k+1} + c_j U_{j+1,k+1} = d_{j,k}\ , \quad 1 \leq j \leq n \tag{5.6.8}$$

where the coefficients $a_j, b_j, c_j$ and $d_{j,k}$ are defined, for $j=1,2,\ldots,n$ as

$$\left.\begin{aligned} a_j &= \frac{-\Delta t K_{j-\frac{1}{2}}}{(h_j+h_{j+1})h_j}\ , \quad c_j = \frac{-\Delta t K_{j+\frac{1}{2}}}{(h_j+h_{j+1})h_{j+1}} \\ b_j &= 1 + \frac{\Delta t}{(h_j+h_{j+1})}\left(\frac{K_{j-\frac{1}{2}}}{h_j} + \frac{K_{j+\frac{1}{2}}}{h_{j+1}}\right) \end{aligned}\right\} \quad j=1,2,\ldots,n \tag{5.6.9}$$

and

$$d_{j,k} = U_{j,k} + \Psi(x_j,(k+\tfrac{1}{2})\Delta t)\frac{2\Delta t}{h_j+h_{j+1}} + \Delta t\ K_{j+\frac{1}{2}}\left(\frac{U_{j+1,k}-U_{j,k}}{(h_j+h_{j+1})h_{j+1}}\right)$$

$$-\Delta t\ K_{j-\frac{1}{2}}\left[\frac{U_{j,k}-U_{j-1,k}}{(h_j+h_{j+1})h_j}\right]\ , \quad j=1,2,\ldots,n. \tag{5.6.10}$$

It is easily verified that

$$|b_j| > |a_j| + |c_j|\ .$$

As a result of the periodic boundary conditions (5.6.4) we have the periodicity relationships,

$$\left.\begin{aligned} U_{n+1,k} &= U_{1,k} \\ K_{n+\frac{1}{2}} &= K_{\frac{1}{2}}\ , \\ h_{n+1} &= h_1 \quad \text{etc.} \end{aligned}\right\} \tag{5.6.11}$$

and

Hence, the difference equations (5.6.8) together with the periodic conditions (5.6.11) lead to the general diagonally dominant matrix system,

$$A\underline{u}_{k+1} = \underline{d}_k \tag{5.6.12}$$

or

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \begin{bmatrix} u_{1,k+1} \\ u_{2,k+1} \\ \vdots \\ u_{n,k+1} \end{bmatrix} = \begin{bmatrix} d_{1,k} \\ d_{2,k} \\ \vdots \\ d_{n,k} \end{bmatrix} \tag{5.6.13}$$

which is to be solved in order to obtain the values $\underline{u}_{k+1}$ on the new line $(k+1)\Delta t$ from the values of $\underline{u}_k$ on the previous line $k\Delta t$. Thus, the process of solving numerically the fourth boundary value problem (5.6.2)-(5.6.4) by a marching technique, which employs the Crank-Nicolson finite difference scheme, consists essentially of solving repeatedly the general matrix equation defined in (5.6.13).

The generalised cyclic factorisation method in the solution of the Crank-Nicolson finite difference equation of the Fourth-Boundary Value Problem

The coefficient matrix A arising from the discretised fourth boundary-value p.d.e. remains unchanged as the marching procedure is advanced from one time-interval to the next. There is therefore some attraction for a factorisation method of solution since the coefficient matrix can be predetermined in its factorised form only once, thus eliminating, for subsequent solution steps, the most computationally tedious part of the method. On this basis, the generalised cyclic factorisation algorithm (i.e. the PQFACT algorithm (3.2), Chapter 3) offers an attractive fast method for the repeated solution of the system (5.6.13), since in that case, the factorisation and the resultant evaluation of a periodic continued fraction which forms the tedious part of the algorithm need be performed only

once; and thereafter, only 5n multiplications and 5n additions are required by the PQFACT algorithm for a further solution on a new line $k\Delta t$ $(k=2,3,\ldots)$.

Numerical Results

The direct method outlined in this section for the solution of the Crank-Nicolson finite difference equation using the PQFACT solver algorithm was programmed for the I.C.L 1904S computer at Loughborough University and used to solve the following heat conduction equation:

$$\frac{\partial U}{\partial t} - \frac{\partial^2 U}{\partial x^2} = 10(1-x)xt \tag{5.6.14}$$

with the initial condition,

$$U(x,0) = x(1-x) \ , \quad 0 \leq x \leq 1 \tag{5.6.15}$$

and the periodic boundary condition,

$$\left.\begin{aligned} U(0,t) &= U(1,t), \quad t \geq 0 \\ \text{and} \quad \frac{\partial U}{\partial x}(0,t) &= \frac{\partial}{\partial x}U(1,t), \quad t \geq 0. \end{aligned}\right\} \tag{5.6.16}$$

The equations (5.6.14)-(5.6.16) have the following analytical solution (Evans (1971A))

$$U(x,t) = \frac{(1+5t^2)}{6} - \frac{5}{8}\sum_{n=1}^{\infty}\frac{\cos 2n\pi x}{n^6\pi^6}\{4n^2\pi^2 t - 1 + e^{-4n^2\pi^2 t}\} - \sum_{n=1}^{\infty}\frac{e^{-4n^2\pi^2 t}}{n^2\pi^2}\cos 2n\pi x. \tag{5.6.17}$$

Results are compared in Table (5.5) for the formal solution given by (5.6.17) against the Crank-Nicolson method using the PQFACT solver algorithm for mesh sizes of 0.01 in the x-axis and 0.005 in the t-direction.

Solution of the self-adjoint diffusion equation with periodic condition

t=1.0

| x | Finite Difference Solution Using PQFACT Solver Algorithm | Analytical Solution Using (5.6.17) |
|---|---|---|
| 0.0 | 0.969709 | 0.972884 |
| 0.1 | 0.976697 | 0.979501 |
| 0.2 | 0.991226 | 0.993748 |
| 0.3 | 1.006441 | 1.008762 |
| 0.4 | 1.017494 | 1.019703 |
| 0.5 | 1.021499 | 1.023667 |

x=0.5

| t | Finite Difference Solution Using PQFACT Solver Algorithm | Analytical Solution Using (5.6.17) |
|---|---|---|
| 0.1 | 0.178395 | 0.178758 |
| 0.3 | 0.247804 | 0.248318 |
| 0.5 | 0.385735 | 0.386512 |
| 0.7 | 0.590164 | 0.591373 |
| 0.9 | 0.861092 | 0.862901 |

TABLE 5.5

## 5.7 THE SOLUTION OF LINEAR PERIODIC PARABOLIC P.D.E.'S IN TWO-SPACE DIMENSIONS BY FAST A.D.I. METHODS

In the numerical solution of linear periodic parabolic partial differential equations with constant coefficients involving two-space dimensions by the alternating direction implicit (A.D.I.) methods there occurs the problem of solving repeatedly cyclic tridiagonal matrix equations for which the fast algorithms presented in section (3.3) are particularly applicable.

In the A.D.I. method developed by Peaceman and Rachford (1955), we use two forms of equations alternately in successuve time steps; in the first step the finite difference equations are implicit in the x-direction and explicit in the y-direction; and in the second step the directions are interchanged.

### Problem Definition

We consider the solution of the model problem,

$$\frac{\partial U}{\partial t} = \tau\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right), \qquad \tau = \text{a constant} \tag{5.7.1}$$

in the region $R' = (x,y,t) \in R \times (t>0)$, (5.7.2)

(where $R=\{0\leq x,y\leq 1\}$ is a closed, connected square region in the x-y plane with the boundary $\partial R$) under the periodic boundary conditions in both the x- and y- directions, i.e., we have,

$$\left.\begin{aligned} U(0,y,t) &= U(1,y,t) \\ (\partial U/\partial x)(0,y,t) &= (\partial U/\partial x)(1,y,t) \end{aligned}\right\} \; 0\leq y\leq 1$$

$$\left.\begin{aligned} U(x,0,t) &= U(x,1,t) \\ \frac{\partial U}{\partial y}(x,0,t) &= \frac{\partial U}{\partial y}(x,1,t) \end{aligned}\right\} \; 0\leq x\leq 1 \qquad \left.\begin{array}{l}\\ \\ \\ \\ \end{array}\right\} \; t>0 \tag{5.7.3}$$

and the initial boundary condition,

$$U(x,y,0) = f(x,y) \;, \quad 0\leq x,y\leq 1 \tag{5.7.4}$$

where $f(x,y)$ is a known function of x and y.

For the numerical solution of (5.7.1)-(5.7.4), we consider the mesh points $(x_j,y_j)=(ih,jh)$ for $1\leq i,j\leq n$ where $h=\Delta x=\Delta y=1/n$, $\ell=\Delta t$ and then

approximate (5.7.1) at each node of (RU∂R) by the A.D.I. scheme in which two forms of equations are used alternately in successive time steps, i.e.,

$$\frac{U_{i,j,2k+1}-U_{i,j,2k}}{\ell} = \left(\frac{\tau}{h^2}\right)\left(\delta_x^2 U_{i,j,2k+1}+\delta_y^2 U_{i,j,2k}\right)+O(h^2+\ell) \qquad (5.7.5)$$

and

$$\frac{U_{i,j,2k+2}-U_{i,j,2k+1}}{\ell} = \left(\frac{\tau}{h^2}\right)\left(\delta_x^2 U_{i,j,2k+1}+\delta_y^2 U_{i,j,2k+2}\right)+O(h^2+\ell) \qquad (5.7.6)$$

$i,j=1,2,\ldots,n$ and $k\geq 0$.

where the central difference operator $\delta^2$ is defined by,

$$\delta_x^2 U_{i,j,2k+1} = U_{i-1,j,2k+1}-2U_{i,j,2k+1}+U_{i+1,j,2k+1}, \text{ etc.}$$

The stability of this finite difference procedure is well established (Peaceman and Rachford (1955)).

By setting $\sigma=\ell\tau/h^2$ and omitting the truncation error terms, both (5.7.5) and (5.7.6) can be simplified to give the respective alternative forms,

$$\begin{aligned} &-\sigma U_{i-1,j,2k+1}+(1+2\sigma)U_{i,j,2k+1}-\sigma U_{i+1,j,2k+1} \\ &= \sigma U_{i,j-1,2k}+(1-2\sigma)U_{i,j,2k}+\sigma U_{i,j+1,2k}, \quad 1\leq i,j\leq n, k\geq 0 \end{aligned} \qquad (5.7.7)$$

and

$$\begin{aligned} &-\sigma U_{i,j-1,2k+2}+(1+2\sigma)U_{i,j,2k+2}-\sigma U_{i,j+1,2k+2} \\ &= \sigma U_{i-1,j,2k+1}+(1-2\sigma)U_{i,j,2k+1}+\sigma U_{i+1,j,2k+1}, \quad 1\leq i,j\leq n;\ k\geq 0. \end{aligned} \qquad (5.7.8)$$

Both equations (5.7.7) and (5.7.8) together with the periodic boundary conditions (5.7.3) yield respectively the two sets of compound matrix equations,

$$\underline{GU}_{2k+1} = H\underline{U}_{2k} \qquad (5.7.9)$$

and

$$G\underline{\tilde{U}}_{2k+2} = H\underline{\tilde{U}}_{2k+1} \qquad (5.7.10)$$

where G and H are compound matrices of order (n×n) defined as,

$$G = \begin{bmatrix} A & & & 0 \\ & A & & \\ & & \ddots & \\ 0 & & & A \end{bmatrix}_{(n\times n)}, \quad H = \begin{bmatrix} (1-2\sigma)I & \sigma I & & & \sigma I \\ \sigma I & (1-2\sigma)I & \sigma I & & \\ & \ddots & \ddots & \ddots & \\ & & \sigma I & (1-2\sigma)I & \sigma I \\ \sigma I & & & \sigma I & (1-2\sigma)I \end{bmatrix}_{(n\times n)}, \qquad (5.7.11)$$

and A is given by,

$$A = \begin{bmatrix} 1+2\sigma & -\sigma & & & -\sigma \\ -\sigma & 1+2\sigma & -\sigma & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\sigma \\ -\sigma & & & -\sigma & 1+2\sigma \end{bmatrix}_{(n\times n)} . \qquad (5.7.12)$$

The unknown compund vectors $\underline{U}_p$ and $\underline{\tilde{U}}_p$ of $\underline{U}$ and $\underline{\tilde{U}}$ respectively are ordered row-wise and column-wise respectively and given by,

$$\underline{U}_p^T = [U_{1,1,p}, U_{2,1,p}, \ldots U_{n,1,p}, U_{1,2,p}, U_{2,2,p}, \ldots, U_{n,2,p}, \ldots U_{1,n,p}, U_{2,n,p} \ldots U_{n,n,p}]$$

and

$$\underline{\tilde{U}}_p^T = [U_{1,1,p}, u_{1,2,p}, \ldots U_{1,n,p}, U_{2,1,p}, U_{2,2,p}, \ldots U_{2,n,p}, \ldots U_{n,1,p}, U_{n,2,p}, \ldots U_{n,n,p}],$$

For a known solution at the $(2k)^{th}$ time interval, (k=0,1,...), we have to solve the n independent systems of equation given by (5.7.9) along each of the rows in order to advance the solution over the square region R to the $(2k+1)^{th}$ time interval. Similarly, the n independent systems of equations given by (5.7.10) are to be solved along each of the columns in order to advance the solution from the $(2k+1)^{th}$ to $(2k+2)^{th}$ time interval.

It is evident from the above discussion that the major computational requirement of this method is the repeated solution of the one-dimensional periodic tridiagonal matrix equations of the form,

$$A\underline{\phi} = \underline{g} \qquad (5.7.13)$$

where A is given by (5.7.12) and $\underline{\phi}$ denotes a solution along a row or a column of R. The repeated solution of the periodic tridiagonal matrix systems can be achieved by the use of the cyclic factorisation algorithms (3.6) or (3.10) (see Chapter 3) whose stability is guaranteed under diagonal dominance conditions of the coefficient matrix.

From the representation of the matrix A in (5.7.12) it follows immediately, by definition, that A is both diagonally dominant and positive definite, and hence the application of the fast cyclic tridiagonal

matrix solver algorithms (3.6) or (3.10) in the A.D.I. solutions along the rows and columns of the network are guaranteed to produce stable solutions. Also, as was indicated earlier, (see section 3.3, Chapter 3) the actual amount of computational effort needed to implement the above mentioned cyclic tridiagonal matrix solver algorithms is 4n multiplications and 4n additions for repeated applications. If the method of Gaussian elimination without pivoting were to be used to solve the periodic matrix equations along each row and column of the A.D.I. network, then 5n multiplications, 4n additions would be required per line for repeated applications (Alberg et al (1967)). Hence, the use of algorithm (3.6) or (3.10) in the solution along the rows and columns of the A.D.I. method can result in some considerable savings of computational effort.

In the next chapter, we shall discuss further the fast A.D.I. methods for the solution of linear parabolic partial differential equations involving two space dimensions for problems with the more usual Dirichlet and Neumann boundary conditions.

# CHAPTER 6

# FAST METHODS FOR THE SOLUTION OF DIRICHLET'S AND NEUMANN'S BOUNDARY VALUE PROBLEMS

## 6.1 INTRODUCTION

In this Chapter, we consider further a number of model problems having Dirichlet's, Neumann's and mixed boundary conditions and then introduce fast methods for their solution.

In section (6.2) the simple one-dimensional Poisson and transport equations are considered for which the fast algorithms presented in Chapter 4 are shown to be particularly attractive for the numerical solution of such frequently occurring problems. Next, the block rectangular factorisation method is introduced in section (6.3) for the solution of the discrete Poisson and other elliptic p.d.e.'s in a rectangular region for which, in special cases, an $O(n^2)$ method is obtained. In section (6.4) the Alternate Direction Implicit (A.D.I.) methods for the solution of the two-dimensional heat conduction equation with Dirichlet's and Neumann's boundary conditions are considered, and the use of the recursive point partitioning algorithmic methods in the line by line solution is shown to offer significant economies over older techniques. Finally, we consider the solution of the biharmonic equation with mixed boundary condition in a rectangular region for which fast solution methods by the matrix decomposition and factorisation techniques are considered.

## 6.2 SOME SIMPLE ONE-DIMENSIONAL PROBLEMS OF MATHEMATICAL PHYSICS

First, we consider the simple Dirichlet's problem for the one-dimensional Poisson equation,

$$\frac{d^2U}{dx^2} = f(x) \quad 0 \leq x \leq 1 \tag{6.2.1}$$

with the boundary condition,

$$U(0) = a \quad \text{and} \quad U(1) = b \tag{6.2.2}$$

where f(x) represents the source function and a and b are given constants.

We divide the interval $0 \leq x \leq 1$ into (n+1) equal sub-intervals each of length h=1/(n+1) and approximate (6.2.1) by the second order formulae,

$$\left.\begin{aligned} \frac{(U_{i-1} - 2U_i + U_{i+1})}{h^2} + O(h^2) &= f_i, \quad i=1,2,\ldots,n \\ \end{aligned}\right\} \tag{6.2.3}$$

where $U_0 = a$ and $U_{n+1} = b$.

On neglecting the $O(h^2)$ truncation error term, the difference equation (6.2.3) leads to the (n×n) matrix system,

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & -1 & \\ 0 & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} a-f_1h^2 \\ -f_2h^2 \\ \vdots \\ -f_{n-1}h^2 \\ b-f_nh^2 \end{bmatrix} \tag{6.2.4}$$

or $\quad Au = d.$

It is easy to verify that for $u \neq 0$,

$$(Au,u) > 0$$

and hence by definition (2.7) the coefficient matrix A is positive definite. Hence, for the solution of the system (6.2.4), the use of the recursive point partitioning method (see CTRPP algorithm (4.9)) gives a stable fast solution in 2n multiplications, n/2 divisions and 5n/2 additions.

### The Singularity Associated with the Neumann Boundary Condition

For Neumann's boundary conditions, i.e.,

$$\left.\begin{aligned} \frac{dU}{dx} &= a, \quad x=0 \\ \text{and} \quad \frac{dU}{dx} &= b, \quad x=1 \end{aligned}\right\} \tag{6.2.5}$$

we can approximate these first derivatives by the finite difference analogues, at the points x=0 and x=1 by

$$\frac{(u_1-u_{-1})}{2h} + O(h) = a \ , \quad \frac{(u_{n+2}-u_n)}{2h} + O(h) = b \qquad (6.2.6)$$

if the auxiliary net region,

$$\{x_{-1} = -h,\ x_0 = 0,\ x_i = ih,\ x_{n+1} = 1,\ x_{n+2} = 1+h\}$$

is considered.

On substituting for $u_{-1}$ and $u_{n+2}$ from (6.2.6) into the difference approximation for (6.2.1), i.e.,

$$\frac{(u_{i-1}-2u_i+u_{i+1})}{h^2} + O(h^2) = f_i \ , \quad i=0,1,\ldots,n+1$$

we obtain, on neglecting the $O(h^2)$ terms, the matrix equation,

$$A\underline{u} = \underline{d} \qquad (6.2.7a)$$

or

$$\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & & 2 & -1 & \\ 0 & & & -1 & 1 & \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n+1} \end{bmatrix} = \begin{bmatrix} -\frac{h^2}{2}f_0 - ah \\ -h^2 f_1 \\ \vdots \\ -\frac{h^2}{2}f_{n+1} + bh \end{bmatrix} \qquad (6.2.7b)$$

The coefficient matrix A is singular with its smallest eigenvalue equal to zero. This can be shown easily by recognising the fact that the associated spectral problem

$$A\underline{u} = \lambda\underline{u}$$

is satisfied by the eigenvector $\underline{u}_0$ (say) whose components are all equal, and the corresponding eigenvalue $\lambda_0=0$. The singularity of the matrix A means that the solvability of the singular system (6.2.7) is only guaranteed under special constraints (which the physics of the problem enables us to impose). One such constraint is to require that the right-hand side vector be orthogonal to $\underline{u}_0$ (which in physical terms is equivalent to imposing a constraint on the sum of the sources $\{d_i\}_{i=0}^{n+1}$. This can be achieved by removing a common factor from the components $d_i$, namely, by replacing $d_i$ with $g_i$ $(=d_i-\tau)$ where $\tau = \sum_{i=0}^{n+1} (\frac{d_i}{n+2})$.

Thereafter, the modified system, $A\underline{u}=\underline{g}$ is then immediately solvable by the CTRPP algorithm (4.9). In the actual realisation of the algorithmic solution of the modified singular system using the CTRPP algorithm it is possible to have in the calculation process, operations of the form '0/0' which should be replaced by an arbitrary constant.

Rank-one Modification Method for Neumann's Singular System

Another alternative method for solving Neumann's singular system (6.2.7) is to rearrange the matrix A as a rank one perturbation of a symmetric cyclic matrix C, i.e.,

$$A = C + \sigma \underline{z}\underline{z}^T \tag{6.2.8}$$

where $\underline{z}^T=(1,0,\ldots,0,-1)$, $\sigma = -1$,

and C is now the non-singular, positive definite cyclic matrix,

$$C = \begin{bmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ -1 & & & -1 & 2 \end{bmatrix} \tag{6.2.9}$$

By using (6.2.8) and the Sherman-Morrison formula (Householder (1964)) the solution of the matrix equation (6.2.7) can be written as a linear combination of the solution of two alternative systems of equations, (see equations (3.2.53) under algorithm 3.5), i.e.,

$$C\underline{v} = \sigma \underline{z} \tag{6.2.10}$$

and

$$C\underline{u} = \underline{d}-\beta\underline{z}, \quad \beta = \frac{\underline{v}^T.\underline{d}}{1+v_1+v_n} \tag{6.2.11}$$

Since C is positive definite the two systems in (6.2.10) and (6.2.11) can now be solved, with stability guaranteed, without pivoting for size. A suitably modified form of the PQFACT1 solver algorithm (3.6) or the Temperton's algorithm (3.11) can now be readily applied to obtain fast solutions of the two cyclic matrix systems (6.2.10) and (6.2.11).

A Simple Transport Equation

The propagation problem of a substance along the particle trajectories may be considered to belong to a class of simple problems of Mathematical Physics which are of frequent occurrence in hydrodynamics, weather forecasting and the theory of radiations. A model problem of this kind is as follows:

Consider the equation,

$$\frac{\partial U}{\partial t} + c\frac{\partial U}{\partial x} = 0, \quad c = \text{constant} \tag{6.2.12}$$

in the domain $R=\{a\leq x\leq b,\ t>0\}$

and satisfying the following initial and boundary conditions,

$$\left.\begin{aligned} U(x,0) &= f(x) \\ U(a,t) &= g_1(t) \\ U(b,t) &= g_2(t) \end{aligned}\right\} \begin{aligned} &\\ &\left.\right\} t>0 \\ &\end{aligned} \tag{6.2.13}$$

where c usually denotes the velocity of propagation of the given profile.

For a numerical solution of (6.2.12) we choose a uniform mesh spacing h along the x-axis such that $h=(b-a)/(n+1)$, and a uniform time increment k along the t-axis. Further, we consider the stable Crank-Nicolson implicit method in which the partial derivative $\frac{\partial U}{\partial x}$ is approximated by the average of its values at times $t=jk$ and $(j+1)k$.

Hence,
$$\left(\frac{\partial U}{\partial x}\right)_i = \frac{1}{2}\left[\frac{\partial U_{i,j+1}}{\partial x} + \frac{\partial U_{i,j}}{\partial x}\right]$$

i.e.,
$$\left(\frac{\partial U}{\partial x}\right)_i = \frac{1}{2}\left[\frac{(U_{i+1,j+1}-U_{i-1,j+1})}{2h} + \frac{(U_{i+1,j}-U_{i-1,j})}{2h}\right] + O(h) \tag{6.2.14}$$

and
$$\left(\frac{\partial U}{\partial t}\right)_i = \frac{1}{k}(U_{i,j+1} - U_{i,j}) + O(k). \tag{6.2.15}$$

On substituting (6.2.14) and (6.2.15) into (6.2.12) and setting $\sigma=ck/4h$, the corresponding finite difference approximation for the equation (6.2.12) becomes, on neglecting the truncation error terms,

$$-\sigma U_{i-1,j+1}+U_{i,j+1}+\sigma U_{i+1,j+1} = \sigma U_{i-1,j}+U_{i,j}-\sigma U_{i+1,j} \tag{6.2.16}$$

which relates the unknown values of U at the $(j+1)k^{th}$ time to the known values of U at the $(jk)^{th}$ time.

Next, by applying equation (6.2.16) at the points $x_i$, i=1,2,...,n on each grid line, we obtain the following linear system to be solved at each time step,

$$A\underline{u}_{j+1} = A^T\underline{u}_j + \underline{h}_j = \underline{d}_j \qquad (6.2.17)$$

where $\underline{u}_j$ is the approximation of $U_j$, A is the (n×n) skew symmetric tridiagonal matrix,

$$A[-\sigma,1,\sigma] = \begin{bmatrix} 1 & \sigma & & & \\ -\sigma & 1 & \sigma & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \sigma \\ 0 & & & -\sigma & 1 \end{bmatrix} \qquad (6.2.18)$$

and

$$\underline{h}_j = \begin{bmatrix} \sigma g_1((j+1)k)+\sigma g_1(jk) \\ 0 \\ \vdots \\ 0 \\ -\sigma g_2((j+1)k)-\sigma g_2(jk) \end{bmatrix} .$$

Derivation of an Algorithmic Solution

In Chapter 4, a fast method was given for the solution of the skew symmetric tridiagonal system of the form $A[-c,b,c]\underline{u}=\underline{d}$, in which, by employing the reversed triangular factorisation and expansion concept, a stable algorithm requiring 4n multiplications and 4n additions was derived (algorithm (4.7)). Stability arguments developed in the analysis of the algorithm are easily shown to apply to the matrix (6.2.18) since the multiplier factor (see 4.3.74)

$$\alpha = 2\sigma/(1+\sqrt{1+4\sigma^2})$$

has its absolute value less than unity for any value of σ. Hence we can immediately apply algorithm (4.7) in a line by line solution of the difference equation (6.2.17) in 5n multiplications, 4n additions per line. Furthermore, the Tridiagonal Recursive Point Partitioning (TRPP) algorithm (4.8) can also

be suitably modified to give a faster solution of (6.2.17) in 5n/2 multiplications, 3n/2 divisions and 5n/2 additions; and only 3n/2 multiplications, n divisions and 3n/2 additions for repeated solutions. The special modified variant of the TRPP algorithm, suitable for a repeated fast solution of the skew-symmetric system of the form, $A[-\sigma,1,\sigma]\underline{u}=\underline{d}$, is summarised below as follows:

Step 1:

First, we initialise the quantities,

$$\beta_1^{(1)} = \beta_n^{(1)} = 1, \qquad d_1^{(1)} = d_1, \qquad d_n^{(1)} = d_n$$

and set $q=n/2$ (n even), $q=(n+1)/2$, (n odd).

Step 2:

Next, we compute the recurrence relation,

$$\left.\begin{aligned} t_{i-1} &= -\sigma/\beta_{i-1}^{(i-1)}, \quad s_{i-1} = -t_{i-1}, \\ \beta_i^{(i)} &= \beta_{n-i+1}^{(i)} = 1-\sigma t_{i-1}, \\ d_i^{(i)} &= d_i^{(i-1)} - t_{i-1} d_{i-1}^{(i-1)}, \\ d_{n-i+1}^{(i)} &= d_{n-i+1}^{(i-1)} - s_{i-1} d_{n-i+2}^{(i-1)} \end{aligned}\right\} \quad \begin{aligned} &i=2,3,\ldots,q-1, \ \text{(n odd)} \\ &i=2,3,\ldots,q, \quad \text{(n even)} \end{aligned}$$

and for n odd only,

$$\beta_q^{(q)} = 1-2\sigma t_{q-1},$$

$$d_q^{(q)} = d_q^{(q-1)} - t_{q-1}(d_{q-1}^{(q-1)} - d_{n-q+2}^{(q-1)}).$$

Step 3:

Then, we obtain the centre elements of the solution vector as,

$$u_q = d_q^{(q)}/\beta_q^{(q)} \quad \text{(n odd)}$$

or

$$\left.\begin{aligned} u_q &= (d_q^{(q)}\beta_{n-q+1}^{(q)} - \sigma d_{n-q+1}^{(q)})/(\beta_{n-q+1}^{(q)}\beta_q^{(q)} + \sigma^2) \\ u_{n-q+1} &= (d_q\beta_{n-q+1}^{(q)} - \sigma d_{n-q+1}^{(q)})/(\beta_{n-q+1}^{(q)}\beta_q^{(q)} + \sigma^2) \end{aligned}\right\} \text{ n even}$$

and finally, the remaining elements of the solution vector become

$$\left.\begin{aligned} u_i &= (d_i^{(i)} - \sigma u_{i+1})/\beta_i^{(i)}, \\ \text{and} \quad u_{n-i+1} &= (d_{n-i+1}^{(i)} + \sigma u_{i-1})/\beta_{n-i+1}^{(i)}. \end{aligned}\right\} \quad i=q-1,q-2,\ldots,1.$$

## 6.3 BLOCK RECTANGULAR FACTORISATION METHOD FOR THE SOLUTION OF THE DISCRETE POISSON EQUATION WITH DIRICHLET'S AND NEUMANN'S BOUNDARY CONDITIONS IN A RECTANGULAR REGION

Many physical problems, such as the calculation of electrostatic potential from a known charge distribution or the stream function from the vorticity distribution, require the solution of the Poisson equation which relates the potential (or stream function) to the source distribution. We are thus required, for example, to solve the p.d.e.,

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = q(x,y) \tag{6.3.1}$$

in the rectangle $R=\{0\leq x\leq a,\ 0\leq y\leq b\}$

under the Dirichlet boundary conditions,

$$U(x,y) = f(x,y),\ (x,y)\in\partial R \tag{6.3.2}$$

where $\partial R$ is the boundary of R; $q(x,y)$ and $f(x,y)$ are known functions and $f(x,y)$ is assumed to be sufficiently smooth on the boundary $\partial R$.

In the rectangular region $R=(0,a)\times(0,b)$, let us define the mesh spacings $\Delta x=a/(n+1)$ and $\Delta y=b/(m+1)$ where n and m are integers. Hence, by using the mesh points $(x_i,y_j) = (i\Delta x,\ j\Delta y)$ we define the discrete interior region $R_h$ and the discrete boundary region $\partial R_h$ as follows:

$$R_h = \{(x_i,y_j) \mid 1\leq i\leq n,\ \ 1\leq j\leq m\},$$

$$\partial R_h = \partial R \cap \{(x_i,y_j) \mid 0\leq i\leq n+1,\ 0\leq j\leq m+1\}.$$

Further, we denote $U(x_i,y_j)$ as $U_{i,j}$ and apply the usual five point difference approximation to the Laplacian operator $\nabla^2$, i.e.

$$\nabla^2 U_{i,j} = \frac{1}{(\Delta x)^2}(U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) + \frac{1}{(\Delta y)^2}(U_{i,j-1} - 2U_{i,j} + U_{i,j+1}) + O(\Delta x^2 + \Delta y^2).$$

Hence, on omitting the truncation error terms, the finite difference analogue of the Poisson equation at the point $(x_i,y_j)$ is given, for $r=(\frac{\Delta x}{\Delta y})^2$, as

$$U_{i-1,j} + rU_{i,j-1} - 2(1+r)U_{i,j} + rU_{i,j+1} + U_{i+1,j} = (\Delta x)^2 q_{i,j},$$

$$1\leq i\leq n,\ 1\leq j\leq m. \tag{6.3.3}$$

A row-wise ordering of the mesh points in R and the application of the difference equations produces a discrete model for the differential

equation (6.3.1) which consists of the matrix equation,

$$M_D \underline{u} = \underline{d} \tag{6.3.4}$$

where

$$M_D = \begin{bmatrix} B & I & & & \\ I & B & I & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & & & I \\ & & & I & B \end{bmatrix}_{(m\times m)}, \quad B = \begin{bmatrix} -2(1+r) & r & & & \\ r & -2(1+r) & r & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & & & r \\ & & & r & -2(1+r) \end{bmatrix}_{(n\times n)} \tag{6.3.5}$$

and I is the (n×n) identity matrix.

The block vector $\underline{u}$ (which denotes the approximation of U) has components $\underline{u}_j$ which represent the solution along the $j^{th}$ horizontal line, i.e.,

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \quad \text{and} \quad \underline{u}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix}, \; j=1,2,\ldots,m.$$

The vector $\underline{d}$ has the same block structure as $\underline{u}$ and its components are given explicitly as,

$$d_{i,j} = (\Delta x)^2 q_{i,j} - r(\delta_{i,1} f_{0,j} + \delta_{i,n} f_{n+1,j}) - (\delta_{j,1} f_{i,0} + \delta_{j,m} f_{i,m+1}) \tag{6.3.6}$$

where $\delta_{i,j}$ is the Kronecker delta defined as,

$$\delta_{i,j} = 1 \quad i=j$$

and

$$\delta_{i,j} = 0 \quad i\neq j.$$

Block Rectangular Factorisation Method for the Solution of the Poisson Equation over a Rectangle

We consider the system of equations,

$$M_D[C,B,C]\underline{u} = \underline{d}$$

or

$$\begin{bmatrix} B & C & & & \\ C & B & C & & \\ & \ddots & \ddots & \ddots & \\ & & C & B & C \\ & & & C & B \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \vdots \\ \underline{u}_m \end{bmatrix} = \begin{bmatrix} \underline{d}_1 \\ \underline{d}_2 \\ \vdots \\ \underline{d}_m \end{bmatrix} \qquad (6.3.7)$$

where the matrices B and C are (n×n) matrices which are assumed to be commutative. For the Poisson equation, for example, $C=I_{(n\times n)}$, the identity matrix.

In recent years, considerable interst and attention has been given to the direct methods of solution of the model Dirichlet's matrix equation (6.3.7). The two basic techniques which have been most frequently applied include the Fourier transform method (Hockney (1965)) which relies on the a priori knowledge of trigonometric eigenvectors; and the cyclic reduction method (Buzbee et al (1970)) which relies on the simple block structure of the coefficient matrix.

Here, we introduce a block rectangular factorisation (BKREF) algorithm for the solution of the system (6.3.7) and show that this leads to a computationally fast method.

Following the method of the reversed triangular factorisation and expansion (ReTriFE) algorithm (4.4), the block matrix system (6.3.7) can be rewritten in the form,

$$\tilde{A}\underline{u} = \underline{g} \qquad (6.3.8)$$

or

$$\begin{bmatrix} I+\Phi^2 & -\Phi & & & \\ -\Phi & I+\Phi^2 & -\Phi & & \\ & \ddots & \ddots & \ddots & \\ & & -\Phi & I+\Phi^2 & -\Phi \\ & & & -\Phi & I+\Phi^2 \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \vdots \\ \underline{u}_m \end{bmatrix} = \begin{bmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \vdots \\ \underline{g}_m \end{bmatrix} \qquad (6.3.9)$$

where we have made the following transformations,

$$B^{-1}C = -(I+\Phi^2)^{-1}\Phi$$

or
$$\Phi = -(B+\sqrt{B^2-4C^2})^{-1}2C, \|B^2\| \geq 4\|C^2\| \qquad (6.3.10)$$

and
$$\underline{g}_j = B^{-1}(I+\Phi^2)\underline{d}_j \ , \ j=1,2,\ldots,m. \qquad (6.3.11)$$

It is easily verified by a simple multiplication that the matrix $\tilde{A}$ now has a unique factorisation,

$$\tilde{A} = \tilde{P}\tilde{P}^T \ ,$$

where $\tilde{P}$ is an $m\times(m+1)$ block rectangular matrix of the form,

$$P = \begin{bmatrix} I & -\Phi & & \\ & I & -\Phi & 0 \\ & & \ddots & \ddots \\ 0 & & & I \quad -\Phi \end{bmatrix}_{(m\times(m+1))} \ . \qquad (6.3.12)$$

Instead of the system (6.3.7), we thus consider the alternative form,

$$\tilde{P}\tilde{P}^T\underline{u} = \underline{g}$$

which, on the introduction of the auxiliary block vector $\underline{y}=(\underline{y}_1,\underline{y}_2,\ldots\underline{y}_{m+1})^T$, gives the two simpler systems,

$$\tilde{P}\underline{y} = \underline{g} \qquad (6.3.13a)$$

and
$$\tilde{P}^T\underline{u} = \underline{y} \qquad (6.3.13b)$$

which have to be solved in order to obtain $\underline{u}$.

It is easily deduced, from the expression for the matrix $\Phi$ given in (6.3.10), that for $\|B^2\| > 4\|C^2\|$,

$$||\Phi|| < 1 \ .$$

This allows the use of $\Phi$ as a matrix multiplier in an elimination process without the risk of growth in the rounding errors.

Hence, by following the approach of algorithm (4.4) (see equations (4.3.13)) and with $\Phi$ and $\underline{g}_j$ defined by (6.3.10) and (6.3.11) respectively, we obtain, on considering the solution of the system (6.3.13), the following algorithm:

$$\underline{u}_m = (I-\Phi^{2m+2})^{-1}[\hat{\underline{k}}-\Phi^{m+1}\hat{\underline{h}}] \qquad (6.3.14)$$

where $\hat{\underline{k}}$ and $\hat{\underline{h}}$ are vectors which are expressed as polynomial powers of $\Phi$

and given by,

$$\hat{\underline{k}} = \underline{g}_m + \Phi\underline{g}_{m-1} + \ldots \Phi^j \underline{g}_{m-j} + \ldots \Phi^{m-1}\underline{g}_1 \qquad (6.3.15)$$

and

$$\hat{\underline{h}} = \underline{g}_1 + \Phi\underline{g}_2 + \ldots \Phi^j \underline{g}_{j+1} + \ldots \Phi^{m-1}\underline{g}_m. \qquad (6.3.16)$$

Similarly, the intermediate solution sub-vectors $\underline{y}_j$ are given (see (4.3.14) for the equivalent point form), by

$$\left.\begin{aligned} \underline{y}_{m+1} &= -\Phi\underline{u}_m \\ \underline{y}_j &= \underline{g}_j + \Phi\underline{y}_{j+1}, \quad j=m,m-1,\ldots,1. \end{aligned}\right\} \qquad (6.3.17)$$

Finally, the solution sub-vectors $\underline{u}_j$ ($j=1,2,\ldots,m-1$) are given (see (4.3.15)), by,

$$\left.\begin{aligned} \underline{u}_1 &= \underline{y}_1 \\ \text{and} \quad \underline{u}_j &= \underline{y}_j + \Phi\underline{u}_{j-1}, \quad j=2,3,\ldots,m-1. \end{aligned}\right\} \qquad (6.3.18)$$

The BKREF algorithm (6.3.14)-(6.3.18) represents a direct method for the solution of the equation (6.3.7), i.e., the Poisson difference equation on a rectangle. We now show that for special cases, this approach is particularly efficient and fast and offers considerable economy over known methods. We shall also indicate methods by which a computationally simple and compact implementation of the given BKREF algorithm can be realised for the more general cases.

Special case 1 of the BKREF algorithm

For the case when B=2C, we derive immediately from (6.3.10) that $\Phi=-I$ (where I is the identity matrix) and hence the expression for $\underline{u}_m$ in (6.3.14) becomes indeterminate since the matrix $(I-\Phi^{2m+2})$ is singular in such a case. The following alternative scheme (cf equation (4.3.16(a)-(c)) is therefore applicable for the solution of the system $M_D[C,2C,C]$:

$$\underline{u}_m = [m\underline{g}_m - (m-1)\underline{g}_{m-1} + \ldots + (-1)^{m-1}\underline{g}_1]/(m+1) \qquad (6.3.19)$$

together with,

$$\left.\begin{aligned} \underline{y}_{m+1} &= \underline{u}_m \\ \underline{y}_j &= \underline{g}_j - \underline{y}_{j+1}, \quad j=m-1,m-2,\ldots,1, \end{aligned}\right\} \qquad (6.3.20)$$

and finally, the solution vector is obtained from,

$$\left.\begin{aligned} \underline{u}_1 &= \underline{y}_1 \\ \text{and} \quad \underline{u}_j &= \underline{y}_j - \underline{u}_{j-1}, \quad j=2,3,\ldots,m-1. \end{aligned}\right\} \qquad (6.3.21)$$

Special case 2 for the BKREF algorithm

Further, for the case when B=-2C, we obtain

$$\Phi = I$$

and hence the following scheme (cf equation (4.3.17(a)-(c))) applies in the solution of the system $M_D[C,-2C,C]$:

$$\underline{u}_m = [m\underline{g}_m+(m-1)\underline{g}_{m-1}+\ldots(m-j+1)\underline{g}_{m-j+1}+\ldots+\underline{g}_1]/(m+1) \qquad (6.3.22)$$

together with

$$\underline{y}_{m+1} = -\underline{u}_m \ ,$$

$$\underline{y}_j = \underline{g}_j+\underline{y}_{j+1} \ , \ j=m-1,m-2,\ldots 1 \qquad (6.3.23)$$

and finally, the solution sub-vector $\underline{u}_j$ is given by

$$\underline{u}_1 = \underline{y}_1 \ ,$$

$$\underline{u}_j = \underline{y}_j + \underline{u}_{j-1}, \quad j=2,3,\ldots,m-1. \qquad (6.3.24)$$

These two special cases considered above provide very fast methods for the solution of the block tridiagonal systems of the form A[C,2C,C] and A[C,-2C,C] respectively in O(2nm) arithmetic operations, where C can be either a diagonal, a tridiagonal or a full matrix.

Implementation of the general BKREF algorithm

It is assumed that B and C are (n×n) symmetric matrices and both commute, i.e., BC=CB. Hence by Frobenius theorem, there exists an (n×n) orthogonal matrix Q such that

$$B = Q\Lambda_B Q^T$$
$$C = Q\Lambda_C Q^T \qquad (6.3.25)$$

where Q is the set of eigenvectors of B and C; and $\Lambda_B=\text{diag}(\lambda_1,\lambda_2,\ldots\lambda_n)$ and $\Lambda_C=\text{diag}(\mu_1,\mu_2,\ldots,\mu_n)$ are the diagonal matrices of eigenvalues of B and C respectively. For the Poisson equation in a rectangle, the matrix B is an (n×n) constant element tridiagonal matrix (see (6.3.5)) of the form B[a,b,a] and C is a diagonal matrix; and hence the orthogonal matrix $Q=(q_{i,j})$ of eigenvectors of B is given by (see (5.4.18)),

$$q_{i,j} = \sqrt{\frac{2}{n+1}} \ \sin\left(\frac{ij\pi}{n+1}\right) \ , \qquad i,j=1,2,\ldots,n, \qquad (6.3.26)$$

and the associated eigenvalues are of the form,

$$\lambda_i = b + 2a\cos\left(\frac{i\pi}{n+1}\right) \ , \qquad i=1,2,\ldots,n. \qquad (6.3.27)$$

We now introduce the normal forms of B and C in order to obtain a computationally simpler form of the quantities in (6.3.10), (6.3.14), (6.3.15) and (6.3.16).

We consider first the expression in (6.3.10), i.e.,

$$\Phi = -(B+\sqrt{B^2-4C^2})^{-1}2C.$$

Since $\Phi$ is expressed as a function of B and C, it is easily shown (Lemma 5.2) that $\Phi$ has the same set of eigenvectors as B and C but different eigenvalues; and hence, $\Phi$ can be written as,

$$\Phi = Q\Lambda_\phi Q^T$$

where $\Lambda_\phi=\text{diag}(\gamma_1,\gamma_2,\ldots,\gamma_n)$ is the diagonal matrix of eigenvalues of $\Phi$ whose elements are given by,

$$\gamma_i = -(\lambda_i+\sqrt{\lambda_i^2-4\mu_i^2})^{-1}2\mu_i,\ i=1,2,\ldots,n; \qquad (6.3.28)$$

$\lambda_i,\mu_i$ being the eigenvalues of B and C respectively.

Further, from (6.3.11), we have

$$\underline{g}_j = B^{-1}(I+\Phi^2)\underline{d}_j \quad , j=1,2,\ldots,m$$

which can similarly be written as,

$$\underline{g}_j = Q\Lambda_g Q^T\underline{d}_j \quad , j=1,2,\ldots,m \qquad (6.3.29)$$

where $\Lambda_g$ is the diagonal matrix of eigenvalues of $B^{-1}(I+\Phi^2)$ which is given by,

$$\Lambda_g = \begin{bmatrix} (1+\gamma_1^2)/\lambda_1 & & & \\ & (1+\gamma_2^2)/\lambda_2 & & 0 \\ & & \ddots & \\ 0 & & & (1+\gamma_n^2)/\lambda_n \end{bmatrix}_{(n\times n)}$$

Determination of $\underline{U}_m$ (equation (6.3.14))

From (6.3.15) we have the expression,

$$\hat{\underline{k}} = \underline{g}_m+\Phi\underline{g}_{m-1}+\Phi^2\underline{g}_{m-2}+\ldots\Phi^j\underline{g}_{m-j}+\ldots+\Phi^{m-1}\underline{g}_1$$

which on substituting the normal form of $\Phi$ (i.e. $\Phi=Q\Lambda_Q Q$ where $QQ^T=I$) we have,

$$\hat{\underline{k}} = QQ^T\underline{g}_m+Q\Lambda_\phi Q^T\underline{g}_{m-1}+\ldots+Q\Lambda_\phi^j Q^T\underline{g}_{m-j}+\ldots+Q\Lambda^{m-1}Q^T\underline{g}_1.$$

We now introduce the transformation,

$$\tilde{g}_j = Q^T g_j \quad , \; j=1,2,\ldots,m \tag{6.3.30a}$$

which on using the formula for $g_j$ in (6.3.29) has the alternative form,

$$\tilde{g}_j = \Lambda_g Q^T d_j \; . \tag{6.3.30b}$$

Hence, we have,

$$\hat{k} = Q\hat{\hat{k}} \tag{6.3.31}$$

where the vector quantity $\hat{\hat{k}}$ can be written and evaluated in an efficient manner by using a nesting technique of the form,

$$\hat{\hat{k}} = [\Lambda_\phi(\ldots(((\Lambda_\phi(\Lambda_\phi(\Lambda_\phi\tilde{g}_1+\tilde{g}_2)+\tilde{g}_3)+g_4\ldots+\tilde{g}_{m-1})+\tilde{g}_m] \; . \tag{6.3.32}$$

1st loop

2nd loop

$(m-1)^{th}$ loop

Similarly, $\hat{h}$ (equation (6.3.16)) is expressed in the form,

$$\hat{h} = Q\hat{\hat{h}} \tag{6.3.33}$$

where $\hat{\hat{h}}$ is given by,

$$\hat{\hat{h}} = [\Lambda_\phi(\ldots(((\Lambda_\phi(\Lambda_\phi(\Lambda_\phi\tilde{g}_m+\tilde{g}_{m-1})+\tilde{g}_{m-2})+\ldots+\tilde{g}_2)+\tilde{g}_1] \; . \tag{6.3.34}$$

Hence, the expression for $u_m$ (equation (6.3.14)) i.e.,

$$u_m = (I-\phi^{2m+2})(\hat{k}-\phi^{m+1}\hat{h})$$

can be written, using the normal form for $\phi$, as

$$u_m = Q\Lambda_r Q^T[\hat{k}-Q\Lambda_s Q^T\hat{h}] \; . \tag{6.3.35}$$

On substituting equations (6.3.31) and (6.3.33) into (6.3.35) and using the orthogonality property of Q we have,

$$u_m = Q\Lambda_r(\hat{\hat{k}}-\Lambda_s\hat{\hat{h}}) \quad , \tag{6.3.36}$$

where $\Lambda_r, \Lambda_s$ are diagonal matrices whose elements are the eigenvalues of $(I-\phi^{2m+2})^{-1}$ and $\phi^{m+1}$ respectively and are given by,

$$\Lambda_r = \begin{bmatrix} 1/(1-\gamma_1^{2m+2}) & & & \\ & 1/(1-\gamma_2^{2m+2}) & & 0 \\ & & \ddots & \\ 0 & & & 1/(1-\gamma_n^{2m+2}) \end{bmatrix} \text{ and } \Lambda_s = \begin{bmatrix} \gamma_1^{m+1} & & & \\ & \gamma_2^{m+1} & & 0 \\ & & \ddots & \\ 0 & & & \gamma_n^{m+1} \end{bmatrix} \; .$$

The use of the form in (6.3.36) represents a more efficient method for the computation of $\underline{u}_m$ than the form given in (6.3.14).

Each loop of the nested multiplication in (6.3.32) or (6.3.34) involves a diagonal matrix-vector multiplication and a vector addition, which for a system of order n, requires n multiplications and n additions. Hence, we require a total of 2n(m-1) arithmetic operations to evaluate both $\hat{\underline{k}}$ and $\hat{\underline{h}}$ using the nesting technique. Thereafter, $\underline{u}_m$ is computed, using (6.3.36) in $(n^2+n)$ multiplications and $(n^2+2n)$ additions. Once $\underline{u}_m$ has been calculated in the most efficient manner, then the intermediate sub-vectors, $\underline{y}_j$ (j=m+1,m, ...1) and the solution sub-vectors $\underline{u}_j$, j=1,2,...,m-1, are easily computed by using equations (6.3.17) and (6.3.18) respectively.

Arithmetic Operation Count

Altogether, the arithmetic operation count required for the block rectangular factorisation (BKREF) algorithm is $O(2n^2m)$ as given in Table (6.1). This does not include the determination of the eigensystem of the submatrices B and C, and the evaluation of the matrix $\Phi$ which need to be computed only once for repeated solutions.

Arithmetic Operation Count for BKREF Algorithm

| Quantities in (Equation no.) | Nature of Computation (mat/vectors of order n) | Multiplications (×) | Additions (+) |
|---|---|---|---|
| 1. $\hat{\underline{k}}, \hat{\underline{h}}$ ((6.3.32), (6.3.34) | vector*vector+vector (nested 2(m-1) times) | 2n(m-1) | 2n(m-1) |
| 2. $\underline{u}_m$ (6.3.35) | mat×vector(vector+vector) | $n^2+n$ | $n^2+2n$ |
| 3. $\underline{y}_j$, j=1,...,m+1 (6.3.17) | vector+(mat×vector) ((m+1) times) | $n^2m+n^2$ | $n^2m+mn+n^2$ |
| 4. $\underline{u}_j$, j=1,...,m-1 (6.3.18) | vector+(matrix×vector) (m-1) times | $n^2m-n^2$ | $nm^2+mn-n^2$ |
| Total | BKREF | $2n^2m+2mn+n^2$ | $2n^2m+4nm$ |
| | Spectral Resolution Method (Algorithm 5.1) | $2n^2m+5nm$ | $2n^2m+4nm$ |

TABLE 6.1

Direct fast methods for the algorithmic solution of the one-dimensional tridiagonal matrix systems, using the factorisation and partitioning strategies, were presented in Chapter 4. The usefulness of those algorithms can now be extended to the solution of the block tridiagonal difference equation of the form ((6.3.7) by employing the spectral decomposition method (Algorithm 5.1).

Briefly we outline this here, using the notations of Algorithm (5.1) as follows:

1. Determine the matrix Q of eigenvectors of B and the matrices of eigenvalues $\Lambda_B=\text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$ and $\Lambda_C=\text{diag}(\mu_1,\mu_2,\ldots\mu_m)$ of B and C respectively.
2. Compute $\underline{\tilde{d}}_j=Q^T\underline{d}_j$, $j=1,2,\ldots,m$.
3. Solve the set of linear systems,

$$\Gamma_i[\mu_i,\lambda_i,\mu_i]\underline{\hat{u}}_i = \underline{\hat{d}}_i, \quad i=1,2,\ldots,n$$

where

$$\Gamma_i[\mu_i,\lambda_i,\mu_i]= \begin{bmatrix} \lambda_i & \mu_i & & & \\ \mu_i & \lambda_i & \mu_i & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \mu_i \\ 0 & & & \mu_i & \lambda_i \end{bmatrix}_{(m\times m)} \tag{6.3.36}$$

4. Finally, compute $\underline{u}_j=Q\underline{\bar{u}}_j$, $j=1,2,\ldots,m$.

The numerical solution of the set of simple linear systems in step (3) above can be achieved by the application of one of the following one-dimensional fast algorithms discussed in Chapter 4:-

(a) the ReTriFE Algorithm (4.3) which requires 5m multiplications, and 4m additions for each set of equations.

(b) the TRPP algorithm (4.8) with a requirement of less than 3m multiplications, 2m divisions and 3m additions per system.

(c) the CTRPP algorithm (4.9) which, with the coefficient matrix first scaled to the form $\Gamma[-1,\lambda_i,-1]$, requires just 2m multiplications m/2 divisions and 3m additions per system.

The arithmetic operation count for the spectral resolution method using, for example, the ReTriFE algorithm for step 3, is $O(2n^2m)$ as indicated in Table (6.1).

The Neumann's Boundary Condition

For the Poisson equation defined in (6.3.1)-(6.3.2), if instead of the Dirichlet's boundary condition (6.3.2), the normal derivatives $\frac{\partial U}{\partial n}$ is specified on the boundary, then we have the Neumann's boundary condition, i.e.,

$$\frac{\partial U}{\partial n} = f(x,y), \quad (x,y)\in\partial R. \tag{6.3.37}$$

By using the central difference approximations of the boundary conditions, i.e.,

$$\frac{\partial U}{\partial x} = \frac{U(x+\Delta x,y)-U(x-\Delta x,y)}{2\Delta x} + O(\Delta x)$$

and

$$\frac{\partial U}{\partial y} = \frac{U(x,y+\Delta y)-U(x,y-\Delta y)}{2\Delta y} + O(\Delta y)$$

then, the finite difference analogue of the Neumann Poisson equation becomes,

$$M_N\underline{u} = \underline{d} \tag{6.3.38}$$

where $M_N$ is of the form,

$$M_N = \begin{bmatrix} B & 2C & & & \\ C & B & C & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & & C \\ 0 & & & 2C & B \end{bmatrix}_{(m\times m)} \tag{6.3.39}$$

Here C represents the identity matrix $I_{(n+1\times n+1)}$ and B is given by,

$$B = \begin{bmatrix} -2(1+r) & 2r & & & \\ r & -2(1+r) & r & & \\ & \ddots & \ddots & \ddots & \\ & & r & -2(1+r) & r \\ & & & 2r & -2(1+r) \end{bmatrix}_{(n+1)\times(n+1)} \qquad (6.3.40)$$

Again, B and C commute but $M_N$ no longer has the same structure as that in (6.3.5).

The block rectangular factorisation (BKREF) algorithm cannot be modified easily to cope with the solution of the matrix equation (6.6.39) and hence we consider an alternative approach. The application of the spectral resolution algorithm in the solution of the Neumann's problem proceeds as that of the Dirichlet's problem, the only slight difference being the modification of the tridiagonal matrices $\Gamma_i$ in (6.3.36) to the form,

$$\Gamma_i = \begin{bmatrix} \lambda_i & 2\omega_i & & & \\ \omega_i & \lambda_i & \omega_i & & \\ & \ddots & \ddots & \ddots & \\ & & \omega_i & \lambda_i & \omega_i \\ & & & 2\omega_i & \lambda_i \end{bmatrix}_{(m\times m)}, \quad i=1,2,\ldots,n. \qquad (6.3.41)$$

A fast solution of the matrix system (6.3.41) can be obtained by the use of the TRPP algorithm (4.8); or, with an initial scaling of the coefficient matrix, by the CTRPP algorithm (4.9), which is considerably faster.

Experimental Results on the BKREF Solver for Poisson Equation

A Fortran program which implements the BKREF algorithm, using a single precision arithmetic is given in Appendix I as Program (16). This program was test run on the Loughborough University I.C.L. 1904S computer by considering the following special form of the Poisson equation, i.e., the Laplace equation,

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \qquad (6.3.42)$$

inside a rectangle $R=\{0\leq x\leq a,\ 0\leq y\leq b\}$, with the Dirichlet's boundary condition,

$$\left.\begin{aligned} U(x,0) &= f(x) = x(1-x), \qquad 0\leq x\leq a \\ U(x,b) &= 0 \qquad\quad 0\leq x\leq a \\ U(0,y) &= U(a,y) = 0, \qquad 0\leq y\leq b. \end{aligned}\right\} \qquad (6.3.43)$$

For a=b=1, the analytical solution of the above Laplace equation under the given boundary conditions simplifies to the form (Badak et al (1964), p.445),

$$U(x,y) = \sum_{k=1}^{\infty} \left\{ [\bar{f}_k \frac{\sinh k\pi(1-y)}{\sinh (k\pi)}] \sin k\pi x \right\} \qquad (6.3.44)$$

where

$$\left.\begin{aligned} \bar{f}_k &= 8/(k\pi)^3, \qquad k \text{ odd} \\ &= 0\ , \qquad\quad k \text{ even} \end{aligned}\right\} . \qquad (6.3.45)$$

Choosing a uniform mesh spacing $\Delta x=\Delta y=h=1/(n+1)$, where n is an integer, we obtain the computed solution of the given Laplace equation, using the BKREF algorithm and then compare the results with those derived from the known theoretical formula (6.3.44), as shown in Table (6.2).

The Solution of the Laplace Equation (6.3.42) using the BKREF Algorithm with Results Compared to a Known Analytical Solution, for $\Delta x=\Delta y=h=1/(n+1)$, n=20

| | | | | | | |
|---|---|---|---|---|---|---|
| 20h | c | 0.001891 | 0.003124 | 0.003281 | 0.002289 | 0.000502 |
| | a | 0.001890 | 0.003122 | 0.003270 | 0.002282 | 0.000500 |
| 16h | c | 0.010315 | 0.017045 | 0.017860 | 0.012455 | 0.002733 |
| | a | 0.010316 | 0.017048 | 0.017854 | 0.012456 | 0.002730 |
| 12h | c | 0.022550 | 0.037260 | 0.039023 | 0.039023 | 0.027228 |
| | a | 0.022549 | 0.037261 | 0.039024 | 0.039024 | 0.027226 |
| 8h | c | 0.430100 | 0.071223 | 0.074593 | 0.052039 | 0.022552 |
| | a | 0.430099 | 0.071220 | 0.074591 | 0.052039 | 0.022551 |
| 4h | c | 0.079546 | 0.131455 | 0.137672 | 0.096050 | 0.021045 |
| | a | 0.079548 | 0.131452 | 0.137674 | 0.096050 | 0.021046 |
| $y_j$ \ $x_i$ | | 4h | 8h | 12h | 16h | 20h |

TABLE 6.2

*c=computed solution using BKREF*
*a=analytical solution using formula (6.3.44)*

## 6.4 FAST A.D.I. METHODS FOR THE SOLUTION OF LINEAR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS WITH DIRICHLET'S AND NEUMANN'S BOUNDARY CONDITIONS

In section (5.7) of Chapter 5, we considered the solution of the two space linear periodic parabolic problem by A.D.I. methods. A further discussion of an identical problem, with the more usual Dirichlet's and Neumann's boundary conditions is considered here, in which the use of the tridiagonal recursive point partitioning (TRPP) algorithmic idea applied to the Peaceman and Rachford (1955) A.D.I. methods is shown to offer some substantial computational gains over some other known methods.

We consider the model two-space heat conduction equation,

$$\frac{\partial U}{\partial t} = \tau\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right), \ \tau = \text{constant} \tag{6.4.1}$$

in the region $R' = (x,y,t) \epsilon R \times (t>0)$, (6.4.2)

(where $R = \{0 \leq x,y \leq 1\}$ is a closed, connected square region in the x-y plane) under the Dirichlet boundary condition,

$$U(x,y,t) = g(x,y,t), \ (x,y,t) \epsilon \partial R \tag{6.4.3}$$

and the initial boundary condition,

$$U(x,y,0) = f(x,y), \ 0 \leq x,y \leq 1 \tag{6.4.4}$$

where $g(x,y,t)$, $f(x,y)$ are known functions and $\partial R$ is the boundary of $R'$.

By imposing a uniform mesh of size $h = \Delta x = \Delta y = 1/(n+1)$, and $\ell = \Delta t > 0$ on $R'$, the A.D.I. finite difference analogue of (6.4.1) at each internal node $(x_i, y_j) \epsilon R$ is given by the two alternate equations, (6.4.5) and (6.4.6) where we have omitted the $O(h^2+\ell)$ truncation error terms and set $r = \ell\tau/h^2$, i.e.,

$$-rU_{i-1,j,2k+1} + (1+2r)U_{i,j,2k+1} - rU_{i+1,j,2k+1}$$

$$= rU_{i,j-1,2k} + (1-2r)U_{i,j,2k} + rU_{i,j+1,2k}, \ 1 \leq i,j \leq n, \ k>0 \tag{6.4.5}$$

and

$$-rU_{i,j-1,2k+2} + (1+2r)U_{i,j,2k+2} - rU_{i,j+1,2k+2}$$

$$= rU_{i-1,j,2k+1} + (1-2r)U_{i,j,2k+1} + rU_{i+1,j,2k+1}, \ 1 \leq i,j \leq n. \tag{6.4.6}$$

It has been established (Peaceman and Rachford (1955)) that a combination

of these two finite difference procedures is stable provided r(and hence $\ell$) remain the same for two adjacent time steps.

Equations (6.4.5) and (6.4.6) together with the boundary and initial conditions (6.4.3) and (6.4.4) yield the two sets of compound matrix equations, which in their normalised form can be expressed as,

$$G\underline{u}_{2k+1} = H\underline{u}_{2k} + \underline{b} \tag{6.4.7}$$

$$G\underline{\tilde{u}}_{2k+2} = H\underline{\tilde{u}}_{2k+1} + \underline{d} \tag{6.4.8}$$

where G and H are the compound matrices of order $(n\times n)$, given by,

$$G = \begin{bmatrix} A & & & \\ & A & & 0 \\ & & \ddots & \\ & 0 & & A \end{bmatrix} \qquad H = \begin{bmatrix} \gamma I & I & & & \\ I & \gamma I & I & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & \ddots & I \\ & 0 & & I & \gamma I \end{bmatrix} \tag{6.4.9}$$

where A is an $(n\times n)$ matrix

$$A = \begin{bmatrix} \delta & -1 & & & \\ -1 & \delta & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & \ddots & -1 \\ & 0 & & -1 & \delta \end{bmatrix} \tag{6.4.10}$$

and $\delta = \dfrac{1+2r}{r}$, $\quad \gamma = \dfrac{1-2r}{r}$. (6.4.11)

The vector $\underline{b}$ of order n is known and derived from the initial and boundary conditions, ordered row-wise and given explicitly as,

$$\underline{b} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \\ \vdots \\ \underline{b}_n \end{bmatrix} \quad \text{where} \quad \underline{b}_j = \begin{bmatrix} u_{j,j,2k+1} \\ 0 \\ \vdots \\ 0 \\ u_{n+1,j,2k+1} \end{bmatrix}, \quad j=2,3,\ldots,n-1,$$

and

$$\underline{b}_1 = \begin{bmatrix} u_{1,0,2k} + u_{0,1,2k+1} \\ u_{2,0,2k} \\ \vdots \\ u_{n-1,0,2k} \\ u_{n,0,2k} + u_{n+1,1,2k+1} \end{bmatrix} \text{ and } \underline{b}_n = \begin{bmatrix} u_{1,n+1,2k} + u_{0,n,2k+1} \\ u_{2,n+1,2k} \\ \vdots \\ u_{n-1,n+1,2k} \\ u_{n,n+1,2k} + u_{n+1,n,2k+1} \end{bmatrix}$$

Similarly, the vector $\underline{d}$ ordered column-wise, is given explicitly as,

$$\begin{bmatrix} \underline{d}_1 \\ \underline{d}_2 \\ \vdots \\ \underline{d}_n \end{bmatrix} \quad \text{where } \underline{d}_i = \begin{bmatrix} u_{i,0,2k+1} \\ 0 \\ \vdots \\ 0 \\ u_{i,n+1,2k+1} \end{bmatrix} \quad i=2,\ldots,n-1,$$

$$\underline{d}_1 = \begin{bmatrix} u_{0,1,2k+1} + u_{1,0,2k+1} \\ u_{0,2,2k+1} \\ \vdots \\ u_{0,n-1,2k+1} \\ u_{0,n,2k+1} + u_{1,n+1,2k+1} \end{bmatrix} \text{ and } \underline{d}_n = \begin{bmatrix} u_{n,1,2k+1} + u_{n,0,2k+1} \\ u_{n,2,2k+1} \\ \vdots \\ u_{n,n-1,2k+1} \\ u_{n,n,2k+1} + u_{n,n+1,2k+1} \end{bmatrix}$$

The unknown compound vectors $\underline{u}_p$ and $\tilde{\underline{u}}_p$ (which are approximations of $\underline{U}_p$ and $\tilde{\underline{U}}_p$ respectively for $p=2k,2k+1,2k+2,\ldots$) are ordered row-wise and column-wise respectively and can be written in the form,

$$\underline{u}_p^T = [u_{1,1,p}, u_{2,1,p}, \ldots u_{n,1,p}, u_{1,2,p}, u_{2,2,p}, \ldots u_{n,2,p}, \ldots u_{1,n,p}, u_{2,n,p} \ldots \ldots u_{n,n,p}]$$

and

$$\tilde{\underline{u}}_p^T = [u_{1,1,p}, u_{1,2,p}, \ldots u_{1,n,p}, u_{2,1,p}, u_{2,2,p}, \ldots u_{2,n,p}, \ldots u_{n,1,p}, u_{n,2,p}, \ldots \ldots u_{n,n,p}].$$

The use of the Recursive Point Partitioning Algorithm in the A.D.I. Method

From equations (6.4.7) and (6.4.8) it is easily seen that the major computational task in the A.D.I. procedure is the line-by-line solution, along the rows and then columns, of sets of n tridiagonal systems of equation of the form,

$$A\underline{\phi} = \underline{g} \tag{6.4.12}$$

where A is given by (6.4.10), $\underline{\phi}$ represents the solution along a row or column and $\underline{g}$ is the appropriate right hand side vector. Hence, substantial computational effort can be saved if a very fast algorithmic method is employed in the repeated solution of the tridiagonal matrix systems derived from the A.D.I. techniques.

The CTRPP algorithm (4.9) for the solution of the simple normalised tridiagonal matrix system of the form (6.4.10) was given in Chapter 4 as a variant of the more general TRPP algorithm (4.8). It is easily established, by using equation (6.4.11), that for $r>0$, $|\delta|>2$ and hence the matrix A in (6.4.10) is diagonally dominant. The stability of the numerical solution of sets of tridiagonal matrix systems in (6.4.7) and (6.4.8) by the CTRPP algorithm (4.9) is therefore immediately guaranteed under the diagonal dominance condition.

The actual amount of arithmetic operation needed to solve each tridiagonal matrix system (6.4.12) using the CTRPP solver along a line is 2n multiplications and 2n additions for repeated applications, assuming that the mesh spaces, and hence the matrix A, remain unchanged. Compared to the more usual method of solution by the Gaussian elimination method (Varga (1962)) which requires 5n multiplications and 4n additions per system, the CTRPP algorithm, when applied in the A.D.I. procedure along the rows and columns of the network, offers substantial computational gains.

Neumann's Boundary Condition

For a Neumann's boundary condition of the form,

$$\frac{\partial U}{\partial n} = 0, \quad (x,y,t)\in\partial R,$$

where n denotes the direction of the outward normal, the use of a one sided difference approximation on a standard grid to approximate the boundary conditions, e.g.

$$\frac{\partial U}{\partial x} = \frac{(U_{i+1}-U_i)}{h} + O(h), \quad i=0,n+1, \text{ etc.}$$

leads to linear systems of the form (6.4.7) and (6.4.8) but now A is given as an $(n+2)\times(n+2)$ matrix of the form,

$$A = \begin{bmatrix} \delta-1 & -1 & & & \\ -1 & \delta & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \delta & -1 \\ 0 & & & -1 & \delta-1 \end{bmatrix}_{(n+2)\times(n+2)} \qquad (6.4.13)$$

A fast solution of (6.4.7) and (6.4.8) where A is of the form (6.4.13) can now be obtained by a suitably modified form of the CTRPP algorithm provided A is non-singular, i.e., $|\delta|>2$.

## 6.5 FAST NUMERICAL SOLUTION OF THE BIHARMONIC EQUATION WITH MIXED BOUNDARY CONDITIONS IN A RECTANGULAR REGION

We define the system of linear algebraic equations

$$M[F_i,A_i,B_i,C_i,E_i]\underline{u} = \underline{d} \tag{6.5.1}$$

to be of block five diagonal form if the matrix M is given by

$$M = \begin{bmatrix} B_1 & C_1 & E_1 & & & & & & \\ A_2 & B_2 & C_2 & E_2 & & & 0 & & \\ F_3 & A_3 & B_3 & C_3 & E_3 & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & & & & & E_{n-2} \\ & 0 & & & F_{n-1} & A_{n-1} & B_{n-1} & C_{n-1} & \\ & & & & & F_n & A_n & B_n & \end{bmatrix}_{(n\times n)} \tag{6.5.2}$$

where each of the submatrices $F_i,A_i,B_i,C_i,E_i$ is square of order m. Linear equations of the form (6.5.1) together with (6.5.2) are obtained, for example, from finite difference approximation of boundary-value problems for fourth-order elliptic p.d.e.'s.

### Statement of the Problem and the Formation of the Finite Difference Equation

We consider the solution of the two-dimensional Biharmonic elliptic equation,

$$\nabla^4 U(x,y) \equiv \frac{\partial^4 U}{\partial x^4} + 2\frac{\partial^4 U}{\partial x^2 \partial y^2} + \frac{\partial^4 U}{\partial y^4} = \psi(x,y) \ . \tag{6.5.3}$$

in a rectangluar region $R=\{0\le x\le \ell_1,\ 0\le y\le \ell_2\}$ bounded by $\partial R$. This problem, together with appropriate boundary conditions, describes, for example, the small deflections in a thin homogeneous plate under various forces exerted on the plate boundaries.

We consider first the following set of boundary conditions in $\partial R$:

$$\left.\begin{aligned} U(x,y) &= a(x), \quad \frac{\partial^2 U}{\partial y^2}(x,y) = b(x), \quad y = \ell_2, \quad 0\le x\le \ell_1 \\ U(x,y) &= c(x), \quad \frac{\partial^2 U}{\partial y^2}(x,y) = d(x), \quad y = 0, \quad 0\le x\le \ell_1 \\ U(x,y) &= e(y), \quad \frac{\partial^2 U}{\partial x^2}(x,y) = f(y), \quad x = 0, \quad 0\le y\le \ell_2 \\ \text{and} \quad U(x,y) &= g(y), \quad \frac{\partial^2 U}{\partial x^2}(x,y) = h(y), \quad x = \ell_1, \quad 0\le y\le \ell_2 . \end{aligned}\right\} \tag{6.5.4}$$

The functions b(x), d(x), f(y), h(y) represent the bending moments distributed along the two opposite edges.

In the rectangular region R, we define a set of mesh points $R_h$, i.e.,

$$x_i = i\Delta x , \qquad i=1,2,\ldots,n,$$

$$y_i = i\Delta y , \qquad j=1,2,\ldots m,$$

and the discrete boundary points $\partial R_h$, i.e.,

$$x_i = i\Delta x, \qquad i=0,n+1$$

and

$$y_i = j\Delta y, \qquad j=0,m+1$$

where $n\Delta x=\ell_1$, $m\Delta y=\ell_2$ and m,n are integers.

Further, we define the following quantities,

$$\theta_x = \frac{\Delta y^2}{2(\Delta x^2+\Delta y^2)} , \quad \theta_y = \frac{\Delta x^2}{2(\Delta x^2+\Delta y^2)}, \quad \tau = 1+2\theta_x^2+2\theta_y^2 , \tag{6.5.5}$$

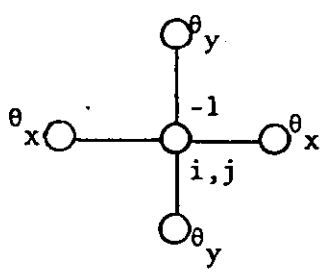
and denote $a(x_i)$ as $a_i$, $e(y_j)$ as $e_j$ etc.

By Taylor's series expansion, the haromínic operator $\nabla^2 U_{i,j}$ is given, on neglecting the truncation error term $O(\Delta x^2+\Delta y^2)$ as,

$$\nabla^2 U_{i,j} = \theta_x U_{i-1,j}+\theta_y U_{i,j-1}-U_{i,j}+\theta_y U_{i,j+1}+\theta_x U_{i+1,j} ,$$

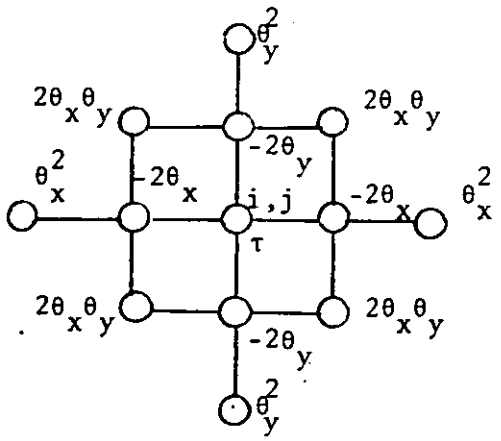
which is represented by the computational molecule given in Figure 6.1(a), from which the 13 point computational molecule for the Biharmonic operator $\nabla^4 U_{i,j}$ is easily obtained as given in Figure 6.1(b).

Computational Molecule for the Harmonic and Biharmonic Operators




$\nabla^2 U_{i,j}$

(a)




$\nabla^4 U_{i,j}$

(b)

FIGURE 6.1

Hence, a finite difference equation for the problem (6.5.3) is given, on neglecting $O((\Delta x)^2+(\Delta y)^2)$ terms, by

$$\tau U_{i,j}-2\theta_x(U_{i-1,j}+U_{i+1,j})-2\theta_y(U_{i,j-1}+U_{i,j+1})+\theta_x^2(U_{i-2,j}+U_{i+2,j})$$

$$+\theta_y^2(U_{i,j-2}+U_{i,j+2})+2\theta_x\theta_y(U_{i-1,j-1}+U_{i-1,j+1}+U_{i+1,j-1}+U_{i+1,j+1})$$

$$=\Psi_{i,j} \quad 1\leq i\leq n, \quad 1\leq j\leq m, \tag{6.5.6}$$

where $U_{i,j}$ and $\Psi_{i,j}$ denote the values $U((i\Delta x, j\Delta y)$ and $\Psi(i\Delta x, j\Delta y)$ respectively.

By choosing a column-wise ordering of the mesh points and applying (6.5.6) we obtain the discrete representation of (6.5.3) together with the boundary condition (6.5.4) as

$$M_D\underline{u} = \underline{b}_D + \underline{\Psi} \ (=\underline{d}_D) \tag{6.5.7a}$$

where $\underline{u}$ is the approximation of $\underline{U}$ ordered to conform with $M_D$, and $M_D$ is the $(n\times n)$ block five diagonal matrix,

$$M_D = \begin{bmatrix} T^2+\theta_x^2 I & -2\theta_x T & \theta_x^2 I & & & & \\ -2\theta_x T & T^2+2\theta_x^2 I & -2\theta_x T & \theta_x^2 I & & 0 & \\ \theta_x^2 I & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \theta_x^2 I \\ & 0 & & \theta_x^2 I & -2\theta_x T & T^2+2\theta_x^2 I & -2\theta_x T \\ & & & & \theta_x^2 I & -2\theta_x T & T^2+\theta_x^2 I \end{bmatrix}_{(n\times n)} \tag{6.5.7b}$$

where T is given by the $(m\times m)$ tridiagonal matrix,

$$T = \begin{bmatrix} 1 & -\theta y & & & \\ \theta y & 1 & -\theta y & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & \ddots & -\theta y \\ & 0 & & -\theta y & 1 \end{bmatrix}_{(m\times m)}. \tag{6.5.8}$$

By using the central difference schemes,

$$\frac{\partial^2 U}{\partial x^2}(x,y) = \frac{(U_{i-1,j} - 2U_{i,j} + U_{i+1,j})}{(\Delta x)^2} + O(\Delta x^2)$$

and similarly for $\frac{\partial^2 U}{\partial y^2}$,

in the approximation of the second derivative of the boundary conditions, the vector $\underline{b}_D$ derived from the boundary conditions is given explicitly in the form,

$$\underline{b}^T = [\underline{b}_1, \underline{b}_2, \ldots, \underline{b}_n]$$

where,

$$\underline{b}_1 = \begin{bmatrix} -(\theta_x^2\Delta x^2 f_1 + 2\theta_x^2 e_1 + 2\theta_x\theta_y e_2 - 2\theta_x e_1 + 2\theta_x\theta_y e_0 - 2\theta_y c_1 + 2\theta_x\theta_y c_2 + \Delta y^2\theta_y^2 d_1 + 2\theta_y^2 c_1) \\ -(\theta_x^2\Delta x^2 f_2 + 2\theta_x^2 e_2 + 2\theta_x\theta_y e_3 - 2\theta_x e_2 + 2\theta_x\theta_y e_1 + \theta_y^2 c_1) \\ -(\theta_x^2\Delta x^2 f_3 + 2\theta_x^2 e_3 + 2\theta_x\theta_y e_4 - 2\theta_x e_3 + 2\theta_x\theta_y e_2) \\ \vdots \\ -(\theta_x^2\Delta x^2 f_j + 2\theta_x^2 e_j + 2\theta_x\theta_y e_{j+1} - 2\theta_x e_j + 2\theta_x\theta_y e_{j-1}) \\ \vdots \\ -(\theta_x^2\Delta x^2 f_{m-1} + 2\theta_x^2 e_{m-1} + 2\theta_x\theta_y e_m - 2\theta_x e_{m-1} + 2\theta_x\theta_y e_{m-2} + \theta_y^2 a_1) \\ -(\theta_x^2\Delta x^2 f_m + 2\theta_x^2 e_m + 2\theta_x\theta_y e_{m+1} - 2\theta_x e_m + 2\theta_x\theta_y e_{m-1} - 2\theta_y a_1 + 2\theta_x\theta_y a_2 + \theta_y^2\Delta y^2 b_1 + 2\theta_y^2 a_1) \end{bmatrix}, \quad (6.5.9a)$$

$$\underline{b}_2 = \begin{bmatrix} -(\theta_x^2 e_1 + 2\theta_x\theta_y c_1 - 2\theta_y c_2 + 2\theta_x\theta_y c_3 + \theta_y^2\Delta y^2 d_2 + 2\theta_y^2 c_2) \\ -(\theta_x^2 e_2 + \theta_y^2 c_2) \\ -\theta_x^2 e_3 \\ \vdots \\ -\theta_x^2 e_{m-2} \\ -(\theta_x^2 e_{m-1} + \theta_y^2 a_2) \\ -(\theta_x^2 e_m + 2\theta_x\theta_y a_1 - 2\theta_y a_2 + 2\theta_x\theta_y a_3 + \theta_y^2\Delta y^2 b_2 + 2\theta_y^2 a_2) \end{bmatrix}, \quad (6.5.9b)$$

$$\underline{b}_i = \begin{bmatrix} -(2\theta_x\theta_y c_{i-1} - 2\theta_y c_i + 2\theta_x\theta_y c_{i+1} + \theta_y^2\Delta y^2 d_i + 2\theta_y^2 c_i) \\ -\theta_y^2 c_i \\ 0 \\ \vdots \\ 0 \\ -\theta_y^2 a_i \\ -(2\theta_x\theta_y a_{i-1} - 2\theta_y a_i + 2\theta_x\theta_y a_{i-1} + \theta_y^2\Delta y^2 b_i + 2\theta_y^2 a_i \end{bmatrix}, \quad i=3,\ldots,n-2, \tag{6.5.9c}$$

$$\underline{b}_{n-1} = \begin{bmatrix} -(2\theta_x\theta_y c_{n-2} - 2\theta_y c_{n-1} + 2\theta_x\theta_y c_n + \theta_x^2 g_1 + \theta_y^2\Delta y^2 d_{n-1} + 2\theta_y^2 c_{n-1} \\ -\theta_y^2 c_{n-1} + \theta_x^2 g_2 \\ -\theta_x^2 g_3 \\ \vdots \\ -\theta_x^2 g_{m-2} \\ -(\theta_x^2 g_{m-1} + \theta_y^2 a_{n-1}) \\ -(2\theta_x\theta_y a_{n-2} - 2\theta_y a_{n-1} + 2\theta_x\theta_y a_n + \theta_x^2 g_m + \theta_y^2\Delta y^2 b_{n-1} + 2\theta_y^2 a_{n-1}) \end{bmatrix} \tag{6.5.9d}$$

and

$$\underline{b}_n = \begin{bmatrix} -(\theta_y^2\Delta y^2 d_n + 2\theta_y^2 c_n + 2\theta_x\theta_y c_{n-1} - 2\theta_y c_n + 2\theta_x\theta_y c_{n+1} - 2\theta_x g_1 + 2\theta_x\theta_y g_2 + \theta_x^2\Delta x^2 h_1 + 2\theta_x^2 g_1) \\ -(\theta_y^2 c_n + 2\theta_x\theta_y g_1 - 2\theta_x g_2 + 2\theta_x\theta_y g_3 + \theta_x^2\Delta x^2 h_2 + 2\theta_x^2 g_2) \\ -(2\theta_x\theta_y g_2 - 2\theta_x g_3 + 2\theta_x\theta_y g_4 + \theta_x^2\Delta x^2 h_3 + 2\theta_x^2 g_3) \\ \vdots \\ -(2\theta_x\theta_y g_{j-1} - 2\theta_x g_j + 2\theta_x\theta_y g_{j+1} + \theta_x^2\Delta x^2 h_j + 2\theta_x^2 g_j) \\ \vdots \\ -(2\theta_x\theta_y g_{m-2} - 2\theta_x g_{m-1} + 2\theta_x\theta_y g_m + \theta_x^2\Delta x^2 h_{m-1} + 2\theta_x^2 g_{m-1} + \theta_y^2 a_n) \\ -(\theta_y^2\Delta y^2 b_n + 2\theta_y^2 a_n + 2\theta_x\theta_y g_{m-1} - 2\theta_x g_m + 2\theta_x\theta_y g_{m+1} - 2\theta_y a_n + 2\theta_x\theta_y a_{n-1} + \theta_x^2\Delta x^2 h_m + 2\theta_x^2 g_m) \end{bmatrix} \tag{6.5.9e}$$

The matrix (6.5.7b) is equivalent to that in (6.5.2) if

$$\left.\begin{aligned} B_1 &= B_n = T^2 + \theta_x^2 I, \\ B_i &= B = T^2 + 2\theta_x^2 I, \quad i=2,3,\ldots,n-1, \\ A_{i+1} &= C_i = C = -2\theta_x T, \quad i=1,2,\ldots,n-1, \\ \text{and} \quad F_{i+2} &= E_i = E = \theta_x^2 I, \quad i=3,4,\ldots,n-2. \end{aligned}\right\} \tag{6.5.10}$$

Hence, the matrix (6.5.7b) can be rewritten as,

$$M_D = \begin{bmatrix} B_1 & C & E & & & & \\ C & B & C & E & & 0 & \\ E & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & E & \\ & 0 & & E & C & B & C \\ & & & & E & C & B_n \end{bmatrix}_{(n\times n)} \tag{6.5.11}$$

where each submatrix is of order m.

It is easy to establish, by a simple matrix multiplication, that the matrices $B_1, B_n, B, C$ and $E$ defined in (6.5.10) commute with the matrix $T$ and hence possess a common orthogonal matrix of eigenvectors

$$Q = (q_{i,j}) = \sqrt{2/(m+1)}\,\sin\left(\frac{ij\pi}{m+1}\right), \quad i,j=1,2,\ldots,m \tag{6.5.12}$$

such that these matrices can be written in the normal form,

$$\left.\begin{aligned} B_1 &= Q\Delta' Q^T \\ B_n &= Q\Delta'' Q^T \\ B &= Q\Lambda Q^T \\ C &= Q\Lambda Q^T \\ E &= Q\Omega Q^T \end{aligned}\right\} \tag{6.5.13}$$

where $\Delta', \Delta'', \Delta, \Lambda$, and $\Omega$ are real diagonal matrices whose elements are the eigenvalues of $B_1, B_n, B, C$ and $E$ respectively. These diagonal matrices can be written as,

$$\left.\begin{aligned} \Delta' &= \mathrm{diag}(\delta_1', \delta_2', \ldots, \delta_m'), & \delta_j' &= \theta_x^2 + \mu_j^2, \\ \Delta'' &= \mathrm{diag}(\delta_1'', \delta_2'', \ldots, \delta_m''), & \delta_j'' &= \theta_x^2 + \mu_j^2, \\ \Delta &= \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_m), & \delta_j &= 2\theta_x^2 + \mu_j^2, \\ \Lambda &= \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m), & \lambda_j &= -2\theta_x\mu_j, \\ \Omega &= \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_m), & \omega_j &= \theta_x^2 \end{aligned}\right\} \quad j=1,2,\ldots,m, \tag{6.5.14}$$

where $\mu_j$ are the eigenvalues of the tridiagonal matrix $T$, and are given by

$$\mu_j = 1+2\theta_y\cos(j\pi/(m+1)), \quad j=1,2,\ldots,m. \tag{6.5.15}$$

To solve the quindiagonal system $M_D\underline{u}=\underline{d}$ in (6.5.7a) with $M_D$ given by (6.5.11), we use the method of matrix decomposition in what amounts to an extension of the equivalent algorithm for block tridiagonal system (Algorithm 5.1).

We think of the block vectors $\underline{u}$ and $\underline{d}$ as two dimensional arrays $\underline{u}=(u_{i,j})$ and $\underline{d}=(d_{i,j})$ partitioned to conform with $M_D$, such that $\underline{u}_i$ denotes the $i^{th}$ row of $\underline{u}$ etc.

Then, the solution of $M_D\underline{u}=\underline{d}$ is obtained as follows:

Step 1: Compute or determine the common matrix of eigenvectors Q (equation (6.5.12) and the eigenvalues of $B_1, B, B_n, C$ and E (equation 6.5.14).

Step 2: Next, we compute the vectors,

$$\bar{\underline{d}}_i = Q^T\underline{d}_i \ , \quad i=1,2,\ldots,n$$

where $\bar{\underline{d}}_i$ is a column-wise vector, i.e., each column of $\underline{d}$ is multiplied by $Q^T$.

Step 3: Further, we re-order the array $\bar{\underline{d}}$ by horizontal lines instead of vertical lines to generate the array $\hat{\underline{d}}$ and then solve the set of quindiagonal systems,

$$\Gamma_j\hat{\underline{u}}_j = \hat{\underline{d}}_j \ , \quad j=1,2,\ldots,m. \tag{6.5.16a}$$

where

$$\Gamma_j = \begin{bmatrix} \delta'_j & \lambda_j & \omega_j & & & & \\ \lambda_j & \delta_j & \lambda_j & \omega_j & & 0 & \\ \omega_j & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \omega_j & \\ & & \ddots & \ddots & \ddots & \delta_j & \lambda_j \\ 0 & & & \omega_j & \lambda_j & \delta''_j & \end{bmatrix}_{(n\times n)} \tag{6.5.16b}$$

Step 4: Finally, we re-order the array $\hat{\underline{u}}$ by vertical lines instead of horizontal lines to generate $\bar{\underline{u}}_i$ and then compute the solution vectors,

$$\underline{u}_i = Q\bar{\underline{u}}_i, \quad i=1,2,\ldots,n.$$

Fast Solution of the Quindiagonal Systems

The set of quindiagonal systems (6.5.16) is to be solved many times and hence there is the necessity to apply a fast algorithmic method in the solution of each system. Such a fast method would normally be obtained by an algorithmic process which is mathematically equivalent to Gaussian elimination method without pivoting. The omission of pivoting without the risk of instability in the solution process is normally justified by the diagonal dominance property of the coefficient matrix.

It is easy to show by using the expressions for $u_j$ in (6.5.15), and $\theta_x, \theta_y$ in (6.5.5) that the quindiagonal matrix $\Gamma_j$ in (6.5.16b) is diagonally dominant, i.e.,

$$\left.\begin{aligned} |\delta_j| &\geq 2|\lambda_j| + 2|\omega_j| \\ |\delta'_j| &\geq |\lambda_j| + |\omega_j| \\ |\delta''_j| &\geq |\lambda_j| + |\omega_j| \end{aligned}\right\} \qquad j=1,2,\ldots,m.$$

Hence, there is no instability risk in solving the quindiagonal system (6.5.16) by an algorithmic method which does not include a pivoting strategy. This enables us to apply a slightly modified form of the constant quindiagonal recursive point partitioning method (CQRPP Algorithm (4.10)) in order to obtain a very fast stable solution of the set of constant element quindiagonal matrix systems (6.5.16) in less than 11n/2 multiplications, n divisions and 11n/2 additions per system.

A Factorisation Method and the Application of BKREF Algorithm

The block-five coefficient matrix $M_D$ in (6.5.76) can be expressed as the square of a block tridiagonal matrix, i.e.,

$$M_D = A_F^2 \tag{6.5.17a}$$

where

$$A_F = \begin{bmatrix} T & -\theta_x I & & & \\ -\theta_x I & T & -\theta_x I & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\theta_x I \\ 0 & & & -\theta_x I & T \end{bmatrix} . \qquad (6.5.17b)$$

Hence equation (6.5.7a) can be represented in the form,

$$M_D \underline{u} = A_F^2 \underline{u} = \underline{d}_D \quad , \qquad (6.5.18)$$

which, on introducing the auxiliary vector $\underline{v}$, gives the two alternative systems,

$$\left.\begin{aligned} & A_F \underline{v} = \underline{d}_D \ , \\ \text{and} \quad & A_F \underline{u} = \underline{v} \ . \end{aligned}\right\} \qquad (6.5.19)$$

Each of these two systems can now be solved by using the Block Rectangular Factorisation (BKREF) algorithm (section 6.3) or the corresponding spectral resolution method of Buzbee et al (1970). The above method is equivalent to the splitting of the Biharmonic equation, $\nabla^4 \underline{u} = \underline{\Psi}$ into two coupled Poisson equations,

$$\left.\begin{aligned} & \nabla^2 \underline{v} = \underline{\Psi} \\ \text{and} \quad & \nabla^2 \underline{u} = \underline{v} \ . \end{aligned}\right\} \qquad (6.5.20)$$

It is, however, not always possible to apply the above factorisation method in the solution of the Biharmonic equation with certain boundary conditions such as the one to be considered below.

Mixed Boundary Conditions

We consider further, the solution of the Biharmonic equation (6.5.3) with the following boundary conditions:

$$\left.\begin{aligned} & U(x,y) = a(x), \ \frac{\partial^2 U}{\partial y^2}(x,y) = b(x), \ y=\ell_2, \ 0 \le x \le \ell_1 \\ & U(x,y) = c(x), \ \frac{\partial^2 U}{\partial y^2}(x,y) = d(x), \ y=0, \ 0 \le x \le \ell_1 \\ & U(x,y) = e(y), \ \frac{\partial U}{\partial x}(x,y) = f(x), \ x=0, \ 0 \le y \le \ell_2 \\ & U(x,y) = g(y), \ \frac{\partial U}{\partial x}(x,y) = h(y), \ x=\ell_1, 0 \le y \le \ell_2 . \end{aligned}\right\} \qquad (6.5.21)$$

By applying the finite difference scheme (6.5.6) and using the column-wise ordering as before, the Biharmonic equation (6.5.3) together with the mixed boundary conditions (6.5.21) give the block five diagonal matrix equation,

$$M_E \underline{u} = \underline{d}_E \tag{6.5.22}$$

where $M_E$ is given by,

$$M_E = \begin{bmatrix} T^2+2\theta_x^2 I & -2\theta_x T & \theta_x^2 I & & & & \\ -2\theta_x T & T^2+2\theta_x^2 I & -2\theta_x T & \theta_x^2 I & & & \\ \theta_x^2 I & -2\theta_x T & T^2+2\theta_x^2 I & -2\theta_x T & \theta_x^2 I & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \theta_x^2 I & -2\theta_x T & T^2+2\theta_x^2 I & -2\theta_x T & \theta_x^2 T \\ & & & \theta_x^2 I & -2\theta_x T & T^2+2\theta_x^2 & -2\theta_x T \\ & & & & \theta_x^2 T & -2\theta_x T & T^2+2\theta_x^2 \end{bmatrix} \tag{6.5.23}$$

and $\underline{d}_E$ can be obtained from the boundary conditions in which the forward differences are used to approximate $\frac{\partial U}{\partial x}$ at the boundaries, i.e.,

$\frac{\partial U}{\partial x} = \frac{1}{\Delta x}(U_{i+1,j} - U_{i,j}) + O(\Delta x)$.

Factorisation of the form (6.5.15) cannot be applied to the matrix $M_E$ in (6.5.23).

However, since the matrices $\{T^2+2\theta_x^2 I\}, \{-2\theta_x T\}, \{\theta_x^2 I\}$ and $\{T\}$ are commutative, they possess a common set of eigenvectors, and hence the spectral resolution method is immediately applicable to the solution of the system (6.5.22). The algorithm for this proceeds in an identical manner to the one given earlier in this section except that the matrices, $\Gamma_j$, $1 \leq j \leq m$, are now defined as,

$$\Gamma_j = \begin{bmatrix} \delta_j & \lambda_j & \omega_j & & & & \\ \lambda_j & \delta_j & \lambda_j & \omega_j & & & \\ \omega_j & \lambda_j & \delta_j & \lambda_j & \omega_j & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \omega_j & \lambda_j & \delta_j & \lambda_j & \omega_j \\ & & & \omega_j & \lambda_j & \delta_j & \lambda_j \\ & & & & \omega_j & \lambda_j & \delta_j \end{bmatrix} \tag{6.5.24}$$

for which we are required to solve,

$$\Gamma_j \underline{u}_j = \underline{d}_j, \qquad j=1,2,\ldots,m. \tag{6.5.25}$$

In this case, the CQRPP algorithm (4.10) is immediately applicable to the fast solution of (6.5.24).

Numerical Example

For numerical-comparisons, the two direct methods, i.e.,

1. The spectral resolution method (in which the CQRPP algorithm (4.10)) is employed in the solution of the resulting quin-diagonal systems of equation) and
2. The Factorisation method (which uses the BKREF algorithm in the solution of the resulting two coupled Poisson equations)

outlined in this section for the solution of the Biharmonic equation (6.5.3) were programmed in Fortran and run on the Loughborough University I.C.L. 1904S computer for various boundary conditions. One such model problem considered is equation (6.5.3) with the boundary conditions,

$$\left.\begin{aligned} &U(x,y) = \cos(\pi x/2), \ \frac{\partial^2 U}{\partial^2 y}(x,y) = 0 \text{ at } y=0,1, \ 0\leq x\leq 1; \\ &U(x,y) = 1, \ \frac{\partial^2 U}{\partial^2 x} = 0, \text{ at } x=0,1, \ 0\leq y\leq 1 \\ &\text{and} \quad \psi(x,y)=0 \end{aligned}\right\} \tag{6.5.26}$$

for which a uniform mesh spacing $\Delta x=\Delta y=h$ is adopted.

The theoretical solution of this example is not known; however the numerical results obtained from the two methods discussed above satisfy the finite difference equation (6.5.6) at any random discrete point in the region of interest and are in good agreement with each other to within 10 decimal places as shown in Tables (6.3) and (6.4).

The Solution, $U(x_i,y_i)$ of the Biharmonic Equation (6.5.3) with the Boundary Condition (6.5.26), Using BKREF Algorithm, for $x_i,y_i$=4h(4h)20h where h=1/21, n=m=20

(Unit: $\times 10^2$)

| | | | | | |
|---|---|---|---|---|---|
| 20h | -0.60697393661 | -0.58802376763 | -0.42995577143 | -0.02393741986 | -0.00476049291 |
| 16h | -0.25307986760 | -0.26358331724 | -0.19609616979 | -0.10974750703 | -0.02185807526 |
| 12h | -0.33938752074 | -0.36895157821 | -0.27873112116 | -0.15677535246 | -0.03126674062 |
| 8h | -0.32729494031 | -0.35363959612 | -0.26651152984 | -0.14977743853 | -0.02986423569 |
| 4h | -0.21506311077 | -0.22001733451 | -0.16283395129 | -0.09099006980 | -0.01811478156 |
| $y_j$ / $x_i$ | 4h | 8h | 12h | 16h | 20h |

TABLE 6.3

The Solution, $U(x_i,y_i)$ of the Biharmonic Equation (6.5.3) with the Boundary Condition (6.2.26), Using the CQRPP Algorithm in a Spectral Decomposition Strategy, for $x_i,y_i$=4h(4h)20h where h=1/21, n=m=20

(Unit: $\times 10^2$)

| | | | | | |
|---|---|---|---|---|---|
| 20h | -0.60697394196 | -0.56329344162 | -0.42995577772 | -0.02393741986 | -0.00476049296 |
| 16h | -0.25307986884 | -0.33081563171 | -0.19609616910 | -0.10974750449 | -0.02185807366 |
| 12h | -0.33938752276 | -0.36895158016 | -0.27873112086 | -0.15677534941 | -0.03126673848 |
| 8h | -0.32729494244 | -0.35363959833 | -0.26651152992 | -0.14977743590 | -0.02986423371 |
| 4h | -0.21506311221 | -0.22001733609 | -0.16283395162 | -0.09099006841 | -0.01811478039 |
| $y_j$ / $x_i$ | 4h | 8h | 12h | 16h | 20h |

TABLE 6.4

# CHAPTER 7

# FURTHER RELATED TOPIC:

# METHODS FOR DETERMINING THE EIGENVALUES OF PERIODIC TRIDIAGONAL MATRICES

## 7.1 INTRODUCTION

The standard algebraic eigenvalue problem is the determination of the non-trivial solution of the equation,

$$A\underline{u} = \lambda\underline{u} \tag{7.1.1}$$

where A is a given (n×n) matrix (with real or complex elements), $\lambda$ is an eigenvalue and $\underline{u}$ the eigenvector.

In this chapter, we shall present algorithms for the solution of (7.1.1), for cases in which the matrix A is diagonally dominant and periodic tridiagonal. Such eigenvalue problems which are associated with periodic tridiagonal matrices occur, for example, in the finite difference approximation of the Sturm-Liouville differential equation with periodic boundary conditions (Froberg (1965)) and in the modal analysis of floquet waves in composite materials (Yang and Lee (1974)).

The periodic characteristic Sturm Liouville problem can be defined by the self-adjoint differential equation,

$$\frac{d}{dx}(P(x)\frac{dy}{dx}) + q(x)y + \lambda r(x)y = 0, \tag{7.1.2}$$

$q(x),r(x)>0,\ p(x)\neq 0,$

where we seek the numerical values of $\lambda$ and y(x) which satisfy (7.1.2) in the range R:a≤x≤b, subject to the boundary conditions,

$$\left.\begin{aligned} y(a) &= y(b) \\ p(a)\frac{dy}{dx}(a) &= p(b)\frac{dy}{dx}(b) \end{aligned}\right\}. \tag{7.1.3}$$

Following the standard notation, we denote $y(x_i)$, $p(x_i)$ by $y_i, p_i$ respectively and then use the following approximation for the second difference operator, i.e.,

$$\frac{d}{dx}(p(x)\frac{dy}{dx}) = \frac{p_{i+\frac{1}{2}}(y_{i+1}-y_i)-p_{i-\frac{1}{2}}(y_i-y_{i-1})}{h^2} + O(h^2) \tag{7.1.4}$$

at each of the discrete points $x_i$, i=1,2,...,n, in the closed interval [a,b] where h=(b-a)/n. A substitution of (7.1.4) into (7.1.2) together with the periodicity condition (7.1.3) yields a set of homogeneous linear equations of the form,

$$A\underline{y} = \lambda h^2 R\underline{y} \tag{7.1.5}$$

where R is a diagonal matrix with elements $r_i>0$, i=1,2,...,n; and A is a periodic tridiagonal matrix of the form,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \tag{7.1.6}$$

with elements,

$$\left.\begin{aligned} b_i &= (p_{i+\frac{1}{2}}+p_{i-\frac{1}{2}})-q_i h^2 \\ a_i &= -p_{i-\frac{1}{2}} \\ c_i &= -p_{i+\frac{1}{2}} \end{aligned}\right\} \quad i=1,2,\ldots,n. \tag{7.1.7}$$

Further, if the coefficients p(x), q(x) and r(x) are also periodic functions, with period n, i.e., $p_{n+i}=p_i$, $q_{n+i}=q_i$ and $r_{n+i}=r_i$, then the matrix A becomes symmetric of the form,

$$\tilde{A} = \begin{bmatrix} b_1 & a_2 & & & a_1 \\ a_2 & b_2 & a_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a_n \\ a_1 & & & a_n & b_n \end{bmatrix} \tag{7.1.8}$$

Since R is a diagonal matrix and has positive elements, i.e., $r_i>0$ then equation (7.1.5) can be rewritten as,

$$C\underline{u} = \lambda\underline{u} \tag{7.1.9}$$

where,

$$\left.\begin{aligned} \underline{y} &= R^{-\frac{1}{2}}\underline{u} \\ \text{and} \quad C &= h^{-2}R^{-\frac{1}{2}}AR^{-\frac{1}{2}}. \end{aligned}\right\} \tag{7.1.10}$$

The matrix C has the same form as (7.1.6) for the unsymmetric case, and the form (7.1.7) for the symmetric case. Equation (7.1.9) represents the standard eigenvalue problem to be solved.

In section (7.2), we present a new periodic quotient-difference (Q.D.) algorithm based on a sparse periodic matrix factorisation involving a continued fraction expansion, for the determination of the eigenvalues of a symmetric periodic tridiagonal matrix; the method being an extension of the well-known Rutishauser's QD scheme (Rutishauser (1958)). A reduction of a periodic tridiagonal matrix to a special sparse lower Hessenberg form using a 'P-Q similarity transformation' (a variant of Rutishauser's LR scheme) is discussed in section (7.3). In section (7.4), the simple Sturm sequences of the reduced Hessenberg form are derived and then used in an implicit Bairstow's technique to find the eigenvalues of unsymmetric periodic tridiagonal matrices. For symmetric matrices, the Sturm sequences of the corresponding reduced Hessenberg form are used in a Newton-Raphson iteration to determine the eigenvalues in section (7.5). Finally, an iterative scheme for the determination of the smallest and largest eigen-system of sparse matrices is considered in section (7.6).

## 7.2 PERIODIC QUOTIENT-DIFFERENCE (P.Q.D.) ALGORITHM FOR THE DETERMINATION OF THE EIGENVALUES OF PERIODIC TRIDIAGONAL MATRICES

We consider the determination of the eigenvalues of the $(n\times n)$ symmetric positive definite periodic tridiagonal matrix,

$$A = \begin{bmatrix} b_1 & a_2 & & & & a_1 \\ a_2 & b_2 & a_3 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & a_n \\ a_1 & & & & a_n & b_n \end{bmatrix} . \qquad (7.2.1)$$

where A is assumed to be diagonally dominant and satisfies condition (2.4.20).

Based on the idea of Rutishauser's LR similarity transformation, the Quotient-Difference (QD) algorithm for the evaluation of the eigenvalues of a symmetric tridiagonal matrix is well known (see, for example, Schwarz et al (1973) and Wilkinson (1965)).

The fundamental notion of the LR (and hence the QD) scheme for the determination of the eigenvalues of any given square matrix A ($=A^{(1)}$ say) lies in a factorisation of $A^{(1)}$ into a product of a unit lower triangular matrix $L^{(1)}$ and an upper triangular matrix $R^{(1)}$, i.e.,

$$A^{(1)} = L^{(1)}R^{(1)}$$

and then re-multiply the two triangular matrices $L^{(1)}$ and $R^{(1)}$ in reverse order to form a new matrix $A^{(2)}$, i.e.,

$$A^{(2)} = R^{(1)}L^{(1)} ,$$

which is known to be similar to $A^{(1)}$ and hence preserves the eigenvalues. A repetition of this process leads to an infinite set of similar matrices $\{A^{(s)}\}=L^{(s)}R^{(s)}$.

Under certain convergence conditions it is shown (Rutishauser (1958), Wilkinson (1965)) that $L^{(s)}\to I$ (the identity matrix) and $R^{(s)}\to$ upper triangular matrix (as $s\to\infty$) and hence the eigenvalues $\lambda_i$, $(i=1,\ldots,n)$ which form the diagonal elements of $R^{(s)}$ can then be obtained.

We shall extend this method to periodic tridiagonal matrices of the form (7.2.1) to which we apply a *sparse (P-Q) cyclic factorisation* strategy instead of the more usual strictly lower triangular and strictly upper triangular (LU) decomposition.

By a suitable choice of a diagonal matrix D, the matrix A in (7.2.1) can in the first place, be reduced, by a simple similarity transformation, to the unsymmetric form,

$$A^{(1)} = DAD^{-1} = \begin{bmatrix} b_1^{(1)} & 1 & & & a_1^{(1)} \\ a_2^{(1)} & b_2^{(1)} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & a_n^{(1)} & b_n^{(1)} \end{bmatrix} \qquad (7.2.2)$$

where $a_i^{(1)} = a_i^2$

and $b_i^{(1)} = b_i$.

Now, the matrix $A^{(1)}$ can be factorised (see 3.2.12) into the form,

$$A^{(1)} = P^{(1)}Q^{(1)} \qquad (7.2.3)$$

where

$$P^{(1)} = \begin{bmatrix} 1 & & & \ell_1^{(1)} \\ \ell_2^{(1)} & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & \ell_n^{(1)} & 1 \end{bmatrix} \quad \text{and} \quad Q^{(1)} = \begin{bmatrix} u_1^{(1)} & 1 & & \\ & u_2^{(1)} & 1 & \\ & & \ddots & \ddots \\ & & & 1 \\ 1 & & & u_n^{(1)} \end{bmatrix} \qquad (7.2.4)$$

The sparsity and the band structure of the unsymmetric matrix is preserved by this cyclic decomposition.

From equations (7.2.3) and (7.2.2) we obtain the elements of $P^{(1)}$ and $Q^{(1)}$ as,

$$\left.\begin{aligned} \ell_i^{(1)} &= a_i^{(1)}/u_{i-1}^{(1)} \\ u_i^{(1)} &= b_i^{(1)} - \ell_i^{(1)} \\ u_0^{(1)} &\equiv u_n^{(1)} \end{aligned}\right\} \quad i=1,2,\ldots,n. \tag{7.2.5}$$

and

In order to evaluate $\ell_i^{(1)}$, $u_i^{(1)}$ $(i=1,2,\ldots,n)$ and hence complete the P-Q cyclic factorisation, one of the elements (preferably $\ell_1^{(1)}$) must be determined in some way before the other elements are then obtained easily using formulae (7.2.5). We shall adopt the periodic continued fraction (PCF) concept for this purpose.

Next, a remultiplication of $P^{(1)}$ and $Q^{(1)}$ in the reverse order gives,

$$A^{(2)} = Q^{(1)}P^{(1)}$$

or

$$\begin{bmatrix} b_1^{(2)} & 1 & & & a_1^{(2)} \\ a_2^{(2)} & b_2^{(2)} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & 1 \\ 1 & & & a_n^{(2)} & b_n^{(2)} \end{bmatrix} = \begin{bmatrix} u_1^{(1)}+\ell_2^{(1)} & 1 & & & u_1^{(1)}\ell_1^{(1)} \\ u_2^{(1)}\ell_2^{(1)} & u_2^{(1)}+\ell_3^{(1)} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & 1 \\ 1 & & & u_n^{(1)}\ell_n^{(1)} & u_n^{(1)}+\ell_1^{(1)} \end{bmatrix}$$

from which we derive,

$$\left.\begin{aligned} b_i^{(2)} &= u_i^{(1)}+\ell_{i+1}^{(1)} \\ a_i^{(2)} &= u_i^{(1)}\ell_i^{(1)}, \quad \ell_{n+1}\equiv\ell_1 \end{aligned}\right\} \quad i=1,2,\ldots,n. \tag{7.2.6}$$

and

Further, by a P-Q factorisation of $A^{(2)}$ we have,

$$A^{(2)} = P^{(2)}Q^{(2)}$$

from which we obtain a set of equations, analogous to (7.2.5), and given by,

$$\left.\begin{aligned} \ell_i^{(2)} &= a_i^{(2)}/u_{i-1}^{(2)} \\ u_i^{(2)} &= b_i^{(2)} - \ell_i^{(2)}, \quad u_0^{(2)}\equiv u_n^{(2)} \end{aligned}\right\} \quad i=1,2,\ldots,n. \tag{7.2.7}$$

A substitution of (7.2.6) into (7.2.7) leads to the relations,

$$\left.\begin{aligned} \ell_i^{(2)} &= u_i^{(1)}\ell_i^{(1)}/u_{i-1}^{(2)} \\ u_i^{(2)} &= u_i^{(1)}+\ell_{i+1}^{(1)}-\ell_i^{(2)} \end{aligned}\right\} \quad i=1,2,\ldots,n. \tag{7.2.8}$$

and

Generally, at the $s^{th}$ stage of the above process, a re-multiplication, followed by a decomposition, i.e.,

$$A^{(s)} = Q^{(s-1)}P^{(s-1)}$$

and

$$A^{(s)} = P^{(s)}Q^{(s)}$$

yields the elements of $P^{(s)}$ and $Q^{(s)}$ as:

$$\left.\begin{aligned} \ell_i^{(s)} &= u_i^{(s-1)}\ell_i^{(s-1)}/u_{i-1}^{(s)} \\ \text{and} \quad u_i^{(s)} &= u_i^{(s-1)} + \ell_{i+1}^{(s-1)} - \ell_i^{(s)} \\ u_0 &\equiv u_n, \quad \ell_{n+1} \equiv \ell_1 \end{aligned}\right\} \quad \begin{matrix} s=2,3,\ldots, \\ i=1,\ldots,n. \end{matrix} \qquad (7.2.9a)$$

A combination of equation (7.2.5), for the first step (s=1) and equation (7.2.9), for subsequent steps constitute the PQD scheme which can now be arranged in the so-called Quotient-Difference chart with the elements $\ell_i^{(s)}, u_i^{(s)}$, (i=1,2,...,n), appearing alternately in each skew line as shown in Table (7.1).

The Periodic Quotient-Difference (PQD) Chart

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\ell_1^{(1)}$ | $u_1^{(1)}$ | | | | | | | |
| $\ell_1^{(2)}$ | $u_1^{(2)}$ | $\ell_2^{(1)}$ | $u_2^{(1)}$ | | | | | |
| $\ell_1^{(3)}$ | $u_1^{(3)}$ | $\ell_2^{(2)}$ | $u_2^{(2)}$ | $\ell_3^{(1)}$ | $u_3^{(1)}$ | | | |
| $\ell_1^{(4)}$ | $u_1^{(4)}$ | $\ell_2^{(3)}$ | $u_2^{(3)}$ | $\ell_3^{(2)}$ | $u_3^{(2)}$ | $\ell_4^{(1)}$ | $u_4^{(1)}$ | |
| $\ell_1^{(5)}$ | $u_1^{(5)}$ | $\ell_2^{(4)}$ | $u_2^{(4)}$ | $\ell_3^{(3)}$ | $u_3^{(3)}$ | $\ell_4^{(2)}$ | $u_4^{(2)}$ | $\ell_5^{(1)}$ |
| $\ell_1^{(6)}$ | $u_1^{(6)}$ | $\ell_2^{(5)}$ | $u_2^{(5)}$ | $\ell_3^{(4)}$ | $u_3^{(4)}$ | $\ell_4^{(3)}$ | $u_4^{(3)}$ | $\ell_5^{(2)}$ |
| $\ell_1^{(7)}$ | $u_1^{(7)}$ | $\ell_2^{(6)}$ | $u_2^{(6)}$ | $\ell_3^{(5)}$ | $u_3^{(5)}$ | $\ell_4^{(4)}$ | $u_4^{(4)}$ | $\ell_5^{(3)}$ |
| | | $\ell_2^{(7)}$ | $u_2^{(7)}$ | $\ell_3^{(6)}$ | $u_3^{(6)}$ | $\ell_4^{(5)}$ | $u_4^{(5)}$ | $\ell_5^{(4)}$ |

TABLE 7.1

As soon as a skew-line (corresponding to one P-Q similarity transformation) has been determined the next one is computed element by

element from left to right by using the relations in (7.2.9) which can be interpreted in terms of the so-called rhombus rules consisting of the *quotient* and *difference* rules.

The Quotient Rule

Each $q_i^{(s)}$ (i=1,2,...,n) term is obtained from a rhombus centered on the $q_i^{(s)}$ column of the PQD chart (Figure 7.1a) such that the product of the lower elements of the rhombus equals the product of the two upper elements, i.e.,

$$q_i^{(s)} = \frac{u_i^{(s-1)} q_i^{(s-1)}}{u_{i-1}^{(s-1)}} \quad , \; i=2,3,\ldots,n, \tag{7.2.9b}$$

where $u_0^{(s-1)} \equiv u_n^{(s-1)}$.

The Difference Rule

The $u_i^{(s)}$ term is obtained from a rhombus of the form centred on the $u_i^{(s)}$ column of the PQD chart (Figure 7.1b) such that the sum of the two lower elements equals that of the upper elements, i.e.,

$$u_i^{(s)} = u_i^{(s-1)} + q_{i+1}^{(s-1)} - q_i^{(s)} \quad , \; i=1,2,\ldots,n. \tag{7.2.9c}$$

where $q_{n+1}^{(s-1)} \equiv q_1^{(s-1)}$.

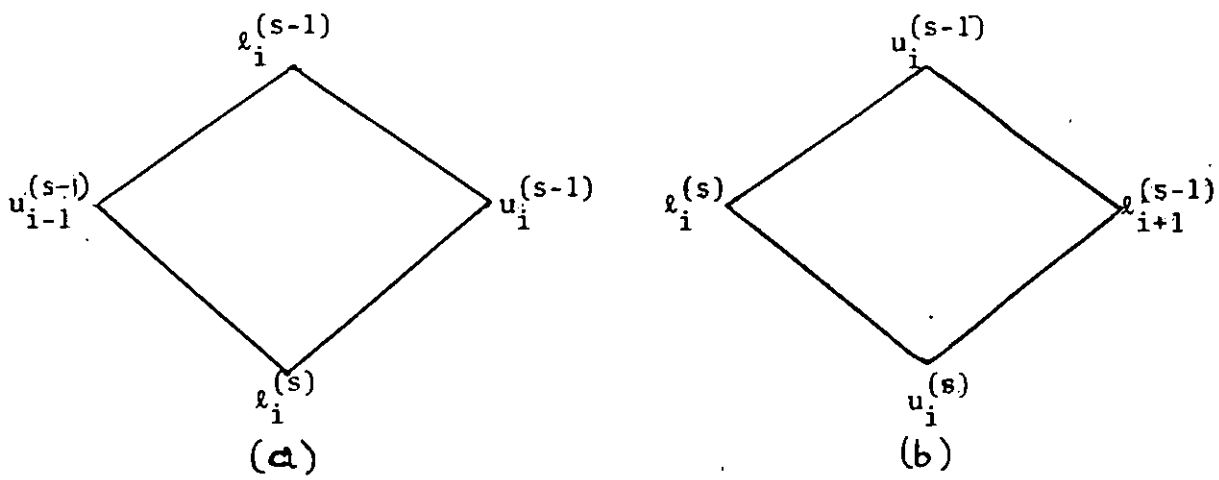



FIGURE 7.1

The Rhombus Rules

Determination of $\ell_1^{(s)}$

At each step of the PQD scheme, the $\ell_1^{(s)}$ (s=1,2,...), must be determined first before the rhombus rules can be applied in computing other elements of a given skew line of the PQD chart.

For the first step, the $\ell_1^{(1)}$ is derived as an infinite periodic continued fraction by a cyclic application of the formula (7.2.5) to give,

$$\ell_1^{(1)} = \underbrace{\frac{a_1^{(1)}}{b_n^{(1)}} - \frac{a_n^{(1)}}{b_{n-1}^{(1)}} - \frac{a_{n-1}^{(1)}}{b_{n-2}^{(1)}} - \cdots \frac{a_2^{(1)}}{b_1^{(1)}} -}_{\text{1st cycle}} \underbrace{\frac{a_1^{(1)}}{b_n^{(1)}} - \frac{a_n^{(1)}}{b_{n-1}^{(1)}} - \frac{a_{n-1}^{(1)}}{b_{n-2}^{(1)}} - \cdots \frac{a_2^{(1)}}{b_1^{(1)}} -}_{\text{2nd cycle}} \cdots \frac{a_1^{(1)}}{b_n^{(1)}} - \cdots \quad \ldots \tag{7.2.10}$$

where we have maintained the notation for continued fraction introduced in (2.4.4). Equation (7.2.10) can be written in the general form as,

$$\ell_1^{(1)} = \underbrace{\frac{\alpha_1}{\beta_1} - \frac{\alpha_2}{\beta_2} - \frac{\alpha_3}{\beta_3} - \cdots \frac{\alpha_n}{\beta_n} -}_{\text{1st cycle}} \underbrace{\frac{\alpha_1}{\beta_1} - \frac{\alpha_2}{\beta_2} - \frac{\alpha_3}{\beta_3} - \cdots \frac{\alpha_n}{\beta_n} -}_{\text{2nd cycle}} \frac{\alpha_1}{\beta_1} - \cdots \quad \ldots \tag{7.2.11}$$

where
$$\left.\begin{aligned} \alpha_i &= a^{(1)}_{(n-i+2)} \\ \beta_i &= b^{(1)}_{(n-i+1)} \end{aligned}\right\} \quad \begin{aligned} &i=1,\ldots,n, \\ &(k) \equiv k \,(\text{modulo } n). \end{aligned}$$

The infinite periodic continued fraction (7.2.11) is said to be generated by the linear fractional transformation,

$$\tau^{(n)}(\omega) = \frac{\alpha_1}{\beta_1} - \frac{\alpha_2}{\beta_2} - \frac{\alpha_3}{\beta_3} - \cdots \frac{\alpha_n}{\beta_n} - \tag{7.2.12}$$

where $\omega$ is known as a fixed point of the transformation at infinity (Wall (1948)).

Similarly, for the $s^{th}$ PQD step, $\ell_1^{(s)}$ (s=2,3,...) is derived as an infinite continued fraction by a cyclic application of the formula (7.2.9a), to give,

$$\ell_1^{(s)} = \underbrace{\frac{u_1^{(s-1)}\ell_1^{(s-1)}}{u_n^{(s-1)}+\ell_1^{(s-1)}} - \frac{u_n^{(s-1)}\ell_n^{(s-1)}}{u_{n-1}^{(s-1)}+\ell_n^{(s-1)}} - \frac{u_{n-1}^{(s-1)}\ell_{n-1}^{(s-1)}}{u_{n-2}^{(s-1)}+\ell_{n-1}^{(s-1)}} - \cdots \frac{u_2^{(s-1)}\ell_2^{(s-1)}}{u_1^{(s-1)}+\ell_2^{(s-1)}} -}_{\text{1st cycle}} \underbrace{\frac{u_1^{(s-1)}\ell_1^{(s-1)}}{u_n^{(s-1)}+\ell_1^{(s-1)}} - \frac{u_n^{(s-1)}\ell_n^{(s-1)}}{u_{n-1}^{(s-1)}+\ell_n^{(s-1)}} - \cdots \frac{u_2^{(s-1)}\ell_2^{(s-1)}}{u_1^{(s-1)}+\ell_2^{(s-1)}} -}_{\text{2nd cycle}} \frac{u_1^{(s-1)}\ell_1^{(s-1)}}{u_n^{(s-1)}+\ell_1^{(s-1)}} - \cdots \tag{7.2.13}$$

which can also be written in the standard form (7.2.11) with

$$\left.\begin{aligned} \alpha_i &= \ell_{(n-i+2)}{}^{u}{}_{(n-i+2)} \\ \text{and} \qquad \beta_i &= \ell_{(n-i+2)}{}^{+u}{}_{(n-i+1)} \end{aligned}\right\} \quad \begin{matrix} i=1,\ldots,n, \\ (k)\equiv k(\text{modulo } n). \end{matrix} \tag{7.2.14}$$

Under the condition specified in (2.4.20) the matrix $A^{(s)}$ in the $s^{th}$ step of the PQD reduction process, the continued fractions (7.2.10) and (7.2.13) are always guaranteed to converge (Theorem 2.13). Furthermore, the numerical determination is obtained by invoking Theorem (2.10) which states that the value of a periodic continued fraction of the form (7.2.11) is equal to the maximum of the two *fixed points* of the linear fractional transformation (7.2.12). An efficient method for the determination of the fixed points, $\omega$, is given in section 3.2 (see (3.2.27)-(3.2.32)) and involves the solution of the quadratic equation,

$$F_{k-1}\omega^2 + (F_k - E_{k-1})\omega - E_k = 0 \tag{7.2.15}$$

where the coefficients, $E_k, F_k$ are obtained from a forward recurrence relation,

$$\left.\begin{aligned} & E_0 = 0, \quad F_0 = 1, \\ & E_1 = \alpha_1, \; F_1 = \beta_1 \; , \\ \text{and} \qquad & E_i = \beta_i E_{i-1}, \; F_i = \beta_i F_{i-1} - \alpha_i F_{i-2}, \; i=2,\ldots,k\leq n. \end{aligned}\right\} \tag{7.2.16}$$

If the recurrence relation (7.2.16) converges for any $k<n$, then the truncation error criterion,

$$\left|\frac{E_k}{F_k} - \frac{E_{k-1}}{F_{k-1}}\right| < \varepsilon \quad , \tag{7.2.17}$$

must be satisfied, for a prescribed truncation error tolerance $\varepsilon$.

The above method for determining $\ell_1^{(s)}$ is best implemented in the form of a simple subroutine (see CF2 subroutine, Program 18 of Appendix I) so that at each step of the PQD process, the value of $\ell_1^{(s)}$ is obtained by a simple call to this periodic continued fraction subroutine.

Convergence of the PQD Algorithm

We expect that the PQD algorithm, as a special case of the Rutishauser's LR transformation, must converge, as does the latter and that $u_i^{(s)}$/the terms of

the PQD chart must converge to the eigenvalues of $A^{(1)}$ and the $\ell_i^{(s)}$ elements to zero. This will now be established by the following theorem.

Theorem (7.1)

The Periodic Quotient Difference algorithm for a positive definite periodic tridiagonal matrix A, satisfying condition (2.4.20), is convergent. The limiting values, $u_i^{(s)}=\lambda_i$, (as $s\to\infty$), are the eigenvalues of A arranged in order of magnitude, $\lambda_1>\lambda_2\ldots>\lambda_n$.

Before we give the proof of Theorem (7.1), we first consider the following lemma which will be required in our proof.

Lemma (7.1)

For a positive definite periodic tridiagonal matrix of the form (7.2.2) the initial values, $u_i^{(1)},\ell_i^{(1)}$ of the first skew line of the PQD chart are positive, i.e., $u_i^{(1)}>0,\ell_i^{(1)}>0$.

Proof of Lemma (7.1)

Let $p_i$ (i=1,...,n) denote the principal minors of order i of the positive definite periodic tridiagonal matrix (7.2.2), and let $p_0$ be defined as 1.

Further, we assume the following induction premise, i.e.,

$$u_i^{(1)} = p_i/p_{i-1} \quad , \ i=1,2,\ldots,k \ (1\leq k\leq n)$$

Now, by using (7.2.5), and the induction premise, we have,

$$u_i^{(1)} = b_i^{(1)} - \ell_i^{(1)} = b_i^{(1)} - a_i^{(1)}/u_{i-1}^{(1)} = b_i^{(1)} - a_i^{(1)} p_{i-2}/p_{i-1} \ ,$$

which yields,

$$u_i^{(1)} = (p_{i-1}b_i^{(1)} - p_{i-2}a_i^{(1)})/p_{i-1} \ .$$

On using the definition of $p_i$ this simplifies to,

$$u_i^{(1)} = p_i/p_{i-1} \ .$$

In order to check the induction, we consider k=2 and obtain,

$$u_2^{(1)} = b_2^{(1)} - \ell_2^{(1)} = b_2^{(1)} - \frac{a_2^{(1)}}{u_1^{(1)}} = b_2^{(1)} - a_2^{(1)} p_0/p_1 = b_2^{(1)} - \frac{a_2^{(1)}}{b_1^{(1)}} .$$

This gives, $u_2^{(1)} = \dfrac{b_2^{(1)} b_1^{(1)} - a_2^{(1)}}{b_1^{(1)}}$

which, by definition, is equal to $p_2/p_1$.

Hence the induction premise is true for k=2. Further checks confirm that it is also true for values of k=3,4,...,n.

The principal minors of a positive definite matrix are known to be positive; hence the $u_i^{(1)}$ terms which are expressed as a ratio of two positive numbers, must therefore be positive. Furthermore, since $a_i^{(1)}>0$ and $u_{i-1}^{(1)}>0$ then, by using (7.2.5) we have,

$$\ell_i^{(1)} = a_i^{(1)}/u_{i-1}^{(1)} > 0 ;$$

and this completes the proof of Lemma (7.1).

This property of the first skew line of the PQD chart having only positive entries is sustained in subsequent skew lines, since such lines can be seen as factorisations of positive definite matrices which are similar to $A^{(1)}$.

The proof of the LR algorithm given in Wilkinson (1965) covers theorem (7.1) but we present an alternative proof below following closely Schwarz et al (1973), P.179.

Proof of Theorem (7.1)

From the difference rule in equation (7.2.9c) we have,

$$u_i^{(s)} = u_i^{(s-1)} + \ell_{i+1}^{(s-1)} - \ell_i^{(s)}$$

and

$$u_{i-1}^{(s+1)} = u_{i-1}^{(s)} + \ell_i^{(s)} - \ell_{i-1}^{(s+1)} .$$

By adding these two equations together we obtain,

$$u_i^{(s)} + u_{i-1}^{(s+1)} = u_i^{(s-1)} + u_{i-1}^{(s)} - (\ell_{i-1}^{(s+1)} - \ell_{i+1}^{(s-1)}) . \qquad (7.2.18)$$

Since the eigenvalues are assumed to be ordered in the form, $\lambda_1 > \lambda_2 \ldots > \lambda_n$, then $(\ell_{i-1}^{(s+1)} - \ell_{i+1}^{(s-1)}) \geq 0$.

Thus, using the positiveness of the PQD technique, we conclude using (7.2.18) that the sequence, $(u_i^{(s)} + u_{i-1}^{(s+1)})$ is a monotonically decreasing sequence for increasing s and is bounded from below by zero.

Hence,

$$\lim_{s\to\infty} (\ell_{i-1}^{(s+1)} - \ell_{i+1}^{(s-1)}) = 0. \tag{7.2.19}$$

One possibility of (7.2.19) is that

$$\lim_{s\to\infty} \ell_{i-1}^{(s+1)} = 0, \quad \lim_{s\to\infty} \ell_{i+1}^{(s-1)} = 0.$$

By a similar argument, we conclude, in general, that for any i, $\ell_i^{(s)} \to 0$ and therefore by using the equations of the difference rule in (7.2.9c), the term $u_i^{(s)}$ must also converge to a limit, which is an eigenvalue.

Acceleration of Convergence of the PQD Algorithm by a Coordinate Translation

Following Wilkinson (1965), we state immediately that the terms $\ell_i^{(s)}$ converge to zero (or the terms $u_i^{(s)}$ to the eigenvalues $\lambda_i$) roughly as the quotient $(\lambda_{i+1}/\lambda_i)^s$, where $\lambda_1 > \lambda_2 \ldots > \lambda_i > \ldots > \lambda_n$ are the ordered eigenvalues of the given matrix A (say). This means that if the separation of some of the eigenvalues is poor the convergence of $\ell_i^{(s)}$ to zero may be slow.

By considering the matrix (A-yI), whose eigenvalues are $(\lambda_i - y)$, where y is a coordinate translation or a shift in the matrix origin, the convergence rate is improved through the decrement of the convergence quotient to,

$$\{(\lambda_{i+1}-y)/(\lambda_i-y)\}^s .$$

This quotient decreases rapidly if y is a good approximation to $\lambda_{i+1}$. However, for the PQD technique, the shift y must not be too large as to destroy its positiveness property.

An introduction of a shift $y_s$ into the $s^{th}$ PQD step leads to the following modified algorithm:-

First PQD step:

We compute $\ell_i^{(1)}, u_i^{(1)}$ from the relation,

$$\left.\begin{aligned} \ell_i^{(1)} &= a_i^{(1)}/u_{i-1}^{(1)}, \quad i=2,3,\ldots,n, \\ u_i^{(1)} &= b_i^{(1)} - \ell_i^{(1)}, \quad i=1,2,\ldots,n \end{aligned}\right\} \tag{7.2.20}$$

where $\ell_1^{(1)}$ is given by the infinite continued fraction given in (7.2.10).

The $s^{th}$ step, (s=2,3,...,):

We choose a shift $y_s$ ($<\lambda_{min}(A^{(s)})$) and then compute $\ell_i^{(s)}, u_i^{(s)}$, from the relations,

$$\left.\begin{aligned} \ell_i^{(s)} &= u_i^{(s-1)}\ell_i^{(s-1)}/u_{i-1}^{(s)}, \quad i=2,3,\ldots,n \\ \text{and} \quad u_i^{(s)} &= u_i^{(s-1)}+\ell_{i+1}^{(s-1)}-\ell_i^{(s)}-y_s, \quad i=1,\ldots,n. \end{aligned}\right\} \tag{7.2.21}$$

together with,

$$z_0 = 0, \quad z_s = z_{s-1}+y_s$$

where $z_s$ is the accumulation of the shifts applied at each step. Before evaluating the $\ell_i^{(s)}, u_i^{(s)}$ terms using (7.2.21), the term $\ell_1^{(s)}$ is first derived from the periodic continued fraction of the form (7.2.13) but with the shift $y_s$ subtracted from each partial denominator, i.e.,

$$\ell_1^{(s)} = \frac{\ell_1^{(s-1)}u_1^{(s-1)}}{u_n^{(s-1)}+\ell_1^{(s-1)}-y_s-} \frac{u_n^{(s-1)}\ell_n^{(s-1)}}{u_{n-1}^{(s-1)}+\ell_n^{(s-1)}-y_s-} \frac{u_{n-1}^{(s-1)}\ell_{n-1}^{(s-1)}}{u_{n-2}^{(s-1)}+\ell_n^{(s-1)}-y_s-\ldots}$$

$$\frac{u_2^{(s-1)}\ell_2^{(s-1)}}{u_1^{(s-1)}+\ell_2^{(s-1)}-y_s-} \frac{\ell_1^{(s-1)}u_1^{(s-1)}}{u_n^{(s-1)}+\ell_1^{(s-1)}-y_s-} \frac{u_n^{(s-1)}\ell_n^{(s-1)}}{u_{n-1}^{(s-1)}+\ell_n^{(s-1)}-y_s-}$$

$$\frac{u_{n-1}^{(s-1)}\ell_{n-1}^{(s-1)}}{u_{n-2}^{(s-1)}+\ell_n^{(s-1)}-y_s\ldots} \frac{u_2^{(s-1)}\ell_2^{(s-1)}}{u_1^{(s)}+\ell_2^{(s-1)}-y_s-\ldots} \quad . \tag{7.2.22}$$

If each shift $y_s$ is suitably chosen to be as large as possible, but smaller than the smallest eigenvalue of $A^{(s)}$ then, the $\ell_i^{(s)}$ terms converge very rapidly to zero and the $u_i^{(s)}$ terms also converge to zero. The $z_s$ term converges to the smallest eigenvalue, so that as soon as the $\ell_n^{(s)}$ term is sufficiently small, within a given tolerance, then $(z_s+u_n^{(s)})$ represents the smallest eigenvalue, $\lambda_n$.

For the continuation of the method, in order to compute other eigenvalues, we apply a deflation technique, whereby the $\ell_n^{(s)}$ and $u_n^{(s)}$ columns of the PQD chart are omitted, so that once more, the smallest eigenvalues turn up in increasing order.

The Choice of a Shift, $y_s$ at the $s^{th}$ PQD Step

The choice of a suitable shift $y_s$ at the $s^{th}$ PQD reduction step presents some difficulties. There is the need, on the one hand, to choose a shift as close as possible to the current smallest eigenvalue of $A^{(s)}$ in order to speed up convergence, and on the other, the need to keep the shift small enough to retain the positive property of the PQD algorithm as well as ensure the convergence of the continued fraction (7.2.22) from which $\ell_1^{(s)}$ is determined.

The following ad-hoc shift strategy was adopted and built into the PQD algorithm (Program 18 of Appendix I).

At the $s^{th}$ PQD step, we choose a shift,

$$y_s = \phi(u_n^{(s-1)}+\ell_1^{(s-1)}) \tag{7.2.23}$$

where $\phi$ is a factor which takes the values,

$$\phi_1 = 0.5$$

$$\phi_k = (1+\phi_{k-1})\phi_1 \ , \ k=2,3,\ldots, \tag{7.2.24}$$

and $(u_n^{(s-1)}+\ell_1^{(s-1)})$ is a crude estimate of the smallest eigenvalue of $A^{(s-1)}$. Thus at the $s^{th}$ PQD step, a trial shift $y_s$ is employed. If this trial is successful, (i.e., the infinite continued fraction for the evaluation of $\ell_1^{(s)}$ converges and the terms, $\ell_i^{(s)}, u_i^{(s)}$, $i=1,\ldots,n$, are all positive) then the total shift $z_s$ (representing the cumulative sum of individual shifts) is logged and the factor $\phi$ is increased from $\phi_k$ to $\phi_{k+1}$ for use in a subsequent shift strategy. However, if the trial shift $y_s$ leads to a negative PQD (i.e. $u_i^{(s)}, \ell_i^{(s)} < 0$ for at least one i) then the shift is retracted. Progressively, smaller shifts, $y_s=\phi(u_n^{(s-1)}+\ell_1^{(s-1)})$ (where $\phi$ takes the values, $\frac{1}{2}\phi, \frac{1}{4}\phi, \frac{1}{8}\phi, \ldots$) are adopted in turn until a successful trial is achieved and the total shift is then logged to complete that step of the PQD process.

Numerical Result

The Periodic Quotient Difference (PQD) algorithm was programmed in

Fortran and run on the Loughborough University 1904S computer. The subroutine is given as Program 18 of Appendix I.

To test the PQD program, we consider the (10×10) diagonally dominant tridiagonal matrix,

$$A = \begin{bmatrix} 4 & 1 & & & & & & & & 1 \\ 1 & 6 & 1 & & & & & & & \\ & 1 & 8 & 1 & & & & & & \\ & & 1 & 10 & 1 & & & & & \\ & & & 1 & 12 & 1 & & & & \\ & & & & 1 & 14 & 1 & & & \\ & & & & & 1 & 16 & 1 & & \\ & & & & & & 1 & 18 & 1 & \\ & & & & & & & 1 & 20 & 1 \\ 1 & & & & & & & & 1 & 22 \end{bmatrix}_{(10\times10)} \qquad (7.2.25)$$

which can easily be verified to satisfy condition (2.4.20).

The results obtained from the PQD procedure are compared with those given by the Nottingham Algorithm Group (N.A.G.) Library routine F02AFF (which computes all the eigenvalues by the QR transformation method of Francis (1961/62)). The results obtained for the matrix in (7.2.25) are given in Table (7.2). These results, whilst not conclusive, indicate that an accuracy of 7 significant figures can be obtained by the PQD scheme.

The Eigenvalues of a Periodic Tridiagonal Matrix Using the PQD Technique

| $\lambda_i$ | PQD Scheme: Program 18 | N.A.G. (F02AFF) Routine |
|---|---|---|
| 1 | 3.504298196 | 3.504298100 |
| 2 | 5.943003306 | 5.943003271 |
| 3 | 7.997003770 | 7.997003268 |
| 4 | 9.999922587 | 9.999923084 |
| 5 | 11.999998885 | 11.999997609 |
| 6 | 14.000001193 | 14.000002391 |
| 7 | 16.000077639 | 16.000076916 |
| 8 | 18.002996412 | 18.002996733 |
| 9 | 20.056996807 | 20.056996730 |
| 10 | 22.495701134 | 22.495701899 |

TABLE 7.2

## 7.3 A REDUCTION OF A PERIODIC TRIDIAGONAL MATRIX TO A SPARSE LOWER HESSENBERG FORM BY THE P-Q SIMILARITY TRANSFORMATION

Definition 7.1

An $(n\times n)$ matrix $H=(h_{i,j})$ is said to be of lower Hessenberg form if

$$h_{i,j} = 0, \quad j\geq i+2.$$

For the purpose of this section, we further define a matrix $H=(h_{i,j})$ to be of sparse cyclic lower Hessenberg form if

$$h_{i,j} = 0, \quad |j-i| > 1; \; h_{n,1}\neq 0$$

and may be written, for n=5, in the form,

$$\begin{bmatrix} x & x & & & 0 \\ x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ x & 0 & & x & x \end{bmatrix} \tag{7.3.1}$$

where x denotes the location of non-zero entries.

Next, we consider a positive definite/periodic tridiagonal matrix satisfying (2·4·20), i.e.,

$$A = \begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \tag{7.3.2}$$

and assume that the diagonal elements of A/also satisfy the condition,

$$b_1 < b_2 <\ldots< b_n \; . \tag{7.3.3}$$

If A is diagonally dominant then condition (7.3.3) implies that

$$\lambda_1 < \lambda_2 <\ldots< \lambda_n \; . \tag{7.3.4}$$

One useful observation is that a periodic tridiagonal matrix (7.3.2) which satisfies condition (7.3.3) can be reduced to a sparse cyclic lower Hessenberg matrix, H of the form (7.3.1) by a repeated P-Q similarity transformation of A. One immediate advantage of such a reduction is that from the sparse Hessenberg form, a much simpler and compact Sturm sequence of polynomials than those obtainable from A can be derived for

use in the determination of the eigenvalues of H (and hence A) by the Bairstow's, Newton's, bisection or any other root-finding procedures.

Firstly, we assume that the matrix A has been reduced by a similarity transformation to the more convenient form,

$$A^{(1)} = \begin{bmatrix} b_1^{(1)} & 1 & & & a_1^{(1)} \\ a_2^{(1)} & b_2^{(1)} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & a_n^{(1)} & b_n^{(1)} \end{bmatrix} \tag{7.3.5}$$

where
$$\left.\begin{aligned} a_i^{(1)} &= a_i c_{i-1} \\ b_i^{(1)} &= b_i \end{aligned}\right\} \quad \begin{aligned} &i=1,2,\ldots,n \\ &c_0 \equiv c_n. \end{aligned} \tag{7.3.6}$$

Starting with $A^{(1)}$, the $s^{th}$ step, (s=1,2,...) of the PQ similarity transformation consists partly of the factorisation of $A^{(s)}$ to yield,

$$A^{(s)} = P^{(s)}Q^{(s)} \tag{7.3.7}$$

where $P^{(s)}, Q^{(s)}$ are cyclic lower and upper triangular matrices respectively defined as in (7.2.4). In matrix notation (7.3.7) can be written in the form,

$$\begin{bmatrix} b_1^{(s)} & 1 & & & a_1^{(s)} \\ a_2^{(s)} & b_2^{(s)} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & a_n^{(s)} & b_n^{(s)} \end{bmatrix} = \begin{bmatrix} u_1^{(s)}+\ell_1^{(s)} & 1 & & & \ell_1^{(s)}u_n^{(s)} \\ \ell_2^{(s)}u_1^{(s)} & u_2^{(s)}+\ell_2^{(s)} & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 1 & & & \ell_n^{(s)}u_{n-1}^{(s)} & u_n^{(s)}+\ell_n^{(s)} \end{bmatrix}. \tag{7.3.8}$$

Next, by a reversed multiplication of $P^{(s)}$ and $Q^{(s)}$ we obtain,

$$A^{(s+1)} = Q^{(s)}P^{(s)} \tag{7.3.9}$$

which can also be written in matrix form as,

$$\begin{bmatrix} b_1^{(s+1)} & 1 & & & a_1^{(s+1)} \\ a_2^{(s+1)} & b_2^{(s+1)} & 1 & & \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & 1 \\ 1 & & & a_n^{(s+1)} & b_n^{(s+1)} \end{bmatrix} = \begin{bmatrix} u_1^{(s)}+\ell_2^{(s)} & 1 & & & u_1^{(s)}\ell_1^{(s)} \\ u_2^{(s)}\ell_2^{(s)} & u_2^{(s)}+\ell_2^{(s)} & 1 & & \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \ddots & 1 \\ 1 & & & u_n^{(s)}\ell_n^{(s)} & u_n^{(s)}+\ell_1^{(s)} \end{bmatrix}. \tag{7.3.10}$$

Our interest is to show that under condition (7.3.3) the above P-Q similarity transformation produces a monotonically decreasing sequence, $a_1^{(1)}, a_1^{(2)}, \ldots a_1^{(s)}, a_1^{(s+1)}, \ldots$ .

Now let $r_s$ denote the ratio of the right-hand corner elements of $A^{(s+1)}$ and $A^{(s)}$. Then, from (7.3.8) and (7.3.10) we have,

$$r_s = a_1^{(s+1)}/a_1^{(s)} = u_1^{(s)}/u_n^{(s)}, \quad s=1,2,\ldots \tag{7.3.11}$$

Under condition (7.3.3) $u_1^{(s)}<u_2^{(s)}<\ldots<u_n^{(s)}$, $s=1,2,\ldots$

and hence using (7.3.11) it follows that

$$0 < r_s < 1, \quad s=1,2,\ldots . \tag{7.3.12}$$

Also from (7.3.11) we have,

$$a_1^{(s+1)} = r_s a_1^{(s)} = r_s r_{s-1} a_1^{(s-1)} = r_s r_{s-1} r_{s-2} a_1^{(s-2)} \tag{7.3.13}$$

and generally,

$$a_1^{(s+1)} = r_s r_{s-1} \ldots r_1 a_1^{(1)} \approx \hat{r}^s a_1^{(1)} \tag{7.3.14}$$

where, for similarity reasons, $\hat{r}$ is of the same order of magnitude as any $r_i$ $(i=1,2,\ldots,s)$ and hence may be estimated by the easily determinable value of $r_1$, i.e.,

$$\hat{r} \approx r_1 = u_1^{(1)}/u_n^{(1)} \approx b_1^{(1)}/b_n^{(1)} = b_1/b_n. \tag{7.3.15}$$

Taking logarithms of both sides of (7.3.14) it follows that

$$s \approx \log[a_1^{(s+1)}/a_1^{(1)}]/\log \hat{r} \tag{7.3.16}$$

and hence an estimate of the number of P-Q similarity transformations

required to reduce the corner element $a_1^{(s+1)}$ to the order of $\varepsilon$, a given small specified tolerance, is given by,

$$s \simeq \log(\varepsilon/a_1^{(1)})/\log \hat{r}. \tag{7.3.17}$$

Case 1

For a periodic tridiagonal matrix (7.3.2) satisfying condition (7.3.3) the ratio $\hat{r}$ is small (i.e., $0<\hat{r}<<1$) and hence (for a given value of $a_1^{(1)}$) s is also small. This means that the reduction to Hessenberg form requires only a few steps of the P-Q similarity transformation and hence such a reduction approach is recommended in the eigenvalue determination of periodic tridiagonal systems that satisfy condition (7.3.3).

Example

Consider a (50×50) periodic tridiagonal matrix,

$$A = \begin{bmatrix} 2 & 1 & & & & 1 \\ 1 & 4 & 1 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & 98 & 1 \\ 1 & & & & 1 & 100 \end{bmatrix}_{(50\times 50)} \tag{7.3.18}$$

for which $\hat{r}\simeq 2/100=1/50$, and $a_1^{(1)}=1$. The number of P-Q similarity transformations required to reduce the $A_{1,50}$ corner-element to the order of $\varepsilon(\simeq 10^{-6})$ is given by,

$$s = \log 10^{-6}/\log(0.2) \simeq 4.$$

This is in agreement with actual experimental results obtained by using the SLHM subroutine (see Program 19 of Appendix I) which implements the PQ similarity reduction process as discussed in this section.

Case 2

For periodic tridiagonal matrices of the form (7.3.2) which do not satisfy condition (7.3.3), the estimate of $r_s$ and hence $\hat{r}$ as given in (7.3.12) and (7.3.15) will not always hold. Instead we may have,

$$r_s \geq 1 \quad \text{for some } s=1,2,... \tag{7.3.19}$$

and therefore the sequence $\{a_1^s\}$, s=1,2,...

is no longer monotonically decreasing but may instead oscillate about an initial value $a_1^{(1)}$.

Example

Consider the (10×10) periodic tridiagonal matrix of the form,

$$A = \begin{bmatrix} b & 1 & & & & & & & & 1 \\ 1 & b & 1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & & \ddots & \ddots & \ddots & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & \ddots & \ddots & \ddots & & \\ & & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & \ddots & \ddots & 1 \\ 1 & & & & & & & & 1 & b \end{bmatrix}_{(10\times10)} \quad \text{, for any } b\geq 2. \tag{7.3.20}$$

In this example, the ratio $\hat{r}=1$.

By applying the P-Q similarity transformation, the sequence $\{a_1^{(s)}\}$, s=1,2,..., oscillates about $a_1^{(1)}$ and hence cannot converge to zero. The proposed reduction to sparse Hessenberg form cannot therefore be recommended for matrices of the form (7.3.20).

It may be observed, however, that a matrix A of the form (7.3.20) can be written as a rank one perturbation of another matrix, B (say) such that

$$A = B - \sigma \underline{u}\underline{u}^T, \qquad \sigma = \text{constant} \tag{7.3.21}$$

where,

$$B = \begin{bmatrix} b & 1 & & & & & & & & 1 \\ 1 & b & 1 & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & & \ddots & \ddots & \ddots & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & \ddots & \ddots & \ddots & & \\ & & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & \ddots & \ddots & 1 \\ 1 & & & & & & & & 1 & b+\sigma \end{bmatrix} \quad , \quad \underline{u} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.3.22}$$

The matrix B is different from A only in the last diagonal element of B which has been modified by the addition of the value σ, suitably chosen such that the ratio r of the first and last diagonal elements of B satisfy the condition,

$$0 < r=b/(b+\sigma) < 1. \tag{7.3.23}$$

It is now possible to reduce B to sparse Hessenberg form after only a few steps of the P-Q similarity transformation by a suitably chosen value of $\sigma$ such that,

$$0 < r=b/(b+\sigma) << 1. \tag{7.2.23}$$

Using perturbation theory, the eigenvalues of A can be related to those of B. If we denote the respective eigenvalues by $\lambda_i$ and $\mu_i$ (i=1,2,...,n) in descending order, then they satisfy the relations (Wilkinson (1965)),

$$\lambda_i = \mu_i - \sigma m_i \tag{7.3.24}$$

where $$0 \leq m_i \leq 1, \quad \sum_{i=1}^{n} m_i = 1, \tag{7.3.25}$$

i.e., the eigenvalues of A and B are interleaved.

The only consequence of the splitting of A as in (7.3.22) is that we can obtain the bounds for the eigenvalues of A once those of B have been found.

Finally, the above discussion on the reduction of a periodic tridiagonal matrix to a sparse cyclic lower Hessenberg form can be formalised in the following theorem:

Theorem 7.2

If $\{A^{(s)}\}$ is a sequence of similar matrices resulting from the P-Q similarity transformations of a given periodic tridiagonal matrix $A=A^{(1)}$ such that

$$\left.\begin{aligned} A^{(s)} &= P^{(s)}Q^{(s)} \\ \text{and} \quad A^{(s+1)} &= Q^{(s)}P^{(s)} \end{aligned}\right\} \quad s=1,2,\ldots$$

where $P^{(s)}, Q^{(s)}$ are as defined in (7.2.4), then under certain restrictions on the eigenvalues of A,

$P^{(s)}$ tends to I (the identity matrix)

and $Q^{(s)}$ tends to H, as $s\to\infty$, where H is a sparse cyclic lower Hessenberg form defined as in (7.3.1).

We consider first the following Lemma which is useful in the proof of Theorem (7.2).

Lemma 7.2

The matrix $[A^{(1)}]^s$ can be decomposed into the product of two matrices, $T_s$ and $V_s$, where $T_s$ is a product of cyclic lower triangular matrices $P^{(s)}$, and $V_s$ is a product of cyclic upper triangular matrices $Q^{(s)}$.

Proof

By the P-Q similarity transformation,

$$A^{(s)} = P^{(s)}Q^{(s)} \text{ and also } A^{(s)} = Q^{(s-1)}P^{(s-1)}, \quad s=1,2,\ldots,$$

and hence,

$$P^{(s)}Q^{(s)} = Q^{(s-1)}P^{(s-1)} . \tag{7.3.26}$$

Now, for simplicity, let us consider $[A^{(1)}]^s$, when s=4. Then,

$$[A^{(1)}]^4 = P^{(1)}Q^{(1)}P^{(1)}Q^{(1)}P^{(1)}Q^{(1)}P^{(1)}Q^{(1)}$$
$$= P^{(1)}\underline{Q^{(1)}P^{(1)}}\,\underline{Q^{(1)}P^{(1)}}\,\underline{Q^{(1)}P^{(1)}}Q^{(1)}$$

Using the relations in (7.3.26) we have,

$$[A^{(1)}]^4 = P^{(1)}P^{(2)}\underline{Q^{(2)}P^{(2)}}\,\underline{Q^{(2)}P^{(2)}}Q^{(2)}Q^{(1)}$$
$$= P^{(1)}P^{(2)}P^{(3)}\underline{Q^{(3)}P^{(3)}}Q^{(3)}Q^{(2)}Q^{(1)}$$
$$= P^{(1)}P^{(2)}P^{(3)}P^{(4)}Q^{(4)}Q^{(3)}Q^{(2)}Q^{(1)}$$
$$= T^{(4)}V^{(4)}$$

where
$$\left.\begin{aligned} T^{(4)} &= P^{(1)}P^{(2)}P^{(3)}P^{(4)} \\ V^{(4)} &= Q^{(1)}Q^{(2)}Q^{(3)}Q^{(4)} \end{aligned}\right\} \tag{7.3.27}$$

In general, it can be similarly shown that

$$[A^{(1)}]^s = T^{(s)}V^{(s)} \tag{7.3.28}$$

where,

$$T^{(s)} = P^{(1)}P^{(2)}\ldots P^{(s)} \tag{7.3.29}$$
$$V^{(s)} = Q^{(1)}Q^{(2)}\ldots Q^{(s)} .$$

Proof of Theorem 7.2

We assume that the (n×n) matrix $A(=A^{(1)})$ has real distinct eigenvalues,

$$\Lambda = (\lambda_1,\lambda_2,\ldots\lambda_n)$$

and that $X=(x_1,x_2,\ldots,x_n)$ is the corresponding independent matrix of column eigenvectors such that

$$A^{(1)}\underline{x}_i = \lambda\underline{x}_i, \quad i=1,2,\ldots,n.$$

Similarly, we let $Y=(y_1^T, y_2^T, \ldots, y_n^T)$ be the corresponding matrix of row (left-hand) eigenvectors such that

$$y_i^T A^{(1)} = \lambda_i y_i^T \quad i=1,2,\ldots,n.$$

XY is a diagonal matrix and if both X and Y are normalised, then, XY=YX=I. Hence we have,

$$A^{(1)} = X\Lambda X^{-1} = X\Lambda Y.$$

Similarly, by Lemma 5.1, $[A^{(1)}]^s$ can be written in the form,

$$[A^{(1)}]^s = X\Lambda^s Y. \tag{7.3.30}$$

Further, if we assume that the matrix of eigenvectors has a special periodic tridiagonal form / satisfying condition (2·4·20), then X and Y can be factorised into a P-Q form, i.e.,

$$\left.\begin{aligned} X &= P_x Q_x \\ Y &= P_y Q_y \end{aligned}\right\} \tag{7.3.31}$$

where $P_x, Q_x, P_y, Q_y$ can be written in the forms,

$$P_x = \begin{bmatrix} 1 & & & \ell_1 \\ \ell_2 & 1 & & 0 \\ & \ell_3 & \ddots & \\ 0 & & \ell_n & 1 \end{bmatrix}, \quad Q_x = \begin{bmatrix} u_1 & 1 & & \\ & u_2 & 1 & 0 \\ & & \ddots & \ddots \\ & 0 & & 1 \\ 1 & & & u_n \end{bmatrix} \tag{7.3.32a}$$

$$P_y = \begin{bmatrix} 1 & & & v_1 \\ v_2 & & 0 & \\ & \ddots & & \\ 0 & & v_n & 1 \end{bmatrix} \quad \text{and} \quad Q_y = \begin{bmatrix} w_1 & 1 & & \\ & w_2 & 1 & 0 \\ & & \ddots & \ddots \\ & 0 & & 1 \\ 1 & & & w_n \end{bmatrix}. \tag{7.3.32b}$$

Hence, we have,

$$\begin{aligned} [A^{(1)}]^s &= P_x Q_x \Lambda^s P_y Q_y \\ &= P_x Q_x [\Lambda^s P_y (\Lambda^{-1})^2] \Lambda^s Q_y. \end{aligned} \tag{7.3.33}$$

It is easily shown that $[A^{(1)}]^s$ is similar to $[\tilde{A}^{(1)}]^s$ where

$$[\tilde{A}^{(1)}]^s = P_x \Lambda^s \{Q_x [\Lambda^s P_y \Lambda^{-s}] \Lambda^s Q_y\} \Lambda^{-s} \tag{7.3.34}$$

and hence, using $[\tilde{A}^{(1)}]^s$ instead of $[A^{(1)}]^s$ in our similarity transformation argument we have,

$$[\tilde{A}^{(1)}]^s = P_x \Lambda^s Q_x [\Lambda^s P_y \Lambda^{-s}][\Lambda^s Q_y \Lambda^{-s}]$$
$$= P_x \hat{P}\hat{Q}\hat{R}$$

where $\hat{P} = \Lambda^s Q_x$,

$\hat{Q} = \Lambda^s P_y \Lambda^{-s}$

and $\hat{R} = \Lambda^s Q_y \Lambda^{-s}$ .

Next, by using equation (7.3.32) we can express $\hat{R}$ in matrix form as,

$$R = \begin{bmatrix} w_1 & (\lambda_1/\lambda_2)^s & & & \\ & w_2 & (\lambda_2/\lambda_3)^s & & 0 \\ & & \ddots & \ddots & \\ & 0 & & & (\lambda_{n-1}/\lambda_n)^s \\ (\lambda_n/\lambda_1)^s & & & & w_n \end{bmatrix} \qquad (7.3.35)$$

Similarly, $\hat{Q}$ can be written as,

$$\hat{Q} = \begin{bmatrix} 1 & & & & (\lambda_1/\lambda_n)^s v_1 \\ v_2(\ell_2/\ell_1)^s & 1 & & 0 & \\ & v_3(\ell_3/\ell_2)^s & 1 & \ddots & \\ & & \ddots & \ddots & \\ & 0 & & & v_n(\ell_n/\ell_{n-1})^s \quad 1 \end{bmatrix} \qquad (7.3.36)$$

and $\hat{P}$ as,

$$\hat{P} = \begin{bmatrix} u_1\lambda_1^s & \lambda_1^s & & & \\ & u_2\lambda_2^s & \lambda_2^s & & 0 \\ & & \ddots & \ddots & \\ & 0 & & & \lambda_{n-1}^s \\ \lambda_n^s & & & & u_n\lambda_n^s \end{bmatrix} \qquad (7.3.37)$$

If we assume that $\lambda_1 < \lambda_2 < \ldots \lambda_n$ (as in (7.3.4)) then as $s \to \infty$, $\hat{R}$ in (7.3.35) and $\hat{Q}$ in (7.3.36) converge to the forms $\tilde{R}, \tilde{Q}$ respectively, where,

$$\tilde{R} = \begin{bmatrix} w_1 & & & \\ & w_2 & & 0 \\ & & \ddots & \\ & 0 & & \\ & (\lambda_n/\lambda_1)^s & & w_n \end{bmatrix} \quad \text{and} \quad \tilde{Q} = \begin{bmatrix} 1 & & & & \\ (\lambda_2/\lambda_1)^s v_2 & 1 & & & \\ & (\lambda_3/\lambda_1)^s v_3 & 1 & & 0 \\ & & \ddots & \ddots & \\ & 0 & & (\lambda_n/\lambda_{n-1})^s v_n & 1 \end{bmatrix}$$

(7.3.38)

Hence as $s \to \infty$, $[A^{(1)}]^s = P_x \hat{P}\hat{Q}\hat{R}$ converges to a matrix of the form $P_x H$, where,

$$H = \hat{P}\tilde{Q}\tilde{R} = \begin{bmatrix} w_1 u_1 \lambda_1^s + \lambda_1^s (\frac{\lambda_2}{\lambda_1})^s v_2 w_1, & \lambda_1^2 w_2 & & \\ \lambda_2^s u_2 (\frac{\lambda_2}{\lambda_1})^s v_2 w_1, & w_2 u_2 \lambda_2^s + \lambda_2^s (\frac{\lambda_3}{\lambda_2})^s v_3 w_2, \lambda_2^s w_3 & & 0 \\ & & \ddots & \\ & 0 & & \lambda_{n-1}^s w_n \\ u_n \lambda_n^s (\frac{\lambda_n}{\lambda_1})^s & & \lambda_n^s u_n (\frac{\lambda_n}{\lambda_{n-1}})^s v_n w_{n-1}, & w_n u_n \lambda_n^s \end{bmatrix}$$

(7.3.39)

We have thus obtained a decomposition of $[\tilde{A}^{(1)}]^s$ in the form $P_x H$ where the factor $P_x$ given in (7.3.32a) is independent of s.

Also, by Lemma 7.1, $[\tilde{A}^{(1)}]^s$ can be factorised into the form,

$$[\tilde{A}^{(1)}]^s = T^{(s)} V^{(s)}$$

where $T^{(s)}, V^{(s)}$ are defined in (7.3.29).

Therefore, $T^{(s)} \to P_x$

and $V^{(s)} \to H$.

Since $T^{(s)} = P^{(1)} P^{(2)} \ldots P^{(s)} = T^{(s-1)} P^{(s)}$

it follows that if $T^{(s)}$ tends to a limit, $P^{(s)}$ must also tend to a limit. Since $T^{(s)} \to P_x$, then $P^{(s)}$ must therefore tend to a unit diagonal matrix I, which completes the proof.

## 7.4 A NEW APPROACH TO BAIRSTOW'S METHOD IN FINDING THE EIGENVALUES OF AN UNSYMMETRIC PERIODIC TRIDIAGONAL MATRIX

The Bairstow's method is a procedure for finding the real quadratic factors of a given polynomial; thus it can determine the roots of the polynomial in real or complex conjugate pairs without the need to use complex arithmetic. This method is useful in the determination of eigenvalues of matrices for which complex eigenvalues are expected. By finding the quadratic factors of the characteristic polynomials of such matrices implicitly the method obtains the eigenvalues in pairs as roots of the quadratic factors; leaving a reduced polynomial of order 2 less, upon which the process is repeated.

For an $(n \times n)$ periodic unsymmetric tridiagonal matrix satisfying condition (2.4.20)

i.e.,

$$A = \begin{bmatrix} b_1 & c_1 & & & & a_1 \\ a_2 & b_2 & c_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & c_{n-1} \\ c_n & & & & a_n & b_n \end{bmatrix} \tag{7.4.1}$$

the eigenvalues are given by the determinantal equation,

$$\det(A-\lambda I) = 0. \tag{7.4.2}$$

In order to calculate these eigenvalues by the Bairstow's method, we need to obtain the sequence of polynomials which form the leading principal minors of $(A-\lambda I)$, by a Laplace expansion of the matrix $(A-\lambda I)$.

Following Evans (1971B), the characteristic polynomials of the matrix $(A-\lambda I)$ are given by the relationships,

$$\left.\begin{aligned} P_{-1}(\lambda) &= 0 , \\ P_0(\lambda) &= 1 , \\ P_1(\lambda) &= b_1-\lambda , \\ P_2(\lambda) &= (b_2-\lambda)P_1(\lambda)-a_2c_1P_0(\lambda) , \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_i(\lambda) &= (b_i-\lambda)P_{i-1}(\lambda)-a_ic_{i-1}P_{i-2}(\lambda), \quad i=3,\ldots,n-1, \end{aligned}\right\} \tag{7.4.3}$$

and

$$\left.\begin{aligned} Q_0(\lambda) &= 0 \ , \\ Q_1(\lambda) &= 1 \ , \\ Q_2(\lambda) &= b_2-\lambda \ , \\ &\cdots\cdots\cdots\cdots\cdots \\ Q_i(\lambda) &= (b_i-\lambda)Q_{i-1}(\lambda)-a_ic_{i-1}Q_{i-2}(\lambda) \ , \ i=3,\ldots,n-1. \end{aligned}\right\} \qquad (7.4.4)$$

Finally,

$$P_n(\lambda) = (b_n-\lambda)P_{n-1}(\lambda)-a_nc_{n-1}P_{n-2}(\lambda)-a_1c_nQ_{n-1}(\lambda) + (-1)^{n-1}\prod_{i=1}^{n}a_i+(-1)^{n-1}\prod_{i=1}^{n}c_i \ . \qquad (7.4.5)$$

These polynomials $\{P_i\}$, i=1,...n, form a properly signed sequence of polynomials with $P_i(\lambda)>0$ for a sufficiently large value of $\lambda$, positive or negative. The zeros of $P_i(\lambda)$ separate those of $P_{i+1}(\lambda)$ and hence by Gerschgorin's separation theorem (Wilkinson (1965)), the sequence $\{P_i\}$ i=1,2,...,n, form a Sturm sequence of polynomials on the interval $(-\infty,\infty)$.

The eigenvalues of the unsymmetric matrix A in (7.4.1) can thus be determined, using Bairstow's implicit synthetic division procedure, by finding the quadratic factors of $P_n(\lambda)$.

Rick (1977) has given an algorithm, using this approach, for the determination of the eigenvalues of a specialised type of periodic unsymmetric tridiagonal matrix. For a generalised matrix of the form (7.4.1) having the Sturm sequence of polynomials given in (7.4.3), (7.4.4) and (7.4.5), the derivation of the Bairstow's algorithm becomes complicated and tedious.

We consider here a simpler approach in which we employ a more compact Sturm sequence of polynomials than those given in (7.4.3), (7.4.4) and (7.4.5) in the derivation of Bairstow's algorithm.

It has been shown in section (7.3) that the matrix A in (7.4.1) can, after s P-Q similarity transformations, be reduced to the sparse cyclic lower Hessenberg form,

$$A^{(s)} = \begin{bmatrix} b_1^{(s)} & 1 & & & 0 \\ a_2^{(s)} & b_2^{(s)} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & 1 \\ 1 & & & a_n^{(s)} & b_n^{(s)} \end{bmatrix} \tag{7.4.6}$$

which in a more general form may be written as,

$$H = \begin{bmatrix} \beta_1 & 1 & & & 0 \\ \alpha_2 & \beta_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & 1 \\ 1 & & & \alpha_n & \beta_n \end{bmatrix}. \tag{7.4.7}$$

A Laplacian expansion of $\det(H-\lambda I)$ in terms of the reduced determinants of order (n-1) gives,

$$\det(H-\lambda I)=(\beta_n-\lambda)\det(T'_{n-1}-\lambda I)-\alpha_n\det(T'_{n-2}-\lambda I)+(-1)^{n-1}. \tag{7.4.8}$$

where $T'_i$ is a tridiagonal matrix defined as,

$$T'_i = \begin{bmatrix} \beta_1 & 1 & & & 0 \\ \alpha_2 & \beta_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & 0 & & & 1 \\ & & & \alpha_i & \beta_i \end{bmatrix}_{(i\times i)}. \tag{7.4.9}$$

Hence, if the sequence $\{P_i\}$, i=1,...,n-1 denotes the characteristic polynomials of $(T'_{n-1}-\lambda I)$ then those of $(H-\lambda I)$ are also given by $\{P_i\}$, i=1,...,n, where

$$\left.\begin{aligned} P_{-1}(\lambda) &= 0 , \\ P_0(\lambda) &= 1 , \\ P_1(\lambda) &= (\beta_1-\lambda) , \\ P_2(\lambda) &= (\beta_2-\lambda)P_i(\lambda)-\alpha_2 P_0(\lambda) , \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_i(\lambda) &= (\beta_i-\lambda)P_{i-1}(\lambda)-\alpha_i P_{i-2}(\lambda), \quad i=3,\ldots,n-1, \end{aligned}\right\} \tag{7.4.10a}$$

and finally, for i=n, we obtain from (7.4.8), the expression,

$$P_n(\lambda) = (\beta_n-\lambda)P_{n-1}(\lambda)-\alpha_n P_{n-2}(\lambda) + (-1)^{n-1} . \tag{7.4.10b}$$

These sequences are much simpler than those in (7.4.3)-(7.4.5). Apart from the last term of (7.4.10b), they are the same as those of a tridiagonal matrix.

Algorithmic Derivation of Bairstow's Implicit Synthetic Division Method

If each of the polynomials $P_i(\lambda)$, i=1,2,...,n in (7.4.10) is divided by a trial quadratic factor of the form,

$$\lambda^2-G\lambda-H \tag{7.4.11}$$

then a remainder of the form,

$$C_i\lambda+D_i, \quad i=1,2,\ldots,n \tag{7.4.12}$$

is produced. Thus, the polynomials $P_i(\lambda)$ can now be written in the form,

$$P_i(\lambda) = (\lambda^2-G\lambda-H)Q_i(\lambda)+C_i\lambda+D_i, \quad i=1,\ldots,n \tag{7.4.13}$$

where $Q_i(\lambda)$, i=1,2,...,n, is a polynomial in $\lambda$ of degree (i-2). Next, by substituting (7.4.13) into (7.4.10b) we have, after some simplification

$$\begin{aligned} (\lambda^2-G\lambda-H)Q_i(\lambda)+C_n\lambda+D_n = (\lambda^2-G\lambda-H)\{&(\beta_n-\lambda)Q_{n-1}(\lambda)-\alpha_n Q_{n-2}(\lambda)-C_{n-1}\} \\ &+\lambda\{(\beta_n-G)C_{n-1}-\alpha_n C_{n-2}-D)\} +\beta_n D_{n-1} - \\ &\alpha_n D_{n-2}-HC_{n-1}+(-1)^{n-1}. \end{aligned} \tag{7.4.14}$$

A comparison of the coefficients of $(\lambda^2-G\lambda-H)$, $\lambda^1$ and $\lambda^0$ in both sides of equation (7.4.14) yields,

$$\left.\begin{aligned} Q_n(\lambda) &= (\beta_n-\lambda)Q_{n-1}-\alpha_n Q_{n-2}-C_{n-1} , \\ C_n &= (\beta_n-G)C_{n-1}-\alpha_n C_{n-2}-D_{n-1} \\ \text{and} \qquad B_n &= \beta_n D_{n-1}-\alpha_n D_{n-2}-HC_{n-1}+(-1)^{n-1}. \end{aligned}\right\} \tag{7.4.15}$$

Similarly, by substituting for $P_i(\lambda)$, i=1,...,n-1, from (7.4.13) into (7.4.10a), we obtain similar expressions for $Q_i(\lambda)$, $C_i$ and $D_i$, i=1,2,...,n-1, as,

$$\left.\begin{aligned} Q_i(\lambda) &= (\beta_i-\lambda)Q_{i-1}(\lambda)-\alpha_i Q_{i-2}(\lambda)-C_{i-1}, \\ C_i &= (\beta_i-G)C_{i-1}-\alpha_i C_{i-2}-D_{i-1} \\ \text{and} \quad D_i &= \beta_i D_{i-1}-\alpha_i D_{i-2}-HC_{i-1}. \end{aligned}\right\} \tag{7.4.16}$$

Further, by substituting the starting values, $P_{-1}=0$, $P_0=1$, $P_1=\beta-\lambda$ obtained from (7.4.10a) into equation (7.4.13), we obtain the initial values of the sequence in (7.4.16) as follows:

$$\left.\begin{aligned} Q_0(\lambda) &= 0, \quad C_0 = 0, \quad D_0 = 1, \\ Q_1(\lambda) &= 0, \quad C_1 = -1, \quad D_1 = 1, \\ \text{and} \quad Q_2(\lambda) &= 1, \quad C_2 = \beta_1-\beta_2+G, \quad D_2 = \beta_1\beta_2-\alpha_2+H. \end{aligned}\right\} \tag{7.4.17}$$

For any initial trial values G and H, we can generate the sequence given in (7.4.16) and (7.4.15) by starting with the initial values in (7.4.17) and hence the values of $C_n$,$D_n$ are thus determined. If $C_n$,$D_n$ are zeros, then the quadratic factor $(\lambda^2-G\lambda-H)$ becomes a factor of $P_n(\lambda)$ and so a solution has been found. The basic problem is therefore that of finding G and H such that the following non-linear equation is satisfied, i.e.,

$$C_n(G,H) = D_n(G,H) = 0. \tag{7.4.18}$$

For an arbitrary value of G and H, equation (7.4.18) is not satisfied in general and so correction factors $\Delta G,\Delta H$ must be found such that

$$C_n(G+\Delta G,H+\Delta H) = D_n(G+\Delta G,H+\Delta H) = 0. \tag{7.4.19}$$

This is achieved by dividing $Q_i(\lambda)$, (i=1,...,n-1), in (7.4.16) together with $Q_n$ in (7.4.15) by the same trial quadratic factor in (7.4.11). Hence we have,

$$Q_i(\lambda) = (\lambda^2-G\lambda-H)T_i(\lambda)+L_i\lambda+M_i, \quad i=1,2,...,n \tag{7.4.20}$$

where $L_i\lambda+M_i$ is the linear remainder and $T_i(\lambda)$ is a polynomial of order 2 less than $Q_i(\lambda)$.

Substituting for $Q_i(\lambda)$, (i=1,2,...,n), from (7.4.20) into (7.4.16)

and (7.4.15) we have, after some simplifications,

$$Q_i(\lambda) = (\lambda^2-G\lambda-H)\{(\beta_i-\lambda)T_{i-1}-\alpha_i T_{i-2}-L_{i-1}\}+\lambda\{(\beta_i-G)L_{i-1} -M_{i-1}-\alpha_i L_{i-2}\}+\{\beta_i M_{i-1}-\alpha_i M_{i-2}-C_{i-1}-HL_{i-1}\}. \tag{7.4.21}$$

By comparing the right hand sides of (7.4.20) and (7.4.21) and equating coefficients of $(\lambda^2-G\lambda-H)$, $\lambda^1$ and $\lambda^0$ we obtain,

$$\left.\begin{aligned} T_i(\lambda) &= (\beta_i-\lambda)T_{i-1}-\alpha_i T_{i-2}-L_{i-1}\ , \\ L_i &= (\beta_i-G)L_{i-1}-M_{i-1}-\alpha_i L_{i-2}\ , \\ M_i &= \beta_i M_{i-1}-\alpha_i M_{i-2}-C_{i-1}-HL_{i-1}\ , \end{aligned}\right\} \quad i=1,\ldots,n. \tag{7.4.22}$$

Using the initial values given by (7.4.17) in equation (7.4.20), we obtain the initial values for the sequence given in (7.4.22), as,

$$\left.\begin{aligned} T_0(\lambda) &= 0\ , & L_0 &= 0\ , & M_0 &= 0\ , \\ T_1(\lambda) &= 0\ , & L_1 &= 0\ , & M_1 &= 0\ , \\ T_2(\lambda) &= 0\ , & L_2 &= 0\ , & M_2 &= 1\ , \\ T_3(\lambda) &= 0\ , & L_3 &= -1, & M_3 &= \beta_3-C_2 \end{aligned}\right\} \tag{7.4.23}$$

Thus, starting with (7.4.23), the sequence in (7.4.22) can be computed and thus $L_n, M_n$ obtained.

Finally, the correction factors $\Delta G$ and $\Delta H$ to the quadratic factor $(\lambda^2-G\lambda-H)$ are then given by the formulae (Wilkinson (1965)),

$$\begin{aligned} \theta &= HL_n + GM_n\ , \\ \phi &= M_n^2 - \theta L_n\ , \\ \Delta G &= [L_n(D_n+GC_n)-(M_n+GL_n)C_n]/\phi \\ \text{and} \qquad \Delta H &= [\theta C_n-(M_n+GL_n)(D_n+GC_n)]/\phi. \end{aligned} \tag{7.4.24}$$

New values of G and H are now calculated as,

$$G = G + \Delta G \qquad \text{and} \qquad H = H + \Delta H \tag{7.4.25}$$

which are then used to form a new quadratic factor.

The process is repeated until $\Delta G, \Delta H$ are small enough with respect to a given specified error tolerance. At this stage, the quadratic equation $\lambda^2-G\lambda-H=0$ is solved to yield the required pair of eigenvalues.

The sequence of polynomials $Q_i(\lambda)$, i=1,2,...,n, of maximum degree (n-2)

defined in (7.4.16) and (7.4.15) can now be used to determine further pairs of eigenvalues by a repetition of the above process. As the degree of the original polynomial n is effectively reduced by two at each stage, the recursive sequences are shorter and further eigenvalues are progressively easier and faster to calculate.

The convergence of this algorithm is quadratic when close to a pair of eigenvalues (Wilkinson, (1965)). However, in practice the algorithm takes a number of iterations to 'search' for the pair of eigenvalues before it finally settles down on a pair and then converges rapidly. This 'search' to identify a pair of eigenvalues may be explained by the fact that even if close estimates to a certain pair of eigenvalues are chosen the resulting quadratic factors need not be close to the actual one. For example, if a pair of eigenvalues are .01 and 1000, then the quadratic factor becomes $\lambda^2$-1000.01+10. If estimates of the eigenvalues were taken as -.01 and 1000, then the quadratic factor is $\lambda^2$-999.99$\lambda$-10 and hence may converge to a different pair of eigenvalues. For an $n^{th}$ order system, the eigenvalues can be combined to produce ${}^{n}C_{2}$(=n(n-1)/2) possible quadratic factors and the fact that there are a large number of factors to choose from serves to slow down the convergence process. A speed-up of the algorithm can be improved by choosing sufficiently close initial guesses to the pair of eigenvalues, instead of just any arbitrary initial guesses. A numerical experiment was carried out to investigate the convergence behaviour of the algorithm based upon the choice of the initial guess. An ad-hoc method which was found to give the best convergence results (an average of 8 to 10 iterations for each pair of eigenvalues) is as follows:

For the first pair of eigenvalues, initial values chosen from the diagonal elements of the unsymmetric matrix H defined in (7.4.7) was used as the initial guess. Thereafter, the most recently computed pair of eigenvalues were adopted as further initial guesses to the next pair to be

sought. This scheme is particularly suitable for matrices with ordered eigenvalues.

Implementation of Bairstow's Algorithm

The two main advantages of the Bairstow's algorithm are:-

(1) It finds complex eigenvalues without having to work in complex arithmetic which is time consuming on a computer.

(2) It requires minimal computer storage for its implementation because all the correction factors and remainders can be calculated without explicitly calculating and storing the new polynomials $Q_i(\lambda)$, $T_i(\lambda)$, i=1,2,...,n produced at each stage of the implicit synthetic division.

To accomplish (2), in programming terms, we need to carry out the synthetic division one step further. Hence, if we consider the polynomials $T_i(\lambda)$ in (7.4.22) and divide these by the quadratic factor $\lambda^2-G\lambda-H$, we have,

$$T_i(\lambda) = (\lambda^2-G\lambda-H)Y_i + \lambda E_i + F_i. \qquad (7.4.26)$$

By substituting (7.4.26) for $T_i(\lambda)$ in (7.4.22) we obtain, after some simplification, the recurrence relation,

$$\left.\begin{aligned} Y_i &= (\beta_i-\lambda)Y_{i-1}-\alpha_i Y_{i-2}-E_{i-1}\,, \\ E_i &= (\beta_i-G)E_{i-1}-F_{i-1}-\alpha_i E_{i-2} \\ \text{and}\quad F_i &= \beta_i F_{i-1}-HE_{i-1}-\alpha_i F_{i-2}-L_{i-1} \end{aligned}\right\} \quad i=1,\ldots,n. \qquad (7.4.27)$$

By using (7.4.26) and the initial values of $T_i(\lambda)$ given in (7.4.23), we obtain the initial values of the sequence in (7.2.27) as,

$$\left.\begin{aligned} Y_0 &= 0, \quad E_0 = 0, \quad F_0 = 0\,, \\ Y_1 &= 0, \quad E_1 = 0, \quad F_1 = 0\,. \end{aligned}\right\} \qquad (7.4.28)$$

By the above approach, it is possible to perform the synthetic division in the same manner at each stage because after finding one pair of eigenvalues the process is continued exactly in the same manner by overwriting

$C_i$ with $L_i$ ,
$D_i$ with $M_i$ ,
$L_i$ with $E_i$ ,
and $M_i$ with $F_i$ ;

and hence this eliminates the need to store explicitly the polynomials $Q_i(\lambda)$, $T_i(\lambda)$, i=1,...,n, produced at each stage of the process. The storage requirement of the Bairstow's method is therefore kept to a minimum.

Numerical Results

The algorithm which implements the Bairstow's method as described in this section is given as Program 20 in Appendix I. A number of tests were used to demonstrate the performance of this algorithm. These test matrices were first reduced to the sparse cyclic lower Hessenberg form as in (7.4.7) by using Program 19 of Appendix I.

We first consider the (10×10) matrix A satisfying condition (2.4.20) and given by,

$$A = \begin{bmatrix} 4 & 1 & & & & & & & & 1 \\ 2 & 6 & 1 & & & & & & & \\ & 1 & 8 & 1 & & & & & & \\ & & 2 & 10 & 1 & & & & & \\ & & & 1 & 12 & 1 & & & & \\ & & & & 2 & 14 & 1 & & & \\ & & & & & 1 & 16 & 1 & & \\ & & & & & & 2 & 18 & 1 & \\ & & & 0 & & & & 1 & 20 & 1 \\ 1 & & & & & & & & 2 & 22 \end{bmatrix}_{(10\times10)} \qquad (7.4.29)$$

The experimental results obtained are compared with those given by the N.A.G. (F02AFF) subroutine as shown in Table (7.3). Both results agree to within 8 significant figures; and while these cannot be conclusive, they do indicate that an accuracy of at least 8 significant figures has been obtained by the Bairstow's scheme.

The Bairstow's algorithm took 6 mill units of time to calculate all the eigenvalues of the matrix (7.4.29) whilst the N.A.G. (F02AFF) subroutine took only 3 mill units.

| $\lambda_i$ | Bairstow's Method (Program 20) | | N.A.G. (FO2AFF) Subroutine | |
|---|---|---|---|---|
| | Real | Imaginary | Real | Imaginary |
| 1 | 3.1776805601 | 0.0000000000 | 3.1776805033 | 0.0000000000 |
| 2 | 6.1439159399 | 0.0000000000 | 6.1439159783 | 0.0000000000 |
| 3 | 7.7775316071 | 0.0000000000 | 7.7775316033 | 0.0000000000 |
| 4 | 10.2121100623 | 0.0000000000 | 10.2121100640 | 0.0000000000 |
| 5 | 11.7877109588 | 0.0000000000 | 11.7877107845 | 0.0000000000 |
| 6 | 14.2122892338 | 0.0000000000 | 14.2122892159 | 0.0000000000 |
| 7 | 15.7878895183 | 0.0000000000 | 15.7878899379 | 0.0000000000 |
| 8 | 18.2224682996 | 0.0000000000 | 18.2224683966 | 0.0000000000 |
| 9 | 19.8560841913 | 0.0000000000 | 19.8560840210 | 0.0000000000 |
| 10 | 22.8223194761 | 0.0000000000 | 22.8223194999 | 0.0000000000 |

TABLE 7.3

Further, we consider the (15×15) unsymmetric periodic tridiagonal matrix A possessing complex eigenvalues, i.e.,

$$A=\begin{bmatrix} 2 & 1 & & & & & & & & & & & & & 0.5 \\ 1 & 3 & 1 & & & & & & & & & & & & \\ & 2 & 4 & 1 & & & & & & & & & & & \\ & & 3 & 5 & 2 & & & & & & & & & & \\ & & & 2 & 6 & 2 & & & & & & & & & \\ & & & & 3 & 7 & 3 & & & & & & & & \\ & & & & & 2 & 8 & 2 & & & & & & & \\ & & & & & & 3 & 9 & 3 & & & & & & \\ & & & & & & & 2 & 10 & 3 & & & & & \\ & & & & & & & & 3 & 11 & 3 & & & & \\ & & & & & & & & & 2 & 12 & 3 & & & \\ & & & & 0 & & & & & & 3 & 13 & 2 & & \\ & & & & & & & & & & & 2 & 14 & 3 & \\ & & & & & & & & & & & & 3 & 15 & 2 \\ 10 & & & & & & & & & & & & & 2 & 16 \end{bmatrix}_{(15\times15)} \qquad (7.2.30)$$

The results for this matrix are shown in Table (7.4). Here, the N.A.G. routine took 10 mill-units while the Bairstow's method took 16 mill-units.

To illustrate further the speed performance of the Bairstow's algorithm as compared to the N.A.G. routine, we perform further tests on general unsymmetric periodic tridiagonal matrices of various order. The execution times taken by the Bairstow's method were compared to corresponding results for the N.A.G. (FO2AFF) subroutine for various matrix sizes as shown in Table (7.5).

As is evident from Table (7.5), the N.A.G. (FO2AFF) subroutine, which employs the Francis QR transformation technique, is significantly faster than the Bairstow's method for small order matrices. However, with an increasing size of the matrix system, the Bairstow's scheme improves in its speed competitiveness, essentially due to the sparsity-characteristics of the latter method which the former does not possess. Moreover, the storage for the Bairstow's algorithm is O(N) words while that of the N.A.G. routine is $O(N^2)$ words. Thus, when the N.A.G. routine runs into storage difficulties on large order matrix systems the Bairstow's method becomes a preferred method to use.

| $\lambda_i$ | Program (20) | | N.A.G. (F02AFF) Subroutine | |
|---|---|---|---|---|
| | Real | Imaginary | Real | Imaginary |
| 1 | 0.8805879241 | 0.0000000000 | 0.8805879621 | 0.0000000000 |
| 2 | 1.6481552756 | 0.0000000000 | 1.648552346 | 0.0000000000 |
| 3 | 3.0900980496 | 0.8522928632 | 3.0900980573 | 0.8522928323 |
| 4 | 3.0900980496 | -0.8522928632 | 3.0900980573 | -0.8522928323 |
| 5 | 5.3674818721 | 1.7142297883 | 5.3674818778 | 1.7142297990 |
| 6 | 5.3674818721 | -1.7142297883 | 5.3674818778 | -1.7142997990 |
| 7 | 8.0308615719 | 2.0510701660 | 8.0308615756 | 2.0510701730 |
| 8 | 8.0308615719 | -2.0510701660 | 8.0308615756 | -2.0510701730 |
| 9 | 10.7834776671 | 1.7075564422 | 10.7834776698 | 1.7705564381 |
| 10 | 10.7834776671 | -1.7075564422 | 10.7834776698 | -1.7705564381 |
| 11 | 13.4020080815 | 0.6358051266 | 13.4020080871 | 0.6358051201 |
| 12 | 13.4020080815 | -0.6358051266 | 13.4020080871 | -0.6358051201 |
| 13 | 15.5435843772 | 0.0000000000 | 15.5435843822 | 0.0000000000 |
| 14 | 16.7258676280 | 0.0000000000 | 16.7258676400 | 0.0000000000 |
| 15 | 18.8539502835 | 0.0000000000 | 18.8539502432 | 0.0000000000 |

TABLE 7.4

Times Taken to Obtain All the Eigenvalues of an Unsymmetric Tridiagonal Matrix of Order n

(Unit: Mill-secs)

| Order of Matrix n | Times taken by Bairstow's Method (Program 20): $T_1$ | Times taken by N.A.G. F02AFF Subroutine: $T_2$ | Ratio of Times $T_1/T_2$ |
|---|---|---|---|
| 10 | 6 | 3 | 2.0 |
| 15 | 16 | 10 | 1.6 |
| 20 | 25 | 19 | 1.3 |
| 30 | 68 | 59 | 1.2 |
| 35 | 90 | 88 | 1.02 |
| 40 | 126 | 128 | 0.98 |

TABLE 7.5

## 7.5 THE DETERMINATION OF THE EIGENVALUES OF A SYMMETRIC PERIODIC TRIDIAGONAL MATRIX BY NEWTON'S METHOD

Let A be an $(n\times n)$ symmetric periodic tridiagonal matrix of the form given by (7.1.8) which is derived from the finite difference discretisation of a Sturm-Liouville problem (7.1.2).

We assume that A is diagonally dominant and that the diagonal elements satisfy the condition (7.3.3). Condition (2.4.20) is also assumed.

It was shown in section (7.3) that such a matrix is reducible by a P-Q similarity transformation to a sparse cyclic lower Hessenberg form, H, given by (7.4.7).

The sequence of polynomials which determines the principal minors of $\det(H-\lambda I)$ were given in (7.4.10). Differentiating these polynomials in (7.4.10) with respect to $\lambda$, we have,

$$\left.\begin{aligned}
&\frac{d}{d\lambda}(P_0(\lambda)) = P_0' = 0 ,\\
&\frac{d}{d\lambda}(P_1(\lambda)) = P_1' = -1 ,\\
&\frac{d}{d\lambda}(P_2(\lambda)) = (\beta_2-\lambda)P_1'-P_1-\alpha_2P_0' ,\\
&\frac{d}{d\lambda}(P_i(\lambda)) = (\beta_i-\lambda)P_{i-1}'-P_{i-1}-\alpha_iP_{i-2}', \quad i=3,\ldots,n-1,\\
\text{and}\quad &\frac{d}{d\lambda}(P_n(\lambda)) = (\beta_n-\lambda)P_{n-1}'-P_{n-1}-\alpha_nP_{n-2}' .
\end{aligned}\right\} \qquad (7.5.1)$$

The recurrence formulae in (7.4.10) and their corresponding derivatives in (7.5.1) can now be applied in the determination of the eigenvalues $\lambda$ of H by using the Newton's iterative scheme,

$$\lambda^{(k+1)} = \lambda^{(k)} - [P_n(\lambda^{(k)})/P_n'(\lambda^{(k)})], \quad P_n'(\lambda)\neq 0, \ k>0, \qquad (7.5.2)$$

where $\lambda^{(0)}$ is an initial estimate of $\lambda$.

The method (7.5.2) has quadratic convergence for simple roots (Froberg (1965)), i.e.,

$$|\lambda^{(k+1)} - \lambda| \approx c|\lambda^{(k)} - \lambda|^2 , \quad 0<c<1 \qquad (7.5.3)$$

but the convergence becomes linear if the roots to which the method is converging is multiple. Convergence is attained when

$$|(\lambda^{(k+1)} - \lambda^{(k)})/\lambda^{(k)}| < \varepsilon; \quad |P_n(\lambda^{(k+1)}| < \varepsilon \tag{7.5.4}$$

for some small specified tolerance.

When the first eigenvalue has been found by the Newton's scheme (7.5.2), an alternative scheme, the Secant method, given by,

$$\lambda^{(k+1)} = \lambda^{(k)} - [(\lambda^{(k)} - \lambda^{(k-1)})P_n(\lambda^{(k)})]/[P_n(\lambda^{(k)}) - P_n(\lambda^{(k-1)})], \quad k \geq 1 \tag{7.5.5}$$

can be used in determining subsequent eigenvalues.

When one or more of the eigenvalues has been computed, then in order to avoid redetermining those eigenvalues already found the technique of 'dividing out the roots' is used to suppress the known eigenvalues. Thus, instead of iterating with $P_n(\lambda)$, we use $G_n(\lambda)$ where

$$G_n(\lambda) = P_n(\lambda) / \prod_{i=1}^{s} (\lambda - \lambda_i) \tag{7.5.6}$$

and $\lambda_i$, i=1,...,s are the s eigenvalues already found.

The derivative of $G_n(\lambda)$ with respect to $\lambda$, for use in the Newton's scheme, is then given by

$$\frac{d}{d\lambda}(G_n(\lambda)) = G_n'(\lambda) = G_n(\lambda)\{P_n'(\lambda)/P_n(\lambda) - \sum_{i=1}^{s} (\lambda - \lambda_i)^{-1}\} \quad . \tag{7.5.7}$$

Hence, the Newton's formula (7.5.2) can now be replaced by the form,

$$\lambda^{(k+1)} = \lambda^{(k)} - [G_n(\lambda^{(k)})/G_n'(\lambda^{(k)})] \tag{7.5.8}$$

which, on substituting for $G_n(\lambda^{(k)}/G_n'(\lambda^{(k)})$ from (7.5.7), yields,

$$\lambda^{(k+1)} = \lambda^{(k)} - 1/\{P_n'(\lambda^{(k)})/P_n(\lambda^{(k)}) - \sum_{i=1}^{s} (\lambda^{(k)} - \lambda_i)^{-1}\} \tag{7.5.9}$$

from which we can compute the eigenvalues.

For large order matrices of order greater than 30, the Sturm sequence of polynomials $P_i(\lambda)$, i=1,2,...,n, in (7.4.10a) and $P_i'(\lambda)$ in (7.5.1) can oscillate widely, giving rise to overflow or underflow, particularly for estimates of $\lambda$ far from the actual values.

To overcome this problem of overflow, we adopt the procedure of Barth et al (1967) and replace the sequence of polynomials $P_i(\lambda)$, i=1,2,...,n in (7.4.10a) by a new sequence of scaled polynomials,

$$p_0(\lambda) = 1; \quad p_i(\lambda) = P_i(\lambda))/P_{i-1}(\lambda) \ , \ i=1,2,...,n. \tag{7.5.10}$$

Thus, we have,

$$\left.\begin{aligned} p_0(\lambda) &= 1 , \\ p_1(\lambda) &= (\beta_1-\lambda) , \\ p_i(\lambda) &= (\beta_i-\lambda)-\alpha_i/p_{i-1}(\lambda), \quad i=2,\ldots,n-1 \end{aligned}\right\} \tag{7.5.11}$$

and

$$p_n(\lambda) = (\beta_n-\lambda)-\alpha_{n-1}/p_{n-1}(\lambda)+(-1)^{n-1}/p_{n-1}(\lambda). \tag{7.5.12}$$

By using the relationships in (7.5.10), it follows immediately that

$$P_i(\lambda) = \prod_{j=1}^{i} p_j , \qquad i=1,2,\ldots,n \tag{7.5.13}$$

and hence (7.5.12) becomes,

$$p_n(\lambda) = (\beta_i-\lambda)-\alpha_{n-1}/p_{n-1}+(-1)^{n-1} \Big/ \prod_{j=1}^{n-1} p_j . \tag{7.5.14}$$

Next, by differentiating the sequence of polynomials $p_i(\lambda)$ in (7.5.11) with respect to $\lambda$, we obtain the following sequence,

$$\left.\begin{aligned} \frac{d}{d\lambda}(p_0(\lambda)) &= p_0'(\lambda) = 0 , \\ \frac{d}{d\lambda}(p_1(\lambda)) &= p_1'(\lambda) = -1 , \\ \frac{d}{d\lambda}(p_i(\lambda)) &= p_i'(\lambda) = -1+\alpha_i p_{i-1}^1/p_{i-1}^2(\lambda), \quad i=2\ldots,n-1, \end{aligned}\right\} \tag{7.5.15}$$

and finally, from (7.5.12) we have,

$$\frac{d}{d\lambda}(p_n(\lambda)) = p_n'(\lambda) = -1+\alpha_{n-1}p_{n-1}'(\lambda)/p_{n-1}^2(\lambda)+(-1)^{n-1} \frac{d}{d\lambda}[p_{n-1}^{-1}(\lambda)] \tag{7.5.16}$$

where,

$$\frac{d}{d\lambda}[p_{n-1}^{-1}(\lambda)] = -P_{n-1}'(\lambda)/P_{n-1}^2(\lambda) . \tag{7.5.17}$$

Further, a differentiation of (7.5.13) yields,

$$P_i'(\lambda) = \sum_{k=1}^{i} \prod_{\substack{j=1 \\ j\neq k}}^{i} p_j(\lambda)p_k'(\lambda) \tag{7.5.18}$$

and hence, using (7.5.13) and (7.5.18), we obtain

$$\frac{P_i'(\lambda)}{P_i(\lambda)} = \frac{\sum_{k=1}^{i} \prod_{\substack{j=1 \\ j\neq k}}^{i} p_j(\lambda)p_k'(\lambda)}{\prod_{j=1}^{i} p_j} = \sum_{k=1}^{i} p_k'(\lambda)/p_k(\lambda). \tag{7.5.19}$$

On substituting (7.5.19) into (7.5.17) and using (7.5.13), we have,

$$\frac{d}{d\lambda}[P_{n-1}^{-1}(\lambda)] = -\frac{P'_{n-1}(\lambda)}{P_{n-1}(\lambda)} \cdot \frac{1}{P_{n-1}(\lambda)} = -\sum_{k=1}^{n-1} \{p'_k(\lambda)/p_k(\lambda)\} / \prod_{k=1}^{n-1} p_k \ . \tag{7.5.20}$$

Equation (7.5.16) can now be written in the form,

$$\frac{d}{d\lambda}(p_n(\lambda)) = p'_n(\lambda) = -1+\alpha_{n-1}p'_{n-1}/p^2_{n-1}-(-1)^{n-1}[\sum_{k=1}^{n-1} p'_k(\lambda)/p_k(\lambda)]/\prod_{k=1}^{n-1} p_k . \tag{7.5.21}$$

Finally, the Newton's iterative scheme (7.5.9) can now be replaced, on substituting for $P'_n(\lambda)/P_n(\lambda)$ from equation (7.5.19), by the new equation,

$$\lambda^{(k+1)} = \lambda^{(k)} - 1/ \{ \sum_{i=1}^{n} (p'_i(\lambda)/p'_i(\lambda)) - \sum_{j=1}^{s} (\lambda^{(k)}-\lambda_j)^{-1} \} \ . \tag{7.5.22}$$

The use of the form in (7.5.22) instead of that in (7.5.9) enables the eigenvalues of a sparse cyclic lower Hessenberg matrix of any order to be found without the risk of either overflow or underflow occurring.

Results

Using the formula (7.5.22), the above method was programmed in Fortran on the I.C.L. 1904S computer in single precision arithmetic. A test run was performed on the (16×16) matrix,

$$A = \begin{bmatrix} 3 & 2 & & & & & & & & & & & & & & 1 \\ 2 & 4 & 3 & & & & & & & & & & & & & \\ & 3 & 5 & 2 & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ 1 & & & & & & & & & & & & & & & \end{bmatrix} \tag{7.5.23}$$

(16×16)

which was first reduced to a sparse cyclic lower Hessenberg form before applying the Newton's scheme as given above. The eigenvalues obtained are compared with those of the N.A.G. (F02AAF) routine (which employs the Householder reduction and the QL algorithmic methods). The results are shown in Table (7.6). The N.A.G. routine obtained all the eigenvalues in 11 mill-units of time whilst the Newton's method took 23 mill-units;

which indicates that the N.A.G. routine, while not exactly tailored to fit the particular sparse periodic matrix system in question, is nevertheless much faster than the Newton's scheme. This is a usual feature of transformation methods compared to Sturm sequence methods when storage is not crucial. However, as the storage for the Newton's method is O(n) words whilst that for the N.A.G. routine is $O(n^2)$ words, then the Newton's method becomes preferable when the order of the matrix is large, and also when all the eigenvalues are required; and none, or at least very few, multiple roots are expected.

Eigenvalues of the Symmetric Periodic Tridiagonal Matrix

| $\lambda_i$ | Newton's scheme (7.5.22) | N.A.G. (FO2AAF) subroutine |
|---|---|---|
| 1 | 1.54457116147 | 1.5445711485 |
| 2 | 3.32510102720 | 3.3251010138 |
| 3 | 4.60150483656 | 4.6015048146 |
| 4 | 6.01368761062 | 6.0136874602 |
| 5 | 6.91862358566 | 6.9186233199 |
| 6 | 8.06311292629 | 8.0631129837 |
| 7 | 8.93546534220 | 8.9354653781 |
| 8 | 10.06476778472 | 10.0647677400 |
| 9 | 10.93523225737 | 10.9352322582 |
| 10 | 12.06453467013 | 12.0645346217 |
| 11 | 12.93688788849 | 12.9368870165 |
| 12 | 14.08137643354 | 14.0813766816 |
| 13 | 14.98631253814 | 14.9863125393 |
| 14 | 16.39849514463 | 16.3984951852 |
| 15 | 17.67489895388 | 17.6748989862 |
| 16 | 19.45542884324 | 19.4554288473 |

TABLE 7.6

## 7.6 METHODS FOR THE DETERMINATION OF EIGENVECTORS OF SPARSE MATRICES

### Inverse Iteration

The inverse iteration (also called the inverse Wieldandt method), a variant of the power method, is used for the determination of a particular eigenvector corresponding to a known approximate eigenvalue $\rho$ of a given matrix A and may be defined by the iterative scheme,

$$\left.\begin{aligned} (A-\rho I)\underline{y}^{(i+1)} &= \underline{z}^{i} \quad , \\ \underline{z}^{(i+1)} &= \underline{y}^{(i+1)}/||\underline{y}^{(i+1)}||_2 \end{aligned}\right\} \; i=0,1,2,\ldots \qquad (7.6.1)$$

where the vector $\underline{z}$ converges to the eigenvector corresponding to the eigenvalue of A nearest to $\rho$ (Wilkinson (1965)).

The scheme (7.6.1) is equivalent to a successive solution of the same matrix equation but with different right hand sides. A considerable economy is therefore achieved by constructing the LU decomposition of $(A-\rho I)$ which needs to be performed once only. The matrix factors L and U may be determined by Gaussian elimination but in order to ensure numerical stability, a strategy involving pivoting techniques with interchanges are essential.

Ignoring these interchanges for clarity of presentation, the matrix equation in (7.6.1) can be written in the form,

$$\left.\begin{aligned} L\underline{v} &= \underline{z}^{(i)} \\ U\underline{y}^{(i+1)} &= \underline{v} \end{aligned}\right\} \quad i=0,1,\ldots \qquad (7.6.2)$$

where,

$$LU = (A-\rho I). \qquad (7.6.3)$$

It is well-known (Wilkinson (1965)) that by choosing the initial vector,

$$\underline{z}^{(0)} = L\underline{e} \; , \qquad (7.6.4)$$

then the first iterate of (7.6.2) is given simply by the relation,

$$U\underline{y}^{(1)} = \underline{e} \qquad (7.6.5)$$

where $\underline{e}^T=(1,1,\ldots,1)$ and hence there is no need to determine $L\underline{e}$ explicitly. Once $\underline{y}^{(1)}$ is computed, we can then obtain subsequent iterated vectors $\underline{y}^{(i)}$ by a forward and back substitution process using (7.6.2).

Convergence of the inverse iteration method is very rapid and occurs after only one or two iterations; thus making the method undoubtedly the most powerful technique for finding an eigenvector associated with any eigenvalue in the spectrum which has been determined by a root-finding or other procedures.

However, in certain applications (e.g. the determination of the natural frequency and modes of vibration of harmonic or biharmonic operators in a rectangular region) the use of the five, nine or thirteen point finite difference approximations often leads to the matrix eigenvalue problem for matrices with large sparse banded structure. In applying the inverse iteration for the computation of the eigenvectors of these matrices, the pivoting technique (which must be included for stability reasons) often leads to a complicated algorithm; the complexity of programming the method to take advantage of the sparsity of the matrix often outweighs any gains that would otherwise be made. Furthermore, in such applications involving large symmetric sparse matrices, it is often the lowest and/or largest eigenvalues and their corresponding eigenvectors that are of most practical importance. For such eigenvalue problems, iterative methods which preserve the sparseness of the matrix are particularly suitable (Shavitt et al (1973)).

## An Iterative Calculation of the Lowest and Highest Eigenvalues and the Corresponding Eigenvectors for Large Sparse Matrices

We consider the eigenvalue problem,

$$A\underline{u} = \lambda\underline{u} \tag{7.6.6}$$

and assume that the matrix A is an $(n\times n)$ large order diagonally dominant symmetric sparse matrix with n real eigenvalues, $\lambda_1,\lambda_2,\ldots,\lambda_n$, ordered such that, $\lambda_1 \leq \lambda_2 ,\ldots,\leq \lambda_n$ and that their corresponding orthonormalised eigenvectors are $\underline{u}_1,\underline{u}_2,\ldots,\underline{u}_n$. Further, we assume that A can be decomposed into the form,

$$A = D-L-U \tag{7.6.7}$$

where D=(d,d,...,d) is a constant term diagonal matrix, and L and U are in general strictly lower and upper triangular matrices respectively. Thus, (7.6.6) becomes,

$$(D-L-U)\underline{u} = \lambda\underline{u}$$

or
$$(I-\omega L-\omega U)\underline{u} = 0 \qquad (7.6.8)$$

where
$$\omega = 1/(d-\lambda)\,. \qquad (7.6.9)$$

Equation (7.6.9) can be rewritten in a more easily factorisable form as,

$$(I-\omega L)(I-\omega U)\underline{u} = \omega^2 LU\underline{u}\,. \qquad (7.6.10)$$

Using (7.6.10) we consider the sequence of normalised vectors $\{\underline{x}^{(s)}\}$ obtained from the iterative scheme,

$$(I-\omega^{(\ell)}L)(I-\omega^{(\ell)}U)\underline{x}^{(s+1)} = \omega^{(\ell)^2}LU\underline{x}^{(s)}\,, \quad s=0,1,... \qquad (7.6.11)$$

where the relaxation parameter $\omega^{(\ell)}$ is given by,

$$\omega^{(\ell)} = 1/(d-\mu^{(\ell)}), \qquad \mu^{(\ell)} \neq d \qquad (7.6.12)$$

and
$$\mu^{(\ell)} = \underline{x}^{(\ell)^T}A\underline{x}^{(\ell)}, \qquad \underline{x}^{(\ell)} \neq 0, \ \ell=5,10...(\text{say}) \qquad (7.6.13)$$

is the Rayleigh quotient, i.e., the eigenvalues of A closest to the normalised eigenvector $\underline{x}^{(\ell)}$. The updating of the Rayleigh quotient is arranged to take place after every 5 (say) outer iterations of (7.6.11).

The lower and upper bounds of the Gerschgorin's circle theorem (2.5) applied to the matrix A offer good initial estimates, $\mu^{(0)}$, for the smallest and largest eigenvalues respectively.

The theoretical analysis of the convergence behaviour of the iterative scheme (7.6.11) has not been carried out and further investigation is required in this direction. However, from various numerical experiments conducted with tridiagonal, periodic tridiagonal and sparse quindiagonal eigenvalue systems, the following experimental results were obtained:-

Case 1

For a suitable starting eigenvalue $\mu^{(0)}$ such that

$$\mu^{(0)} < d$$

(i.e., the relaxation parameter $\omega^{(0)}$ is positive), together with a

reasonably good starting vector $\underline{x}^{(0)}$, the sequence $\{\mu^{(\ell)}\}$ converges to the smallest eigenvalue $\lambda_1$ and the sequence of vectors $\{\underline{x}^{(s)}\}$ converges to the corresponding eigenvector $\underline{u}_1$.

Case 2

Similarly, for a starting eigenvalue $\mu^{(0)}$ such that

$$\mu^{(0)} > d$$

(i.e., the parameter $\omega^{(0)}$ is negative) together with a suitable starting vector $\underline{x}^{(0)}$, the sequence $\{\mu^{(\ell)}\}$ converges to the largest eigenvalue $\lambda_n$; and $\{\underline{x}^{(s)}\}$ to the corresponding eigenvector, $\underline{u}_n$.

For values of $\mu^{(0)}$ very close to d, the magnitude of the relaxation factor becomes very large; the convergence behaviour of the sequence $\{\underline{x}^{(s)}\}$ is inconclusive and requires further investigation.

Experimental Result

During the investigation of the proposed iterative method, we considered, as an example, the determination of the smallest and largest eigenvalues and their corresponding eigenvectors of the system,

$$B\underline{u} = \lambda\underline{u} \tag{7.6.14}$$

where B is a symmetric sparse quindiagonal (block tridiagonal) matrix of order (nm×nm) given by,

$$B = \begin{bmatrix} D & -I & & & \\ -I & D & -I & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & & -I \\ & 0 & & -I & D \end{bmatrix}_{(m\times m)} \tag{7.6.15}$$

where D is the (n×n) matrix,

$$D = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & & -1 \\ & 0 & & -1 & 4 \end{bmatrix}. \tag{7.6.16}$$

and I is the (n×n) identity matrix.

The eigenvalue problem (7.6.14) can be derived, for example, by applying the five point finite difference approximation over a rectangular region in the x-y plane to the second-order Helmholtz equation,

$$-\left(\frac{\partial^2 U(x,y)}{\partial x^2} + \frac{\partial^2 U(x,y)}{\partial y^2}\right) = \lambda U(x,y) \tag{7.6.17}$$

with the boundary conditions,

$$\left.\begin{aligned} U(x,y) &= 0; \quad x = -1,1 \; ; \quad -\tfrac{1}{2} \leq y \leq \tfrac{1}{2} \\ U(x,y) &= 0; \quad y = -\tfrac{1}{2}, \tfrac{1}{2} \; ; \quad -1 \leq x \leq 1. \end{aligned}\right\} \tag{7.6.18}$$

Such an eigenvalue problem represents, for example, that of finding the modes of propagation of the transverse field vector component of an electromagnetic wave in a long conducting cylinder of rectangular cross-section.

The iterative scheme (7.6.11) together with (7.6.12) and (7.6.13) was programmed in single precision in order to determine the smallest and largest eigensystem of a matrix of the form (7.6.15). Various numerical tests of the iterative algorithm were performed for different sizes of the coefficient matrix. It was found that on the average, the number of iterative steps required to obtain convergence to the smallest or largest eigensystem (to an accuracy of $10^{-6}$ in the components of the normalised eigenvectors) is about 15. Even though this is much greater than the 1 or 2 steps required by the inverse iteration scheme, there are, however, computational advantages of the former method. For the block tridiagonal matrix (7.6.15), for example, of order (nm×nm), only O(nm) storage locations are required to generate the iterative scheme (7.6.11). On the other hand, for the inverse iteration method, the coefficient matrix has to be stored requiring $O((nm)^2)$ storage locations; and since the LU form of decomposition involved in this method destroys the sparseness of the coefficient matrix as a result of the need to include interchanges, (which cause a fill-up of the bands) more complicated storage management is required in order to take full advantage of the sparsity of the given

coefficient matrix. Furthermore, for the inverse iteration process the bands of the L and U factors of the decomposed iteration matrix are usually filled up; thus increasing the computing effort of the backward and forward substitution steps of the inverse iteration process.

The smallest and largest eigenvalues and their corresponding eigenvectors of a test matrix (7.6.15) of order 25 (i.e. n=5, m=5) were computed using the proposed iterative scheme; and these were compared with results obtained from the N.A.G. library (FO2ABF) subroutine as shown in Table (7.6).

The Smallest and Largest Eigenvalues and Vectors of a Block Tridiagonal Matrix of Order $(5^2 \times 5^2)$

| | Iterative Scheme (7.6.11) | N.A.G. (FO2AFB) Routine | Iterative Scheme (7.6.11) | N.A.G. (FO2ABF Routine |
|---|---|---|---|---|
| Eigenvalues | $\lambda_1$=0.535898387 | $\lambda_1$=0.535898385 | $\lambda_n$=7.46410161 | $\lambda_n$=7.46410162 |
| Eigenvectors | 0.083333700 | 0.083333333 | -0.083333700 | -0.083333333 |
| | 0.144333664 | 0.144333756 | 0.144333663 | 0.144333756 |
| | 0.166667245 | 0.166666667 | -0.166667245 | -0.166666667 |
| | 0.144337621 | 0.144337567 | 0.144337621 | 0.144337567 |
| | 0.083333449 | 0.083333333 | -0.083333449 | -0.833333333 |
| | 0.144337004 | 0.144337567 | 0.144337004 | 0.144337567 |
| | 0.250000900 | 0.250000000 | -0.250000900 | -0.250000000 |
| | 0.288675145 | 0.288675135 | 0.288675145 | 0.288675135 |
| | 0.250000900 | 0.250000000 | -0.250000000 | -0.250000000 |
| | 0.144337583 | 0.144337567 | 0.144337583 | 0.144337567 |
| | 0.166667245 | 0.166666667 | -0.166667245 | -0.166666667 |
| | 0.288675145 | 0.288675135 | 0.288675135 | 0.288675135 |
| | 0.333333449 | 0.333333333 | -0.333333333 | -0.333333333 |
| | 0.288675145 | 0.288675135 | 0.288675135 | 0.288675135 |
| | 0.166666246 | 0.166666667 | -0.166666667 | -0.166666667 |
| | 0.144337580 | 0.144337567 | 0.144337580 | 0.144337567 |
| | 0.250000900 | 0.250000000 | -0.250000000 | -0.250000000 |
| | 0.288675135 | 0.288675135 | 0.288675135 | 0.288675135 |
| | 0.250000900 | 0.250000000 | -0.250000000 | -0.250000000 |
| | 0.144337580 | 0.144337567 | 0.144337567 | 0.144337567 |
| | 0.083333700 | 0.083333333 | -0.083333333 | -0.083333333 |
| | 0.144337580 | 0.144337567 | 0.144337580 | 0.144337567 |
| | 0.166666246 | 0.166666667 | -0.166666245 | -0.166666667 |
| | 0.144337581 | 0.144337567 | 0.144337621 | 0.144337567 |
| | 0.083333700 | 0.083333333 | -0.083333700 | -0.083333333 |

TABLE 7.6

# CHAPTER 8

# CONCLUDING REMARKS

## 8.1

Our principal aim in this thesis was to present several fast direct algorithmic methods for the solution of sparse matrix equations which are derived from the finite difference approximation of partial differential equations associated with some functions of Mathematical Physics.

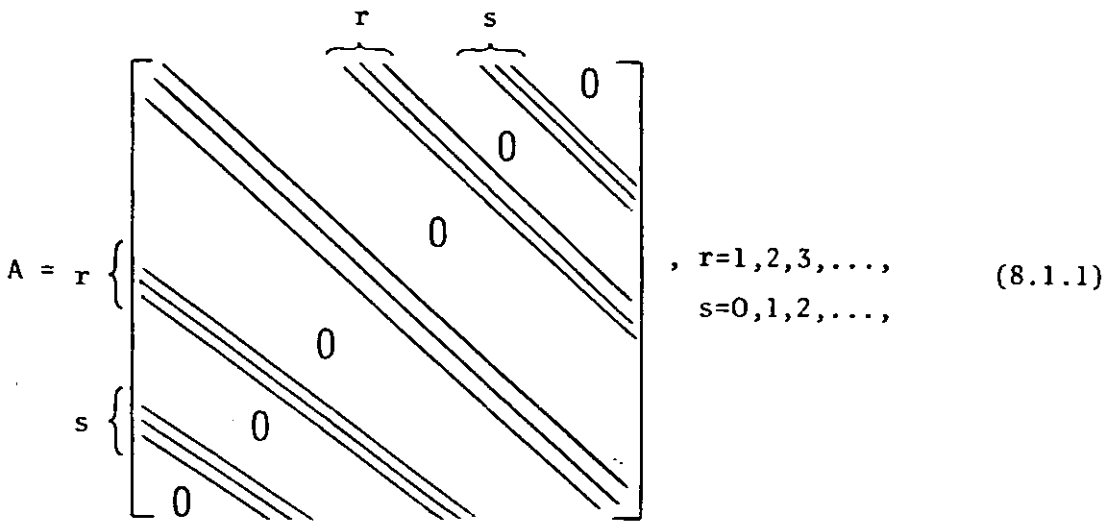
In nearly all cases, our approach in the development of the algorithms has been to assume diagonal dominance of the matrix systems (this being the case in physical problems); and thus this enabled the stability of the algorithmic methods proposed to be guaranteed without the necessity of introducing a pivoting strategy. The consequence of this has been that the algorithms derived possess the speed and efficiency characteristic of fast methods. Under special conditions, variants of some of the techniques proposed have produced algorithms which handle singular systems (including those with zero diagonal entries) with remarkable speed as was the case, for example, in algorithm (4.7). Some of the algorithms, apart from being new, are comparable (and in some cases, show improvements) in speed, storage requirements and/or accuracy, to existing known methods for solving identical problems.

Our emphasis has been largely on methods which are directly applicable to the solution of linear elliptic and parabolic partial differential equations defined over rectangular regions with periodic, Dirichlet's or Neumann's boundary conditions. For problems defined on a non-rectangular region, (e.g., an L-shaped domain), the structure of the discrete matrix equation does not always permit the immediate application of the direct methods we have proposed. However, the use of the 'inbedding' and 'split-up' techniques in a 'capacitance' matrix approach (Buzbee, Dorr, George and Golub (1971)) makes it feasible to extend some of our methods to problems defined in non-rectangular regions under similar boundary conditions.

Finally, it is hoped that a few of the algorithmic methods proposed in this work will be extended in order to widen the scope of their

applicability. In this direction, the following are of special interest:

(a) An extension of the recursive point partitioning (R.P.P.) algorithmic idea to systems with periodic conditions; and others with more complicated sparse banded structure of the form,




which are usually associated with discretised quasi-linear or non-linear p.d.e.'s in two or three space dimensions, is desirable.

(b) The periodic quotient-difference (P.Q.D.) algorithm proposed in section (7.2) for calculating the eigenvalues of periodic tridiagonal matrices converges only under diagonal dominance and positive definite conditions. Further investigation is necessary to widen the scope of this algorithm to apply to less well behaved systems.

(c) Finally, the iterative scheme (7.6.11) proposed in section (7.6) for calculating the largest and smallest eigenvalues and eigenvectors of sparse banded matrix systems of the form (8.1.1) requires further analysis and extension. The theoretical results of the convergence behaviour of the iterative scheme need to be rigorously established. It is also hoped to introduce modifications that will enable intermediate eigenpairs to be obtained.

# REFERENCES

AHLBERG, H.H., NILSON, E.N., and WALSH, J.L., (1967), *"The Theory of Splines and Their Applications"*, Academic Press, New York.

AMES, W.F.(1969), *"Numerical Methods for Partial Differential Equations"*, Nelson.

BABUSKA, I., PRAGER, M., and VITASEK, E.,(1966), *"Numerical Processes in Differential Equations"*, John Wiley.

BANEGAS, A.,(1978), *"Fast Poisson Solver for Problems with Sparsity"*, Maths. of Comp., Vol.32, No.142, p.141-146.

BANK, R.E.,(1975), *"An $O(n^2)$ Method for Solving Constant Coefficient Boundary Value Problems in Two Dimensions"*, SIAM.J.Numer.Anal.,Vol.12,No.4, p.529-542.

BANK, R.E.,(1976), *"Marching Algorithms and Gaussian Elimination"*, Proc. Symposium on Sparse Matrix Computations, Argonne National Lab., (Bunch J.R. and Rose D.J., Eds.), Academic Press

BARLOW R.H., EVANS, D.J., NEWMAN, I.A., SLADE, A.J., and WOODWARD, M.C.,(1977), *'Historical Survey of the Implementation of Parallel Programming on the Interdata Dual Processor Computer"*, Loughborough Univ. of Technology, Dept. Comp.Studies, Report 40.

BARTH W., MARTIN, R.S., and WILKINSON, J.H.,(1967), *"Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection"*, Numerische M., Vol. 9, p.386-393.

BAUER, L. and REISS, E.L.,(1972), *"Block Five Diagonal Matrices and Their Fast Numerical Solution of the Biharmonic Equations"*, Math.Comp. Vol.26, No.18, p.311-326.

BENSON, A.,(1969), *"The Numerical Solution of Partial Differential Equations by Finite Difference Methods"*, Ph.D. Thesis, Univ. of Sheffield.

BJORCK, A., and GOLUB, G.H.,(1977), *"Eigenproblems for Matrices Associated with Periodic Boundary Conditions"*, SIAM. Rev. Vol. 19, p.5-16.

BLANCH, G.,(1964), *"Numerical Evaluation of Continued Fractions"*, SIAM. Rev. Vol. 6, No. 4, p.383-419.

BUDAK, B.M., SAMASKII, A.A. and TIKHONOV, A.N.,(1964), *"A Collection of Problems on Mathematical Physics"*, (translated by Brink, D.M.), Pergamon Press.

BUNEMAN, O.(1969), *"A Compact Non-Iterative Poisson Solver"*, Rep.294, Standford Univ.Inst. of Plasma Research.

BUZBEE, B.L. GOLUB, G.H., and NIELSON, C.W.(1970), *"On Direct Methods for Solving Poisson Equations"*, SIAM.J.Numer.Anal.,Vol. 7, No. 4, p.627-656.

BUZBEE, B.L., DORR, F.W., GEORGE, J.A., GOLUB, G.H.(1971), *"The Direct Solution of the Discrete Poisson Equation on Irregular Regions"*, SIAM. J.Numer.Anal., Vol.8, No. 4, p.722-736.

BUZBEE, B.L., DORR, F.W.,(1974), *"The Direct Solution of the Biharmonic Equation on Rectangular Regions and the Poisson Equation on Irregular Regions"*, SIAM.J.Numer.Anal., Vol.11, p.753-761.

CARASSO, A. and PARTER, S.V.,(1970), *"An Analysis of Boundary Value Techniques for Parabolic Problems"*, Math.Comp., Vol.24, No. 110, p.315-340.

CONTE, S.D. and DAMES, R.T.(1958), *"An Alternating Direction Method for Solving the Biharmonic Equation"*, Math. Tables Aids Comp. Vol.12, p.198-205.

COOLEY, J.W. and TUKEY, J.W.(1965), *"Algorithm for Machine Calculation of Complex Fourier Series"*, Math.Comp., Vol.19, p.297-301.

CRANK, J. and NICOLSON, P.,(1947), *"A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat Conduction Type"*, Proc.Camb.Phil.Soc., 43, p.50-67.

DIAMOND, M.A., FERREIRA, D.L.V.(1976), *"On Cyclic Reduction Method for the Solution of Poisson's Equation"*, SIAM.J.Numer.Anal., Vol.13, No.1, p.54-67.

DORR, F.W.,(1970), *"The Direct Solution of the Discrete Poisson Equation on a Rectangle"*, SIAM. Rev., Vol. 12, No. 2, p.248-272.

EVANS, D.J.,(1970), *"The Numerical Solution of Elliptic and Parabolic Equations Occurring in Computational Physics"*, in 'Computational Physics Conference Digest', p.46-56, Inst. of Physics and Physical Society, London.

EVANS, D.J.,(1971A), *"The Numerical Solution of the Fourth Boundary Value Problem for Parabolic Differential Equations"*, J.Inst.Maths. Applics., Vol. 7, p.61-75.

EVANS, D.J.,(1971B), *"Numerical Solution of the Sturm Liouville Problem with Periodic Boundary Conditions"*, Conf. on Applics. of Num.Anal., Springer Verlag, p.277-288.

EVANS, D.J.,(1972), *"An Algorithm for the Solution of Certain Tridiagonal Systems of Linear Equations"*, Comput.J., Vol. 15, No. 4, p.356-359.

EVANS, D.J. and ATKINSON, L.V.,(1970), *"An Algorithm for the Solution of General Three Term Linear Systems"*, Comput.J., Vol. 13, No. 3, p.323-324.

EVANS, D.J. and OKOLIE, S.O., (1977), *"A Generalised Factorisation Method for the Solution of the General Cyclic Tridiagonal System of Linear Equations"*, Loughborough Univ., Dept. of Comp.Studies, Report 53.

FORSYTHE, G.E., and WASOW, W.R.(1960), *"Finite Difference Methods for Partial Differential Equations"*, John Wiley.

FOX, L.,(1964), *"Introduction to Numerical Linear Algebra"*, O.U.P.

FRANCIS, J.G.F.,(1961/62), *"The Q.R. Transformation - A Unitary Analogue to the L.R. Transformation"*, Parts I and II, Comput.J., Vol. 4, p.265-271, 332.343.

FRANKEL, S.P.,(1950), *"Convergence Rates of Iterative Treatments of Partial Differential Equations"*, Math. Tables Aids Comput., Vol. 4, p.65-75.

FROBERG, C.E.(1965), *"Introduction to Numerical Analysis"*, Addison-Wesley Pub.Co.

GREENSPAN, D.,(1974), *"Discrete Numerical Methods in Physics and Engineering"*, Academic Press.

GREGORY, R.T., KARNEY, D.L.(1969), *"A Collection of Matrices for Testing Computational Algorithms"*, Wiley Interscience.

GOLUB, G.H., ROBERSTON, T.N.,(1967), *"A Generalised Bairstow Algorithm"*, Comm.A.C.M., Vol. 10, No. 6, p.371

HOCKNEY, R.W., (1965), *A Fast Direct Solution of Poisson Equation Using Fourier Analysis"*, J.A.C.M., Vol. 12, p.95-113.

HOCKNEY, R.W.,(1970), *"The Potential Calculation and Some Applications"*, in Methods in Computational Physics, Vol. 9, (B. Adder, S. Fernbach, S. Rotenberg, Eds.), Academic Press, p.135-211.

HOUSEHOLDER, A.S.(1964), *"The Theory of Matrices in Numerical Analysis"*, Blaisdell Publishing Co.

LAASONEN, P.,(1958), *"On the Iterative Solution of the Matrix Equation $AX^2-I=0$"*, Math. Tables Aids Comp., Vol. 12, p.109-116.

MARCHUK, G.I.(1975), *"Methods of Numerical Mathematics"*, (Translated by Ruzicka, J.), Springer Verlag.

MIKLIN, S.G.,(1964), *"Linear Equations of Mathematical Physics"*, Holt, Rinehart and Winston.

MITCHELL, A.R.,(1976), *"Computational Methods in Partial Differential Equations"*, John Wiley.

N.A.G. Library Manual MK 5, N.A.G. Ltd., Oxford.

NOBLE, B.,(1969), *"Applied Linear Algebra"*, Prentice-Hall Inc.

O'BRIEN, G.G., HYMAN, M.A. and KAPLAN, S.J.(1951), *"A Study of the Numerical Solution of Partial Differential Equations"*, J. Math.Phys. 29, p.223-251.

OSBORNE, M.R.,(1965), *"A Note on the Numerical Solution of a Periodic Parabolic Problem"*, Numerische M. Vol. 7, p.155-158.

PEACEMAN, D.W., RACHFORD, H.H.(1955), *"The Numerical Solution of Parabolic and Elliptic Differential Equations"*, J. SIAM, Vol. 3, No. 28, p.28-41.

POLOZHII, G.N.,(1974), *"The Method of Summary Representation for Numerical Solution of Problems of Mathematical Physics"*, (Translated by Gee, G.J.), Pergamon Press.

RICK, C.C.,(1977), *"The Numerical Determination of the Eigenvalues and Eigenvectors of Large Order Sparse Matrices"*, Ph.D. Thesis, Loughborough Univ. of Tech.

RICHTMYER, R. and MORTON K.W.,(1967), *"Difference Methods for Initial Value Problems"*, John Wiley.

ROSSER, J.B.,(1967), *"The Direct Solution of Difference Analogs of Poisson's Equation"*, Rep. MRC-TSR-797, Mathematics Research Centre, Madison, Wisconsin.

RUTISHAUSER, H.,(1958), *"Solution of Eigenvalue Problem with L.R. Transformation"*, NBS Appl.Maths. Series, No.49, p.47-48.

SCHWARTZ, H.R., RUTISHAUSER, H., STIEFEL, E.,(1973), *"Numerical Analysis of Symmetric Matrices"*, Prentice-Hall Inc.

SCOFIELD, D.F.,(1973), *"A Note on Löwdin Orthogonalisation and the Square Root of a Positive Self-Adjoint Matrix"*, Inter.J. of Quantum Chem., Vol. VII, p.561-568.

SHAVITT, I., BENDER, C.F., PIPANO, A., HOSTENY, P.,(1973), *"Iterative Calculation of Several of the Lowest or Highest Eigenvalues and Corresponding Eigenvectors of Very Large Symmetric Matrices"*, J. of Comput. Phy., Vol. 11, p.90-108.

SMITH, G.D.,(1969), *"Numerical Solution of Partial Differential Equations"*, O.U.P.

SPÄTH, H.,(1967), *"Algorithm 298: Determination of the Square Root of a Positive Definite Matrix [F1]"*, Comm.A.C.M., Vol. 10, No. 3, p.182.

STEIN, P. and ROSENBERG, R.K.,(1948), *"On the Solution of Linear, Simultaneous Equations by Iteration"*, J. London Math.Soc., 23, p.111-118.

SWARZTRAUBER, P.N.,(1974), *"A Direct Method for the Discrete Solution of Separable Elliptic Equations"*, SIAM J.Numer.Anal., Vol. 11, No. 6, p.1136-1150.

SWEET, R.A.,(1974), *"A Generalised Cyclic Reduction Algorithm"*, SIAM J. Numer.Anal., Vol. 11, No. 3, p.506-520.

TEE, G.J.,(1964), *"An Application of p-cyclic Matrices, for Solving Periodic Parabolic Problems"*, Numerische M., Vol. 6, p.142-159.

TEMPERTON, C.,(1975), *"Algorithms for the Solution of Cyclic Tridiagonal Systems"*, J. of Comput.Phys., 19, p.317-323.

THOMAS, L.H.,(1949), *"Elliptic Problems in Linear Difference Equations Over a Network"*, Watson Scientific Computing Lab., Columbia Univ.

VARGA, R.S.,(1962), *"Matrix Iterative Analysis"*, Prentice-Hall, Englewood Cliff.

VON ROSENBERG, D.U.,(1969), *"Methods for the Numerical Solution of Partial Differential Equations"*, (Bellman, R. Editor), Elsevier Pub.Co.Inc.

WALL, H.S.,(1948), *"Analytic Theory of Continued Fractions"*, D. Van Nostrand Co.

WILKINSON, J.H.,(1961), *"Error Analysis of Direct Methods of Matrix Inversion"*, J.Assoc.Comput.Mach., 8, p.281-330.

WILKINSON, J.H.,(1963), *"Rounding Error in Algebraic Processes"*, H.M.S.O.

WILKINSON, J.H., (1965), *"The Algebraic Eigenvalue Problem"*, O.U.P.

YANG, W.H., LEE, E.H.(1974), *"Model Analysis of Floquet Waves in Composite Materials"*,

YOUNG, D.M.(1954), *"Iterative Methods for Solving Partial Differential Equations of Elliptic Type"*, Trans.Amer.Math.Soc. 76, p.92-111.

# APPENDIX I

This Appendix contains the listing of some of the programs in the form of subroutines written, using single precision arithmetic, for the various algorithms described in this thesis, and used in obtaining most of the numerical results indicated in the text.

All the programs were written in FORTRAN IV and run on the Loughborough University I.C.L. 1904S computer. The exception to this includes Programs 12 and 13 which were both written for and run on the Loughborough University Interdata parallel computer.

```
C          ***********          PROGRAM      1          ************
C       SUBROUTINE GAUSELIM(N,A,B,C,RHS,X,P,Q,U,S,T,V)
C       THIS SUBROUTINE SOLVES A PERIODIC TRIDIAGONAL
C       MATRIX EQUATION BY A VARIANT OF GAUSS ELIMINATION
C       PROCESS WITHOUT A PIVOTING STRATEGY (SEE ALGORI-
C       THM (3,1).
C       ON INPUT...A,B,C ARE VECTORS HOLDING THE SUB-DIA-
C       GONAL, DIAGONAL AND SUPER-DIAGONAL ELEMENTS OF
C       THE PERIODIC MATRIX OF ORDER N; RHS HOLDS THE
C       RIGHT-HAND SIDE VECTOR. P,Q,U,S,T ARE WORKING
C       SPACE VECTORS. ON EXIT, X HOLDS THE SOLUTION.
        DIMENSION A(N),B(N),C(N),RHS(N),X(N),P(N),Q(N),
      1U(N),S(N),T(N),V(N)
        Q0=0.0
        S0=1.0
        P(1) =A(1)*Q0 +B(1)
        Q(1) =-C(1)/P(1)
        S(1) =-A(1)*S0/P(1)
        DO 5 I=2,N
        P(I)= A(I)*Q(I-1)+B(I)
        Q(I)=-C(I)/P(I)
    5   S(I)=-A(I)*S(I-1)/P(I)
        T(N)=1.0
        DO 10 I= 1,N-1
        II =N-I
   10   T(II)=Q(II)*T(II+1)+S(II)
   15   U0 =0.0
        U(1)=(RHS(1)-A(1)*U0)/P(1)
        DO 20 I =2,N
   20   U(I)=(RHS(I)-A(I)*U(I-1))/P(I)
        V(N)=0.0
        DO 25 I=1,N-1
        II=N-I
   25   V(II)=Q(II)*V(II+1)+U(II)
        X(N)=(RHS(N)-C(N)*V(1)-A(N)*V(N-1))/(C(N)*T(1)+
      1A(N)*T(N-1)+B(N))
        DO 30 I=1,N-1
        II= N-I
   30   X(II)=T(II)*X(N)+V(II)
        RETURN
        END


C          ***********          PROGRAM      2          ************
C       THIS PROGRAM IMPLEMENTS THE PQFACT ALGORITHM(3.2)
C       IN THE FORM OF 3 SUBROUTINES GIVEN BELOW CORRESP-
C       ONDING TO THE 3 STEPS OF THE ALGORITHM.


C       SUBROUTINE FACT(N,A,B,C,ALP,BETA,L,U)
C       THIS SUBROUTINE PERFORMS A GENERALISED SPARSE CY-
C       CLIC FACTORISATION OF A GENERAL PERIODIC TRIDIAG-
C       ONAL MATRIX. A CALL TO CF1 SUBROUTINE IS MADE TO
C       OBTAIN THE VALUE OF AN INFINITE CONTINUED FRACT-
C       ION ARISING FROM THE FACTORISATION. ON ENTRY, A,
C       B,C HOLD THE SUB-DIAGONAL, DIAGONAL AND SUPER-
C       DIAGONAL ELEMENTS OF INPUT PERIODIC TRIDIAGONAL
C       MATRIX. ALP BETA,L,U ARE AUXILIARY VECTORS HOLD-
C       ING COEFFS WHICH ON EXIT, ARE INPUTED TO THE PQS-
```

```
C       OL SUBROUTINE. N IS THE ORDER OF MATRIX.
        REAL L
        DIMENSION A(N),B(N),C(N),L(N),U(N),ALP(N),BETA(N)
        EPS=1.0E-12
        CALL CF1(A,B,C,N,CNL1,ALP,BETA,EPS)
C       OBTAIN  L(I),U(I), I =1,N
        L(1) =CNL1/C(N)
        U(1) =B(1)- CNL1
        DO 5 I=2,N
        L(I)=A(I)/U(I-1)
     5  U(I)=B(I)-L(I)*C(I-1)
C       OBTAIN PRE-COMPUTED COEFS AND STORE IN ALP,BETA;
        DO  30  K=1,N
        ALP(K) =1.0
        DO 20 J =1,N
    20  ALP(K)=ALP(K)*L(J)
        IF((K-K/2*2).EQ.0) ALP(K)= -ALP(K)
    30  CONTINUE
        DO 45 I=1,N
        BETA(I) =1.0
         DO 40 J=1,N
    40  BETA(I)=BETA(I)* C(J)/U(J)
        N1= N-N/2*2
        I1= I-I/2*2
        ISIGN = 1
        IF(N1.NE.I1)ISIGN = -1
    45  BETA(I)= ISIGN * BETA(I)
        RETURN
        END


        SUBROUTINE POSOL(N,C,L,U,RHS,ALP, BETA,X )
C       THIS SUBROUTINE SOLVES A GENERAL PERIODIC TRIDIA-
C       GONAL MATRIX OF ORDER N. PRE-COMPUTED COEFFS HELD
C       IN L,U,ALP,BETA ARE INPUTED FROM FACT SUBROUTINE.
C       ON ENTRY RHS HOLDS THE RIGHT-HAND SIDE VECTOR AND
C       ON EXIT, X HOLDS THE SOLUTION VECTOR.
        REAL L
        DIMENSION L(N),U(N),RHS(N),ALP(N),BETA(N),X(N),
       1C(N)

C       SOLVE P*Y=RHS; P IS A CYCLIC 2-TERM 'LOWER TRIAN-
C       GULAR MATRIX'.
         X(1)=RHS(1)
        DO 10 I=1,N
        IF(I.EQ.1) GOTO 10
         X(I)=RHS(I)-L(I)*X(I-1)
    10  CONTINUE
         X(N)= X(N)/(1.0+ALP(N))
        DO 15 K =1,N-1
    15   X(K)= X(K)-ALP(K)*X(N)
C       SOLVE Q*X=Y; Q IS A CYCLIC 2-TERM 'UPPER TRIANG-
C       ULAR MATRIX'.
        DO 20 I =1,N
    20   X(I)= X(I)/U(I)
        DO 30 I =1,N
        II =N-I+1
        IF(II.EQ.N)GOTO 30
         X(II)=X(II)-X(II+1)*C(II)/U(II)
    30  CONTINUE
```

```
      X(1)=X(1)/(1.0+BETA(1))
      DO 40 I =2,N
   40 X(I)= X(I)-BETA(I)*X(I-1)
      RETURN
      END


      SUBROUTINE CF1(N,A,B,C,VL,ALP,BETA,EPS)
C     THIS SUBROUTINE EVALUATES AN INFINITE PERIODIC
C     CONTINUED FRACTION (3.2.27) BY A FORWARD RECURR-
C     ENCE SCHEME(3.2.31) AND A SOLUTION OF A QUADRATIC
C     EQUATION. N IS THE CYCLE LENGTH OF THE PERIODIC
C     CONTINUED FRACTION. A,B,C ARE VECTORS HOLDING THE
C     ELEMENTS OF THE COEFF MATRIX; ALP,BETA ARE WORK-
C     ING SPACE VECTORS AND EPS IS TRUNCATION ERROR TO-
C     LERANCE. VL IS RETURNED AS THE VALUE OF CONTINUED
C     FRACTION BEING SOUGHT.
      DIMENSION A(N),B(N),C(N),ALP(N),BETA(N)
C     DEFINE ALP,BETA IN TERMS OF A, B, C
      DO 10 I =1,N
      II = N-I+1
      JJ = II+1
      IF(JJ.GT.N)JJ=1
      ALP(I)= C(II)* A(JJ)
      BETA(I)=B(II)
   10 CONTINUE
C     OBTAIN THE NTH APPROXIMANT OF INFINITE C.FRACTION.
      E0 =0.
      F0 =1.0
      F1 = ALP(1)
      F1 = BETA(1)
      DO 20 I=2, N
      E2 =BETA(I)*E1-ALP(I)*E0
      F2 =BETA(I)*F1-ALP(I)*F0
      T2 =E2/F2
      T1 =E1/F1
      T  =T2-T1
      IF( ABS(T).LE.EPS)GOTO 30
      E0 = E1
      E1 = E2
      F0 = F1
      F1 = F2
   20 CONTINUE
C     SOLVE QUADRATIC EQUATION:F1*W**2+(F2-E1)*W -E1 =0
   30 CONTINUE
      R =  F2-E1
      S= 4*F1*E2
      VL = (-R + SQRT(R*R +S))/(2*F1)
      RETURN
      END


C        ***********        PROGRAM     3        ************
      SUBROUTINE  GENHOCNEY(N,A,B,C,RHS,X,K )
C     THIS SUBROUTINE SOLVES A GENERAL PERIODIC TRIDIA-
C     GONAL MATRIX EQUATION OF ORDER N (=2**K) BY A
C     GENERALISATION OF THE CYCLIC REDUCTION METHOD AS
C     GIVEN IN ALGORITHM(3.3). ON ENTRY... A,B,C HOLD
C     THE SUB-DIAGONAL,DIAGONAL AND SUPER-DIAGONAL ELE-
```

```
C      MENTS OF COEFF MATRIX RESPECTIVELY; A(1),C(N) ARE
C      THE CORNER ELEMENTS. RHS IS THE RIGHT-HAND VECTOR.
C      ON EXIT,A,B,C ARE DESTROYED AND X HOLDS THE SOLU-
C      TION VECTOR.
       DIMENSION A(N),B(N),C(N), X(N),RHS(N)
C      PERFORM THE CYCLIC REDUCTION OF THE MATRIX SYSTEM
        DO 15 IT=1, K
        J =IT-1
        DO 10 I=2**IT,N,2**IT
        II=I-2**J
         IJ=I+2**J
        IF(II.LE.0) II=N-II
        IF(IJ.GT.N) IJ=IJ-N
       TEMP1 = B(IJ)* A(I)
       TEMP2 = B(II)* B(IJ)
       TEMP3 = C(I)* B(II)
       A(I)  = TEMP1*A(II)
       B(I)  = TEMP1* C(II)-TEMP2*B(I)+TEMP3*A(IJ)
       C(I)  = TEMP3* C(IJ)
       RHS(I)=TEMP1*RHS(II)-TEMP2*RHS(I)+TEMP3*RHS(IJ)
   10 CONTINUE
   15 CONTINUE
C      OBTAIN X(N)-COMPONENT
         DEN = A(N)+B(N)+C(N)
         X(N)= RHS(N)/DEN
       DO  45 IT=1, K-1
         ITT = K- IT
           DO 40  I = 2** ITT, N-2**ITT,2**(ITT+1)
             II = I - 2** ITT
             JJ = I + 2** ITT
           IF(II.LE.0) II =N -II
           IF(JJ.GT.N) JJ =JJ-N
   40  X(I)=(RHS(I)-A(I)*X(II)-C(I)*X    (JJ))/B(I)
   45 CONTINUE
C      BACK-SUBSTITUTION TO OBTAIN SOLUTION VECTOR.
        DO 50 I=1,N-1,2
        II = I-1
        JJ = I+1
        IF(II.LE.0) II =N-II
        IF(II.GT.N) JJ = JJ-N
   50  X(I)= (RHS(I)-X(II)* A(I)- X(JJ)* C(I)) /B(I)
       RETURN
       END



C         ***********          PROGRAM      4          ************
       SUBROUTINE BUNEMAN(N,A,B,C,P,Q,X,K )
C      THIS SUBROUTINE SOLVES A GENERAL PERIODIC TRIDIA-
C      GONAL MATRIX EQUATION OF ORDER N(=2**K) BY A GEN-
C      ERALISATION OF BUNEMAN'S STABLE MODIFICATION OF
C      THE CYCLIC REDUCTION SCHEME AS GIVEN IN ALGORITHM
C      (3.4). ON ENTRY... A,B,C HOLD THE SUB-DIAGONAL,
C      DIAGONAL AND SUPER-DIAGONAL ELEMENTS; A(1),C(N)
C      ARE THE CORNER ELEMENTS. X HOLDS THE INPUT R.H.S.
C      VECTOR AND P,Q ARE AUXILIARY VECTORS WITH WHICH
C      R.H.S VECTOR IS MODIFIED TO PREVENT THE LATTER
C      GROWING EXCESSIVE IN SIZE. ON EXIT X HOLDS THE
C      SOLUTION VECTOR.
       DIMENSION A(N),B(N),C(N),X(N),P(N),Q(N)
```

```
      DO 5 I=1,N
      P(I)=X(I)/B(I)
      Q(I)= 0.0
    5 CONTINUE
C     PERFORM THE CYCLIC REDUCTION PROCESS
      DO 15 IT=1, K
      J =IT-1
      DO 10 I=2**IT,N,2**IT
      II=I-2**J
       IJ=I+2**J
      IF(II.LE.0) II=N-II
      IF(IJ.GT.N) IJ=IJ-N
      B0=B(I)
      A0=A(I)
      C0=C(I)
      TEMP1 = B(IJ)* A(I)
      TEMP3 = C(I)* B(II)
      TEMP2 =TEMP1* C(II)+TEMP3* A(IJ)
      A(I)  = TEMP1*A(II)
      B(I)= TEMP2 -B(I)*B(II)*B(IJ)
      C(I)  = TEMP3* C(IJ)
      P(I)=P(I)+(Q(I)-A0*P(II)-C0*P(IJ))/B0
      Q(I)=TEMP1*Q(II)+TEMP3*Q(IJ)-TEMP2*P(I)
   10 CONTINUE
   15 CONTINUE
C     BACK-SUBSTITUTION TO OBTAIN SOLUTION VECTOR.
        DEN = A(N)+B(N)+C(N)
      X(N)=(B(N)*P(N)+ Q(N))/DEN
      DO  45 IT=1, K-1
        ITT = K- IT
          DO 40  I = 2** ITT, N-2**ITT,2**(ITT+1)
            II = I - 2** ITT
            JJ = I + 2** ITT
          IF(II.LE.0) II =N -II
          IF(JJ.GT.N) JJ =JJ-N
   40 X(I)=P(I)+(Q(I)-A(I)*X(II)-C(I)*X(JJ))/B(I)
   45 CONTINUE
       DO 50 I=1,N-1,2
       II = I-1
       JJ = I+1
       IF(II.LE.0) II =N-II
       IF(II.GT.N) JJ = JJ-N
   50 X(I)=P(I)+(Q(I)-A(I)*X(II)-C(I)*X(JJ))/B(I)
      RETURN
      END



C        ***********        PROGRAM      5        ***********
      SUBROUTINE RANKONE(N,A,B,C,RHS,X,D,BETA,GEMA)
C     THIS SUBROUTINE SOLVES THE PERIODIC TRIDIAGONAL
C     MATRIX EQUATION BY THE RANK-ONE MODIFICATION METH-
C     OD AS GIVEN IN ALGORITHM(3.5). THOMAS ALGORITHM
C     (3.5) IS USED TO SOLVE THE RESULTING TRID.SYSTEMS.
C     ON ENTRY...A,B,C ARE VECTORS HOLDING THE SUB-DIAG-
C     ONAL, DIAGONAL AND SUPER DIAGONAL ELEMENTS OF
C     COEFF MATRIX; RHS HOLDS THE R.H.S VECTOR AND BETA,
C     ALP,GEMA,D ARE AUXILIARY VECTORS. ON EXIT, X
C     HOLDS THE SOLUTION VECTOR.
      DIMENSION A(N),B(N),C(N),RHS(N),D(N),X(N),
```

```
     1BETA(N),GEMA(N)
C      CONVERT INPUT GENERAL TRIDIAGONAL MATRIX TO
C      SYMMETRIC TRIDIAGONAL FORM.
       D(1)=1.0
       DO 1 I=1,N-1
     1 D(I+1) = D(I) * SQRT(C(I)/A(I+1))
       A(1) = D(1)*A(1)/D(N)
        C(N)=A(1)
       DO 2 I=1,N-1
       A(I+1)=A(I+1)*D(I+1)/D(I)
     2 C(I)  =C(I)*D(I)/ D(I+1)
       W = A(1)
       T1=B(1)
       TN=B(N)
       B(1)= B(1)-W
       B(N)= B(N)-W
       D(1)= W
       D(N)= W
       IFLAG=0
       BETA(1) = B(1)
       DO 10 I =2,N
    10  BETA(I)=B(I)-A(I)*C(I-1)/BETA(I-1)
       DO 5 I =2,N-1
     5 D(I)=0.0
    12 CONTINUE
C      SOLVE TRIDIAGONAL MATRIX BY THOMAS ALGORITHM.
       GEMA(1) = D(1)/B(1)
        DO 15 I=2,N
    15   GEMA(I)= (D(I)-A(I)*GEMA(I-1))/BETA(I)
       X(N) = GEMA(N)
       DO 20 I =1,N-1
       II = N-I
    20 X(II)=GEMA(II)- C(II)*X(II+1)/BETA(II)
       IF(IFLAG.EQ.1)GOTO 25
C      OBTAIN S,USED TO MODIFY 1ST AND LAST ELEMENTS OF
C      R.H.S VECTOR FOR THE SECOND TRIDIAGONAL SYSTEM.
       S=0.
       DO 25 I=1,N
    25 S=S+ X(I)* RHS(I)
       S= S/(1.0 + X(1)+X(N))
C      COMPUTE NEW RHS-VECTOR FOR SECOND TRID  SYSTEM
        D(1)= RHS(1)-S
        D(N)=RHS(N)-S
       DO 30 I=2,N-1
    30 D(I) =RHS(I)
       IFLAG =1
       GOTO 12
    35 B(1) =T1
       B(N)=TN
       RETURN
       END


C        ***********          PROGRAM      6          ************
       SUBROUTINE PQFACT1(N,B,C,RHS,PH,X)
C      THIS SUBROUTINE SOLVES A CONSTANT TERM PERIODIC
C      TRID. MATRIX OF THE FORM A[C,B,C] OF ORDER N BY
C      A FACTORISATION METHOD AS GIVEN IN ALGORITHM(3.6).
C      ON INPUT... RHS HOLDS THE INPUT RIGHT HAND SIDE;
```

```
C       PH IS A WORKING SPACE VECTOR. ON EXIT, THE SOLUT-
C       ION VECTOR IS HELD IN X.
        DIMENSION RHS(N),PH(N),X(N)
        AL=0.5 *(B-SQRT(B*B-4*C*C))/C
        RHO =AL/C
        PH(1)=AL
        DO 8 I=2,N
    8    PH(I)=-AL*PH(I-1)
        DO 10 I =1,N
        IF(I.EQ.1) GOTO 10
        RHS(I) = RHS(I)-AL* RHS(I-1)
   10 CONTINUE
        X(N)= RHS(N)/( 1.0 +PH(N))
        DO 15 I =1, N-1
   15 X(I)= RHS(I) -PH(I)* X(N)
        RHS(N)=RHO *X(N)
        DO 20 I=1,N-1
        II= N-I
   20 RHS(II)= RHO*X(II)- AL* RHS(II+1)
        X(1)=RHS(1)/(1.0+PH(N))
         DO 30 I=2,N
         II=N-I+1
        X(I)=RHS(I)-PH(II)*X(1)
   30  CONTINUE
        RETURN
        END


C         ***********          PROGRAM      7          ***********
        SUBROUTINE PQFACT4(N,B,RHS,PH,X)
C       THIS SUBROUTINE SOLVES THE CONSTANT TERM SKEW-
C       SYMMETRIC PERIODIC TRIDIAGONAL MATRIX OF THE FORM
C       A[1,B,-1] OF ORDER N, WHERE B IS A CONSTANT .GE.2,
C       BY THE METHOD OF ALGORITHM(3.9). ON ENTRY, RHS
C       HOLDS THE RIGHT HAND VECTOR AND ON EXIT X HOLDS
C       THE SOLUTION VECTOR.
        DIMENSION RHS(N),PH(N),X(N)
        AL= 0.5*( -B+SQRT(B*B+4.0))
        PH(1)=-AL
        DO 8 I=2,N
      8 PH(I)=AL*PH(I-1)
        DO 10 I=1,N
        IF(I.EQ.1)GOTO 10
        RHS(I)= RHS(I)+ AL* RHS(I-1)
     10 CONTINUE
        X(N)= RHS(N)/(1.0+ PH(N))
        DO 15 I =1,N-1
     15 X(I)=RHS(I)- PH(I)* X(N)
        RHS(N) = AL* X(N)
        DO 20 I=1,N-1
        II=N-I
     20  RHS(II)=AL*X(II)-AL*RHS(II+1)
        ISIGN =1
        N1=N-N/2*2
        IF(N1.EQ.1) ISIGN = -1
        X(1)=RHS(1)/(1.0+ISIGN*PH(N))
        DO 30 I=2,N
        ISIGN=1
        I1=I-I/2*2
```

```
      II =N-I+1
      IF(N1.EQ.I1) ISIGN =-1
   30 X(I) =RHS(I)= ISIGN* PH(II)* X(1)
      RETURN
      END



C          **********          PROGRAM      8          **********
      SUBROUTINE GENRET(N,NR,B,C,ALPHA,RHS,Y)
C     THIS SUBROUTINE SOLVES A GENERAL SYMMETRIC TRID-
C     MATRIX SYSTEM OF ORDER N BY THE GENERALISED  RECT-
C     ANGULAR FACTORISATION AND EXPANSION STRATEGY AS
C     GIVEN BY ALGORITHM(4.5). ON ENTRY...B,C HOLD THE
C     DIAGONAL AND SUPER-DIAGONAL VECTORS OF THE INPUT
C     MATRIX RESPECTIVELY: RHS IS THE RIGHT-HAND SIDE
C     VECTOR; Y AND ALPHA ARE WORKING SPACE VECTORS OF
C     LENGTH NR=N+1 AND N RESPECTIVELY. ON EXIT THE SO-
C     LUTION VECTOR OVERWRITES RHS
      DIMENSION  B(N),C(N), ALPHA(N), RHS(N),  Y(NR)
       DO  10  I = 1, N-1
       V =  B(I) + SQRT( B(I)*B(I) -4*C(I)*C(I))
       ALPHA(I) = -2.0 * C(I) /V
       BETA  = (1.0 + ALPHA(I)*ALPHA(I))/ B(I)
       RHS(I)= BETA* RHS(I)
   10  CONTINUE
       ALPHA(N)= SQRT(-B(N)*ALPHA(N-1)/C(N-1)-1.0)
      RHS(N)= (1+ ALPHA(N)* ALPHA(N)) *  RHS(N) / B(N)
C     COMPUTE X(N) = XN
       PRODALPHA=1.0
      S0=1.0
       DO 20  I=1,N-1
   20  PRODALPHA = PRODALPHA* ALPHA(I)
       XN = 0.0
       DO 25  I = 1, N-1
       XN = XN + PRODALPHA * S0 * RHS(I)
       S1 = ALPHA(I) * ALPHA(I) * S0  + 1.0
       PRODALPHA = PRODALPHA/ ALPHA(I)
       S0 = S1
   25  CONTINUE
       XN= XN + PRODALPHA*S0*  RHS(N)
       S1 = ALPHA(N)* ALPHA(N)*S0 +1.0
       XN =XN/S1
       Y(N+1) = -ALPHA(N) * XN
       DO 30  I= 1, N
          II = N-I+1
   30    Y(II)= RHS(II) + ALPHA(II)* Y(II+1)
C     USE RHS TO STORE SOLUTION  VECTOR
      RHS(1) = Y(1)
      DO 35 I= 2, N-1
   35 RHS(I)= Y(I)+ ALPHA(I-1)* RHS(I-1)
      RHS(N) = XN
      RETURN
      END
```

```
C          ***********          PROGRAM      9          ***********
      SUBROUTINE SSRET(N,NR,B,C,RHS,Y)
C     THIS SUBROUTINE SOLVES THE SKEW-SYMMETRIC TRIDIA-
C     GONAL MATRIX OF ORDER N(WHERE THE COEFF MATRIX
C     IS OF THE FORM M[-C,B,C], B AND C BEING CONSTAN-
C     TS) BY THE REVERSED REVERSED RECTANGULAR FACTOR-
C     ISATION METHOD OF ALGORITHM(4.7). ON ENTRY...RHS
C     IS THE RIGHT-HAND SIDE VECTOR; Y IS AN AUXILIARY
C     VECTOR OF ORDER NR=N+1. ON EXIT THE SOLUTION VEC-
C     TOR OVERWRITES RHS.
      DIMENSION RHS(N), Y(NR)
      V = B + SQRT(B*B + 4*C*C)
      ALPHA = 2* C/V
      BETA  =(1.0 - ALPHA* ALPHA) /B
       DO 10 I =1,N
   10  RHS(I) = BETA * RHS(I)
C     COMPUTE  X(N) = XN
      S=0.
      SS=0.
      J=1
      IF(N-N/2*2.EQ.0)J=-1
      DO  15  I=1,N
      II=N-I+1
      J=-J
      S=S+ALPHA*RHS(I)
   15 SS=SS+ALPHA+J*RHS(II)
      S1=S-ALPHA**(N+1)*SS
      S2=1.0-(-ALPHA)**(2*N+2)
      XN=S1/S2
      Y(NR) = -ALPHA * XN
      DO 30  I =1,N
      IT =N-I+1
   30 Y(IT)= RHS(IT)- ALPHA* Y(IT+1)
C     USE RHS TO STORE SOLUTION  VECTOR
      RHS(1) = Y(1)
      DO 40 I = 2,N-1
   40 RHS(I)= Y(I) +ALPHA* RHS(I-1)
      RHS(N)= XN
      RETURN
      END


C          ***********          PROGRAM     10          ***********
      SUBROUTINE TRPP(N,A,B,C,RHS)
C     THIS SUBROUTINE SOLVES A GENERAL SYMMETRIC TRID.
C     MATRIX EQUATION OF ORDER N BY THE R.P.P METHOD
C     AS GIVEN BY ALGORITHM(4.8). ON ENTRY...A,B,C  ARE
C     THE SUB-DIAGONAL, DIAGONAL AND SUPER-DIAGONAL
C     ENTRIES OF COEFF MATRIX RESPECTIVELY; RHS IS THE
C     RIGHT HAND SIDE VECTOR. BUT ON EXIT IT IS OVER-
C     WRITTEN BY THE SOLUTION VECTOR.
      DIMENSION A(N),B(N),C(N),RHS(N)

C     PERFORM RECURSIVE PARTIONING PROCESS.
      NH = N/2
      IFLAG =1
      IF(N-N/2*2.EQ.0)GOTO 5
      NH =(N+1)/2
      IFLAG=0
```

```
    5 DO 10 I=2,NH
      IF(I.EQ.NH.AND.IFLAG.EQ.0)GOTO 10
      K=N-I+1
      B(I)=B(I)-A(I-1)*C(I-1)/B(I-1)
      B(K)=B(K)-A(K)*C(K)/B(K+1)
      RHS(I)= RHS(I)-A(I-1)*RHS(I-1)/B(I-1)
      RHS(K)= RHS(K)-C(K)*RHS(K+1)/B(K+1)
   10 CONTINUE
      IF(IFLAG.EQ.1)GOTO 20
C     FOR N ODD ONLY
      K=N-NH+1
      B(NH)=B(NH)-A(NH-1)*C(NH-1)/B(NH-1)-C(K)*A(NH+1)
     1/B(K+1)
      RHS(NH)=RHS(NH)-A(NH-1)*RHS(NH-1)/B(NH-1)-C(K)*
     1RHS(K+1)/B(K+1)
      RHS(NH)= RHS(NH)/B(NH)
      GOTO 30
   20 K=N-NH+1
      V=B(K)*B(NH)-A(K-1)*C(NH)
      R1=(RHS(NH)*B(K)-RHS(K)*C(NH))/V
      R2=(RHS(K)*B(NH)-RHS(NH)*A(K-1))/V
      RHS(NH)=R1
      RHS(K)=R2
C     PERFORM BACK-SUBSTITUTION PROCESS.
   30 DO 40 I=1,NH-1
      IJ=NH-I
      KJ=N-IJ+1
      RHS(IJ)=(RHS(IJ)-C(IJ)*RHS(IJ+1))/B(IJ)
   40 RHS(KJ)=(RHS(KJ)-A(KJ-1)*RHS(KJ-1))/B(KJ)
      RETURN
      END



C       ***********        PROGRAM    11        ************
C     SUBROUTINE GQRPP(N,B,A,C,RHS,X)
C     THIS SUBROUTINE SOLVES A GENERAL SYMMETRIC QUIND-
C     IAGONAL MATRIX SYSTEM OF ORDER N BY THE RECURSIVE
C     POINT PARTITIONING METHOD (GQRPP) GIVEN AS ALGOR-
C     ITHM(4.11). ON ENTRY... B,A,C HOLD THE DIAGONAL,
C     SUB-DIAGONAL AND SECOND SUB-DIAGONAL ENTRIES OF
C     THE COEFF MATRIX WITH C(N),C(N-1),A(N) = 0. RHS
C     IS THE RIGHT HAND SIDE VECTOR. ON EXIT X HOLDS
C     THE SOLUTION VECTOR BUT THIS CAN ALSO OVERWRITE
C     RHS IF DESIRED, THUS SAVING THE VECTOR X.
      DIMENSION B(N),A(N),C(N),RHS(N),X(N)
      IF((N -N/2*2).EQ. 0 ) GOTO  14
      IF((N- N/2*2).EQ.1 ) NH =(N+1)/2
      IFLAG =0
      GOTO 16
   14 NH =N/2
      IFLAG  =1
   16 CONTINUE
      PERFORM THE RECURSIVE POINT PARTIONING STAGE.
      DO 20 I =2, NH
      IF(I.EQ.NH .AND. IFLAG.EQ.0)GOTO 20
      K= N-I+1
      B(I)=B(I)-A(I-1)* A(I-1)/B(I-1)
      B(I+1) = B(I+1)- C(I-1)* C(I-1)/B(I-1)
      B(K) = B(K)- A(K)* A(K)/B(K+1)
```

```
      B(K-1)= B(K-1)- C(K-1)* C(K-1)/B(K+1)
      A(I)  = A(I)- A(I-1)*C(I-1)/B(I-1)
      A(K-1) = A(K-1) - A(K)* C(K-1)/ B(K+1)
      RHS(I) = RHS(I) - A(I-1)*RHS(I-1)/ B(I-1)
      RHS(I+1)= RHS(I+1) - C(I-1)* RHS(I-1)/ B(I-1)
      RHS(K)  = RHS(K) - A(K)* RHS(K+1)/B(K+1)
      RHS(K-1)= RHS(K-1) -C(K-1)* RHS(K+1)/B(K+1)
   20 CONTINUE
      IF(IFLAG.EQ.1)GOTO 30
C     N IS ODD
      K= N -NH+1
       FAC1= A(NH-1)/B(NH-1)
      FAC2=C(N-NH)/B(NH-1)
      FAC3=(A(K)-FAC2*A(NH-1))/B(NH)-FAC1*A(NH-1))
      X(K+1)=(RHS(K+1)-FAC2*RHS(NH-1)-FAC3*(RHS(NH)-
     1FAC1*RHS(NH-1)))/(B(K+1)-FAC2*C(NH-1)-FAC3*(A(NH)
     2-FAC1*C(NH-1)))
       X(NH)=(RHS(NH)-FAC1*RHS(NH-1)-(A(NH)-FAC1*C(NH-1)
     1)*X(K+1))/(B(NH)-FAC1*A(NH-1))
       X(NH-1)=(RHS(NH-1)-A(NH-1)*X(NH)-C(NH-1)*X(K+1))
     1/B(NH-1)
        GOTO  33
C     N IS EVEN
   30  K =N -NH +1
      V = B(K)* B(NH) -A(NH)* A(N-NH)
      IF(V.EQ. 0.0)V= 2.0E -37
      X(NH)= (RHS(NH)*B(K) -RHS(K)* A(NH))/V
      X(K)=( B(NH)*RHS(K)- A(N-NH)*RHS(NH) )/V
   33 CONTINUE
      PERFORM THE BACK-SUBSTITUTION PROCESS.
      DO 35 I =1,NH-1
         IJ = NH-1-I+1
         KJ = N -IJ+1
       IF(    IJ.EQ.NH-1.AND.IFLAG.EQ.0) GOTO 35
      X(IJ)=(RHS(IJ)-A(IJ)*X(IJ+1)-C(IJ)*X(IJ+2))/B(IJ)
      X(KJ)=(RHS(KJ)-A(KJ-1)*X(KJ-1)-C(KJ-2)*X(KJ-2))
      /B(KJ)
   35 CONTINUE
       RETURN
       END


C        ***********          PROGRAM     12          ***********
      SUBROUTINE PARTRPP(N,NH,B,A,C,RHS,X)
C     THIS SUBROUTINE IS A PARALLEL IMPLEMENTATION OF
C     THE R.P.P. ALGORITHM(4.8) FOR THE SOLUTION OF
C     A TRIDIAGONAL  MATRIX. THIS ROUTINE IS WRITTEN
C     IN FORTRAN BUT WITH THE ADDITION OF PARALLEL CON-
C     STRUCTS, FORK, JOIN, PUTRES AND GETRES IN ORDER
C     TO EFFECT PARALLEL CONTROLS AS IMPLEMENTED IN THE
C     LOUGHBOROUGH UNIV DUAL PROCESSOR INTER-DATA 70
C     MACHINE. N IS THE ORDER OF MATRIX. (N IS EVEN),
C     NH=N/2. A,B,C HOLD THE SUB-DIAGONAL, DIAGONAL AND
C     SUPER-DIAGONAL ELEMENTS AND RHS HOLDS THE RIGHT
C     HAND SIDE VECTORS. ON EXIT,X HOLDS THE SOLUTION
C     VECTOR.
      COMMON/CB/N,NH,B(650),A(650),C(650),RHS(650),
     1X(650)
      $USEPAR
```

```
C       PROGRAM 'FORKS' TO PERFORM THE RECURSIVE PARTITI-
C       ONING PROCESS IN PARALLEL.
        $FORK 5,15;20
      5 DO 10 I=2,NH
        IF(I.EQ.NH)CALL GETRES(1)
        B(I)=B(I)-A(I-1)*C(I-1)/B(I-1)
     10 RHS(I)=RHS(I)-A(I-1)*RHS(I-1)/B(I-1)
        CALL PUTRES(1)
        GOTO 20
     15 DO 18 I=2,NH
        IF(I.EQ.NH) CALL GETRES(1)
        K=N-I+1
        B(K)=B(K)-A(K)*C(K)/B(K+1)
     18 RHS(K)=RHS(K)-C(K)*RHS(K+1)/B(K+1)
        CALL PUTRES(1)
     20 $JOIN
C       OBTAIN THE CENTRAL VALUES OF SOLUTION VECTOR,
C       USING ONLY ONE PROCESSOR.
        K=N-NH+1
        V=B(K)*B(NH)-A(K-1)*C(NH)
        X(NH)=(RHS(NH)*B(K)-RHS(K)*C(NH))/V
        X(K)=(B(NH*RHS(K)-A(K-1)*RHS(NH))/V
C       PROGRAM 'FORKS' TO OBTAIN ELEMENTS OF THE SOLUT-
C       ION VECTOR IN PARARELL.
        $FORK 25,28;30
     25 NHM=NH-1
        DO 26 I=1,NHM
        IJ=NH-I
     26 X(IJ)=(RHS(IJ)-C(IJ)*X(IJ+1))/B(IJ)
        GOTO 30
     28 NHM=NH-1
        DO 29 I=1,NHM
        IJ=NH-I
        KJ=N-IJ+1
     29 X(KJ)=(RHS(KJ)-A(KJ-1)*X(KJ-1))/B(KJ)
     30 $JOIN
        RETURN
        END



C         ***********          PROGRAM     13          ************
        SUBROUTINE PARGORPP(N,NH,B,A,C,RHS,X)
C       THIS SUBROUTINE GIVES A PARALLEL IMPLEMENTATION
C       OF THE GORPP ALGORITHM(4.11) FOR THE SOLUTION OF
C       A QUINDIAGONAL MATRIX SYSTEM. THE SUBROUTINE IS
C       WRITTEN IN FORTRAN BUT WITH PARALLEL CONSTRUCTS;
C       FORK, JOIN, GETRES AND PUTRES INCLUDED SO AS TO
C       EFFECT PARALLEL CONTROLS AS IMPLEMENTED IN THE
C       LOUGHBOROUGH UNIV. INTER-DATA MACHINE. N IS THE
C       ORDER OF THE MATRIX (N IS EVEN HERE), AND NH=N/2;
C       B,A,C HOLD THE DIAGONAL, SUB-DIAGONAL AND SECOND
C       SUB-DIAGONAL ELEMENTS OF THE QUINDIAGONAL MATRIX
C       RESPECTIVELY; RHS HOLDS THE INPUT RIGHT-HAND SIDE
C       VECTOR. ON EXIT, X CONTAINS THE SOLUTION VECTOR:
        COMMON/CB/N,NH,B(650),A(650),C(650),RHS(650),
       1X(650)
        $USEPAR
C       PROGRAM 'FORKS' TO PERFORM THE RECURSIVE PARTITI-
C       ONING PROCESS IN PARALLEL.
```

```
      $FORK 5,15;20
    5 DO 10 I=2,NH
      IF(I.EQ.NH) CALL GETRES(1)
      B(I)=B(I)-A(I-1)*A(I-1)/B(I-1)
      B(I+1)=B(I+1)-C(I-1)*C(I-1)/B(I-1)
      A(I)=A(I)-A(I-1)*C(I-1)/B(I-1)
      RHS(I)=RHS(I)-A(I-1)*RHS(I-1)/B(I-1)
   10 RHS(I+1)=RHS(I+1)-C(I-1)*RHS(I-1)/B(I-1)
      CALL PUTRES(1)
      GOTO 20
   15 DO 18 I=2,NH
      $FORK 25,28;30
   25 NHM=NH-1
      IF(I.EQ.NH) CALL GETRES(1)
      K=N-I+1
      B(K)=B(K)-A(K)*A(K)/B(K+1)
      B(K-1)=B(K-1)-C(K-1)*C(K-1)/B(K+1)
      RHS(K)=RHS(K)-A(K)*RHS(K+1)/B(K+1)
   18 RHS(K-1)=RHS(K-1)-C(K-1)*RHS(K+1)/B(K+1)
      CALL PUTRES(1)
   20 $JOIN
C     OBTAIN THE CENTRAL VALUES OF SOLUTION VECTOR,
C     USING ONLY ONE PROCESSOR.
      NHN=N-NH
      K=N-NH+1
      V=B(K)*B(NH)-A(NH)*A(NHN)
      X(NH)=(RHS(NH)*B(K)-RHS(K)*A(NH))/V
      X(K)=(RHS(K)*B(NH)-A(NHN)*RHS(NH))/V
C     PROGRAM 'FORKS TO OBTAIN ELEMENTS OF THE SOLUTION
C     VECTOR IN PARARELL.
      DO 26 I=1,NHM
      IJ=NH-1-I+1
   26 X(IJ)=(RHS(IJ)-A(IJ)*X(IJ+1)-C(IJ)*X(IJ+2))/B(IJ)
      GOTO 30
   28 NHM =NH-1
      DO 29 I=1,NHM
      IJ=NH-I
      KJ=N-IJ+1
   29 X(KJ)=(RHS(KJ)-A(KJ-1)*X(KJ-1)-C(KJ-2)*X(KJ-2))/
     1B(KJ)
   30 $JOIN
      RETURN
      END
```

```
C        ***********        PROGRAM    14        ***********
         SUBROUTINE BKFACT1(NN,N,M,H,ALP,TOI,EVB,EVL,D,BL,
        1EVU,YM)
C        THIS SUBROUTINE SOLVES THE BLOCK PERIODIC TRIDIA-
C        GONAL MATRIX SYSTEM M(I,B,I) OF ORDER M WHERE B
C        IS A PERIODIC TRIDIAGONAL SUB-MATRIX OF ORDER N
C        OF THE FORM B(ALP,-2(1+ALP+TOI),ALP); I IS THE
C        IDENTITY MATRIX OF ORDER N. THE SUBROUTINE GIVES A
C        DIRECT FAST SOLUTION OF A DISCRETISED SECOND ORD-
C        ER ELLIPTIC (HELMHOLTZ'S) P.D.E IN A RECTANGLE.
C        ON ENTRY...NN IS AN INTEGER INPUT SET EQUAL TO
C        THE ROW DIMENSION OF D,BL,H AS SPECIFIED IN THE
C        CALLING PROGRAM. ALP IS A CONSTANT DENOTING THE
C        SQUARE OF MESH RATIO,TOI IS A POSITIVE CONSTANT;
C        EVB,EVL ARE VECTORS HOLDING THE EIGENVALUES OF B
C        AND (B-SQRT(B**2-4I))/2 RESPECTIVELY. D IS AN NXN
C        MATRIX WHOSE COLUMNS ARE THE CORRESPONDING EIGEN-
C        VECTORS OF B. THE M-COLUMNS OF H HOLD THE M RIGHT
C        HAND SIDE SUB-VECTORS CORRESPONDING TO EACH SUB-
C        BLOCK OF COEFF MATRIX. BL,EVU,YM ARE WORKING SPA-
C        CE VECTORS. ON EXIT, THE SOLUTION VECTOR IS OVER-
C        WRITTEN ON H.
         DIMENSION EVB(N),EVL(N),D(NN,N),BL(NN,N),H(NN,M),
        1EVU(N),YM(N)
         PHI =3.141592654
C        OBTAIN THE EIGENVALUES AND EIGENVECTORS OF B AND
C        (B-SQRT(B**2-4*I))/2
         IF(N-N/2*2.EQ.0)IFLG=1
         IF(N-N/2*2.EQ.1)IFLG=0
         IF(IFLG.EQ.1)NH=N/2 +1
         IF(IFLG.EQ.0)NH=(N+1)/2
         DO 20 J=1,NH
         JJ=2*(J-1)
         JJ1=JJ+1
         IF(J.GT.1)GOTO 8
         EVB(1)= -2.0*(1.0+ALP+TOI )+2*ALP
         SQ1=  SQRT(1.0/N)
         DO 5 I=1,N
       5 D(I,1)= SQ1
         GOTO 20
       8 IF(IFLG.EQ.1.AND.J.EQ.NH)GOTO 16
         EVB(JJ)=-2.0*(1.0+ALP+TOI )+2*ALP*COS(JJ*PHI/N)
         EVB(JJ1)= EVB(JJ)
         SQ2= SQRT(2.0/N)
         DO 9  I=1,N
         D(I,JJ)= SQ2*COS(JJ*I*PHI/N)
       9 D(I,JJ1)=SQ2*SIN(JJ*I*PHI/N)
         GOTO 20
      16 EVB(JJ)=-2.0*(1.0+ALP+TOI )-2*ALP
         KS=1
         DO 18 I=1,N
         KS=-KS
      18 D(I,JJ)=KS*SQ1
      20 CONTINUE
         DO 21 I=1,N
      21 EVL(I)=0.5*(EVB(I)-SQRT(EVB(I)*EVB(I)-4.0))
C        COMPUTE BL= D* EVL* (TRANSPOSE OF D).
         DO 30  I =1,N
         DO 30  J =1,N
          S=0.0
```

```
      DO 24 K=1,N
   24 S=S+EVL(K)*D(I,K)*D(     J,K)
   30 BL(I,J)=S
C     COMPUTE MODIFIED R.H.S SUB-VECTORS AS GIVEN IN
C     (5.3.10).
      DO 60 J =1,M
       IF(J.EQ.1) GOTO 60
       DO 55 I=1,N
       S=0.0
       DO 52 K=1,N
   52  S =S + BL(I,K)* H(K,J-1)
   55  EVU(I)=S
       DO 58 I=1,N
   58  H(I,J)= H(I,J)-EVU(I)
   60  CONTINUE
       ISIGN =1
       IFL =1
C     STORE LAST COL OF H IN EVU USED AS TEMP STORE.
   94  DO 95 I =1,N
   95  EVU(I)=H(I,M)
C     COMPUTE INTERMEDIATE SOL.VECTOR AS IN (5.3.11).
   70  CONTINUE
       IF(N-N/2*2.EQ.0)ISIGN=-1
C     FIRST SOLVE FOR LAST INTERMEDIATE VECTOR Y(M)
       DO 73 I=1,N
       S=0.0
       DO 72 K=1,N
   72  S=S+D(K,I)*(1.0/(1.0+ISIGN*EVL(I)**M))*EVU(K)
   73  EVB(I)=S
        DO 82 I=1,N
       S=0.
       DO 76 K=1,N
   76  S=S+D(I,K)*EVB(K)
       YM(I)=S
       IF(IFL.EQ.2)GOTO 82
   82  H(I,M)=YM(I)
      GOTO(122,190) IFL
  122 CONTINUE
      DO 125 I=1,N
  125 EVB(I) =YM(I)
C     NEXT, COMPUTE INTERMEDIATE VECTORS Y(I),I=M-1...1
C     AS GIVEN IN (5.3.11).
      DO 137 I=1,N
      S=0.0
      DO 135 K =1,N
  135 S=S+D(K,I)*EVB(K)
  137 EVU(I)=S
      DO 150  KKK =1,M-1
       JSIGN=1
       IF(KKK-KKK/2*2.EQ.0) JSIGN=-1
       DO 139 I=1,N
       S=0.
       DO 140     K=1,N
  140  S=S+D(I,K)*(JSIGN*EVL(K)**KKK)*EVU(K)
       EVB(I)=S
  145  H(I,KKK)=H(I,KKK)-EVB(I)
  139  CONTINUE
  150 CONTINUE
C     COMPUTE INTERMEDIATE VECTOR G(M) AS IN (5.3.12)
      DO 160 I =1,N
```

```
      S=0.
      DO 155 K =1,N
  155 S =S+ BL(I,K)*YM(K)
  160 EVU(I)=S
      DO 161 I=1,N
  161 H(I,M)=EVU(I)
C     COMPUTE INTERMEDIATE VECTORS G(I),I=M-1...1 AS IN
C     (5.3.12).
      DO 180 K =1,M-1
       II =M-K
       DO 170 I=1,N
       S=0.
       SS=0.
       DO 165 J=1,N
       S=S+BL(I,J)*EVU(J)
  165  SS=SS+BL(I,J)*H(J,II)
  170  EVB(I)=SS-S
       DO 175 I=1,N
       H(I,II)=EVB(I)
  175  EVU(I)=EVB(I)
  180  CONTINUE
C     COMPUTE SOLUTION SUB-VECTOR U(1) AND STORE IN COL.
C     M OF H.
      IFL =2
    1  ISIGN=1
       GOTO 70
  190 CONTINUE
C     COMPUTE SOLUTION SUB-VECTORS U(I),I=M-1...1, AND
C     STORE IN COL. I OF H.
      DO 210 I=1,N
  210 EVB(I)=YM(I)
      DO 225 I =1,N
      S=0.
      DO 220 K=1,N
  220 S=S+D(K,I)*EVB(K)
  225 EVU(I) = S
      DO 240 KKK =2, M
       KSIGN=1
       M1=M-M/2*2
       KKKK=M-KKK+1
       K1= KKK-KKK/2*2
       IF(M1.EQ.1.AND.K1.EQ.0)KSIGN=-1
       IF(M1.EQ.0.AND.K1.EQ.1)KSIGN=-1
      DO  227 I=1, N
       S=0.0
       DO 230  K=1,N
  230  S=S+D(I,K)*KSIGN*EVL(K)**KKKK*EVU(K)
  227  H(I,KKK)=H(I,KKK)-S
  240  CONTINUE
       RETURN
       END
```

```
C        **********        PROGRAM    15        **********
      SUBROUTINE BKFACT4(NN,N,M,H,ALP,EVB,EVL,D,BL,EVU,
     1YM)
C      THIS SUBROUTINE SOLVES A BLOCK SKEW-SYMMETRIC
C      PERIODIC TRIDIAGONAL MATRIX SYSTEM A(-I,B,I) OF
C      ORDER M WHERE B IS A TRIDIAGONAL SUB-MATRIX OF
C      ORDER N AND OF THE FORM B(-2*ALP, 4*ALP,-2*ALP)
C      AND I IS THE IDENTITY MATRIX OF ORDER N. THIS
C      GIVES A DIRECT FAST SOLUTION OF A DISCRETISED ONE
C      DIMENSIONAL HEAT CONDUCTION EQUATION WITH PERIOD-
C      IC BOUNDARY CONDITION IN A RECTANGLE SOLVED BY
C      BOUNDARY-VALUE TECHNIQUES. ON ENTRY...NN IS AN
C      INTEGER SET EQUAL TO THE ROW DIMENSION OF D,BL,H
C      AS SPECIFIED IN THE CALLING PROGRAM. ALP IS THE
C      SQUARE OF MESH RATIO. EVB,EVL ARE VECTORS HOLDING
C      THE EIGENVALUES OF B AND 0.5(-B+SQRT(B**2+4I))
C      RESPECTIVELY, D IS AN (NXN) MATRIX WHOSE COLUMNS
C      ARE THE CORRESPONDING EIGENVECTORS OF B. THE M-
C      COLUMNS OF H HOLD THE M-RIGHT-HAND SIDE SUB-VECT-
C      ORS CORRESPONDING TO EACH SUB-BLOCK OF COEFF MAT-
C      IX. BL,EVU,YM ARE WORKING SPACES. ON EXIT, THE
C      SOLUTION IS OVER WRITTEN ON H.
      DIMENSION EVB(N),EVL(N),D(NN,N),BL(NN,N),H(NN,M),
     1EVU(N),YM(N)
       PHI=3 .141592654
C      DETERMINE THE EIGENVALUES AND EIGENVECTORS OF B
C      AND (-B+SQRT(B**2+4I))/2.
       DO 20 J=1,N
      EVB(J)=4*ALP*(1.0-COS(PHI*J/(N+1)))
       EVL(J)=0.5*(-EVB(J)+SQRT(EVB(J)*EVB(J)+4.0))
       S=0.0
       DO  10 I=1,N
       D(I,J)= SQRT(2.0/(N+1))*SIN(I*J*PHI/(N+1))
   10  S=S+D(I,J)*D(I,J)
       SS=1.0/SQRT(S)
       DO 15 I=1,N
   15  D(I,J)=D(I,J)*SS
   20  CONTINUE
      COMPUTE BL=D*EVL*(TRANSPOSE OF D).
       DO 30 I=1,N
       DO 30 J=1,N
       S=0.
       DO 24 K=1,N
   24  S=S+EVL(K)*D(I,K)*D(J,K)
   30  BL(I,J)=S
      COMPUTE MODIFIED R.H.S SUB-VECTORS AS GIVEN IN
      (5.4.13).
       J=1
       DO 60 J=1,M
       IF(J.EQ.1) GOTO 60
       DO 55 I=1,N
       S=0.
       DO 52 K=1,N
   52  S =S + BL(I,K)* H(K,J-1)
   55  EVU(I)=S
       DO 58 I=1,N
   58  H(I,J)=H(I,J)*EVU(I)
   60  CONTINUE
       ISIGN =1
       IFL =1
```

```
C     STORE LAST COL OF H IN EVU USED AS TEMP STORE.
   95  EVU(I)=H(I,M)
   94  DO 95 I =1,N
   95 EVU(I) = H(I,M)
   70  CONTINUE
       COMPUTE INTERMEDIATE SOLUTION VECTORS Y(M) AS IN
C      (5.4.14)
       DO 73 I=1,N
       S=0.
       DO 72 K=1,N
   72  S=S+D(K,I)*(1.0/(1.0-ISIGN*EVL(I)**M))*EVU(K)
   73  EVB(I)=S
       DO 82 I=1,N
       S=0.
       DO 76 K=1,N
   76  S=S+D(I,K)*EVB(K)
       YM(I)=S
       IF(IFL.EQ.2) GOTO 82
   82  H(I,M)=YM(I)
  120 CONTINUE
       GOTO(122,190) IFL
  122 CONTINUE
C      COMPUTE OTHER INTERMEDIATE VECTORS Y(I),I=M-1...1
C      AS IN (5.4.14).
       DO 125 I=1,N
  125 EVB(I) =YM(I)
       DO 137 I=1,N
       S=0.0
       DO 135 K =1,N
  135 S=S+D(K,I)*EVB(K)
  137 EVU(I)=S
       DO 150  KKK =1,M-1
       DO 139 I =1,N
       S=0.
       DO 140 K=1,N
        DO 140     K=1,N
  140 S=S+D(I,K)*(-EVL(K)**KKK)*EVU(K)
  139 H(I,KKK)=H(I,KKK)-S
  150 CONTINUE
C      COMPUTE INTERMEDIATE VECTOR G(M) AS IN (5.4.15)
       DO 160 I =1,N
       S=0.
        S=0.
       DO 155 K =1,N
  155 S =S+ BL(I,K)*YM(K)
  160 EVU(I)=S
       DO 161 I=1,N
  161 H(I,M)=EVU(I)
C      COMPUTE INTERMEDIATE VECTORS G(I),I=M-1...1 AS IN
C      (5.4.15).
       DO 180 K =1,M-1
       II=M-K
       DO 170  I =1,N
       S=0.
       SS=0.
       DO 165 J=1,N
        S=S+BL(I,J)*EVU(J)
  165   SS=SS+BL(I,J)*H(J,II)
  170   EVB(I)=SS-S
        DO 175 I=1,N
```

```
      H(I,II)=EVB(I)
  175 EVU(I)= EVB(I)
  180 CONTINUE
C     COMPUTE SOLUTION SUB-VECTOR U(I),I=M-1...1 AND
C     STORE IN COLUMN M OF H.
      IFL=2
      ISIGN=1
      IF((M-M/2*2.EQ.1))ISIGN=-1
      GOTO 70
  190 CONTINUE
      IJ=1
C     STEP 4: COMPUTE THE SOLUTION VECTORS.
      DO 210 I=1,N
  210 EVB(I)=YM(I)
      DO 225 I =1,N
      S=0.
      DO 220 K=1,N
  220 S=S+D(K,I)*EVB(K)
  225 EVU(I) = S
C     COMPUTE OTHER SOLUTION SUB-VECTORS U(I),I=M-1,...,1
C     AND STORE IN COLUMN I OF H.
      DO 240 KKK =2, M
      KKKK=M-KKK+1
      KSIGN=1
      M1=M-M/2*2
      K1= KKK-KKK/2*2
      IF(M1.EQ.1.AND.K1.EQ.0)KSIGN=-1
      IF(M1.EQ.0.AND.K1.EQ.1)KSIGN=-1
      DO 227 I=1,N
      S=0.0
      DO 230  K=1,N
  230 S=S+D(I,K)*KSIGN*EVL(K)**KKKK*EVU(K)
      H(I,KKK)=H(I,KKK)-S
  227 CONTINUE
  240 CONTINUE
      RETURN
      END


C       ***********        PROGRAM    16        ************
      SUBROUTINE BLOMC(NN,N,M,H,BL,D,EVU,EVB,EVL,YM,ALF
     1A)                  SOLVES
C     THIS SUBROUTINE THE BLOCK CYCLIC LOWER TRIANGULAR
C     MATRIX EQUATION OF THE FORM M[-1,B,0] OF ORDER M
C     WHERE B IS A TRIDIAGONAL MATRIX OF ORDER N AND OF
C     THE FORM B[-ALFA,1.0+2*ALFA,-ALFA]; ALFA= YK/(XH*
C     XH) AND XH,YK ARE MESH SIZES ALONG X- AND T-AXES
C     RESPECTIVELY. THIS SYSTEM IS DERIVED FROM THE
C     IMPLICIT FINITE DIFFERENCE APPROX. OF 1-DIMENSIO-
C     NAL HEAT EQUATION WITH PERIODIC CONDITION ALONG
C     THE X-AXIS. ON ENTRY...THE VECTOR EVB HOLDS THE
C     EIGENVALUES OF B; AND D IS THE SUB-MATRIX OF EIG-
C     VECTORS OF B. H IS AN NXM SUB-MATRIX HOLDING THE
C     M RIGHT HAND SUB-VECTORS CORRESPONING TO EACH BL-
C     OCK ELEMENT OF COEFF MATRIX. BL,EVU,EVL,YM ARE
C     WORKING SPACES. NN IS AN INTEGER SET EQUAL TO THE
C     ROW DIMENSION OF D,BL,H IN THE CALLING PROGRAM.
C     ON EXIT, THE SOLUTION VECTOR IS OVERWRITTEN ON H.
      DIMENSION H(NN,M),D(NN,M),BL(NN,M),EVU(N),EVB(N),
```

```
     1YM(N),EVL(N)
       PHI=3.141592654
C      OBTAIN EIGENSYSTEM OF TRIDIAGONAL MATRIX B.
       DO 20 J =1,N
       EVB(J)=1.0+2*ALFA-2*ALFA*COS(PHI*J/(N+1))
       S=0.
       DO 10  I = 1, N
   10  D(I,J)=SQRT(2.0/(N+1))*SIN(I*J*PHI/(N+1))
   20  CONTINUE
C      TRANSFORM THE SYSTEM M[=I,B,0] TO THE FORM
C      M[=R,I,0] AND MODIFY R.H.S. VECTOR ACCORDINGLY.
       DO 32 J=1,N
       DO 32 I=1,N
       S=0.
       DO 29 K=1,N
   29  S=S+D(I,K)/EVB(K)*D(I,K)
   32  BL(I,J)=S
       DO 34 I=1,N
       DO 34 J=1,M
       S=0.
       DO 33 K=1,N
   33  S=S+BL(I,K)*H(K,J)
   34  H(I,J)=S
C      CALCULATE MODIFIED R.H.S. VECTOR AS GIVEN BY
C      (5.5.15), INTRODUCING A NORMAL FORM OF MATRIX R.
       DO 60 J =1,M
       IF(J.EQ.1)GOTO 60
       DO 55 I =1,N
       S =0.0
       DO 52 K=1,N
   52  S=S+BL(I,K)*H(K,J-1)
   55  EVU(I)=S
       DO 58 I=1,N
   58  H(I,J)=H(I,J)+EVU(I)
   60  CONTINUE
       IFL=1
C      THE LAST COLUMN OF H IS TEMPORARILY STORED IN EVU.
   94  DO 95 I=1,N
   95  EVU(I) = H(I,M)
       DO 73 I=1,N
       S=0.
       DO 72 K=1,N
   72  S=S+D(K,I)/(1.0-1/(EVB(I )**M))*EVU(K)
   73  EVL(I)=S
       DO 82 I=1,N
       S=0.
       DO 76 K=1,N
   76  S=S+D(I,K)*EVL(K)
       YM(I)=S
   82  H(I,M)=S
C      OBTAIN  THE SOLUTION SUB-VECTORS AND STORE THEM IN H
       DO 137 I=1,N
       S=0.0
       DO 135 K =1,N
  135  S=S+D(K,I)*YM(K)
  137  EVU(I)=S
       DO 150  KKK =1,M-1
       DO 139 I =1,N
       S=0.
       DO 140 K=1,N
```

```
  140 S=S+Q(I,K)/(EVB(K)**KKK)*EVU(K)
  139 H(I,KKK)= H(I,KKK)-S
  150 CONTINUE
      RETURN
      END


C         ***********          PROGRAM    17          ***********
      SUBROUTINE BKREF(NN,MM,N,H,E,Q,EVB,EVT,EVTT,R,
     1SBAR,EVE,EVS,ALP,BETA)
C     THIS SUBROUTINE GIVES THE SOLUTION OF AN (MXM)
C     BLOCK TRIDIAGONAL MATRIX SYSTEM, M[C,B,C], FOR M
C     =MM-1,BY THE BLOCK REVERSED TRIANGULAR (RECTANGU-
C     LAR) FACTORISATION AND EXPANSION METHOD AS GIVEN
C     IN SECTION (6.3). THE MATRIX B IS AN (NXN) TRID.
C     MATRIX OF THE FORM B[ALP,-2(1+ALP),ALP] WHERE
C     ALP IS SQUARE OF MESH RATIO; C IS A CONSTANT-TERM
C     DIAGONAL MATRIX WHOSE ELEMENT IS BETA. NN IS AN
C     INTEGER SET EQUAL TO THE ROW DIMENSION OF H,Q,E:
C     H HOLDS THE INPUT R.H.S VECTOR. EVB,Q HOLD THE
C     EIGENVALUES AND CORRESPONDING EIGEN-VECTORS OF B
C     RESPECTIVELY. THE NXN SUB-MATRIX E HOLDS THE MAT-
C     RIX FACTOR -2C/(B+SQRT(B*B-4*C**2). EVE,EVT,EVTT,
C     R,SBAR ARE WORKING SPACE VECTORS. THE SOLUTION
C     VECTOR IS STORED IN H ON EXIT.
      DIMENSION H(NN,MM),Q(NN,N),E(NN,N),EVC(N),EVB(N),
     1EVE(N),EVT(N),EVTT(N),R(N),SBAR(N),EVS(N)
      PHI=3.141592654
      M= MM-1
C     OBTAIN EIGEN-VALUES AND VECTORS OF B
      DO 15 J=1,N
      EVB(J)=-2*(1.0+ALP)+2.0*ALP* COS(J*PHI/(N+1))
      EVE(J)=-2.0*BETA*(1.0/(EVB(J)+SQRT(EVB(J)*EVB(J)-
     14*BETA*BETA)))
      EVT(J)=(1.0+EVE(J)*EVE(J))/EVB(J)
      EVTT(J)=1.0/(1.0-EVE(J)**(2*M+2))
      EVS(J)=EVE(J)**(M+1)
      DO 14 I=1,N
   14 Q(I,J)= SQRT(2.0/(N+1))* SIN( J*PHI*I/(N+1))
   15  CONTINUE
C     COMPUTE MATRIX FACTOR E USING THE NORMAL FORM,
C     E= Q*EVE*Q(TRASPOSE).
      DO 25 I=1,N
      DO 25 J=1,N
      S=0.
      DO 20 K=1,N
   20 S = S+ EVE(K)*Q(K,I)*Q(K,J)
   25 E(I,J)=S
C     COMPUTE INTERMEDIATE QUANTITIES G(KK)=EVT*Q(TRAS-
C     POSE)*H, FOR KK=1...M.
      DO 50 KK =1,M
      DO 35 I=1,N
      S=0.
      DO 30 K=1,N
   30 S=S+Q(K,I)*EVT(I)*H(K,KK)
   35 R(I)=S
      DO 45 I=1,N
      HH(I,KK)=R(I)
   45 H(I,KK)=R(I)
```

```
   50 CONTINUE
C     USE NESTED MULTIPLICATION TO COMPUTE QUANTITIES
C     IN (6.3.32) AND (6.3.34).
      DO 53 I=1,N
      EVB(I)=H(I,1)
   53 EVT(I)=H(I,M)
C     EVT(I) USED AS TEMP STORE
C     EVB(I) USED AS TEMP STORE
      DO 70 KK =1, M-1
      KKK =M-KK
      DO 60 I=1,N
      R(I)=EVE(I)*EVB(I)+H(I,KK+1)
   60 SBAR(I)=EVE(I)*EVT(I)+H(I,KK)
      DO 65 I=1,N
      EVB(I)= R(I)
   65 EVT(I)= SBAR(I)
   70 CONTINUE
C     COMPUTE LAST SOLUTION SUB-VECTOR U(M) AS GIVEN IN
C     EQUATION (6.3.36).
      DO 80 I=1,N
      S=0.
      DO 75 K=1,N
   75 S=S+Q(I,K)*EVTT(K)*(EVB(K)-EVS(K)*EVT(K))
   80 SBAR(I)=S
      DO 124 KK=1,M
      DO 123 I=1,N
      S=0.
      DO 122 K=1,N
  122 S=S+Q(I,K)*H(K,KK)
  123 R(I)=S
      DO 124 I=1,N
  124 H(I,KK)=R(I)
C     DETERMINE INTERMEDIATE VECTOR Y(M+1) AS IN 6.3.17.
      DO 120 I=1,N
      S=0.
      DO 115 K=1,N
  115 S=S+E(I,K)* SBAR(K)
  120 H(I,MM)= -S
C     DETERMINE INTERMEDIATE SOLUTION VECTOR Y(I),I=1...
C     M AS GIVEN IN (6.3.17).
      DO 135 KK =1,M
       KKK=M-KK+1
      DO 130 I=1,N
      S=0.
      DO 125 K=1,N
  125 S=S+E(I,K)*H(K,KKK+1)
  130 H(I,KKK)=H(I,KKK)+S
  135 CONTINUE
C     OBTAIN SOLUTION SUB-VECTORS U(I),I=M-1...1 AS IN
C     (6.3.36).
      DO 150 KK =2,M-1
      DO 145 I=1,N
      S=0.
      DO 140 K=1,N
  140 S=S+E(I,K)* H(K,KK-1)
  145 H(I,KK)=S +H(I,KK)
  150 CONTINUE
      DO 155 I=1,N
  155 H(I,M)=SBAR(I)
      RETURN
```

```
      END


C          ***********          PROGRAM    18          ***********
      SUBROUTINE PQD(N,A,B,L,U,L2,U2,WKSP1,WKSP2,EV,EP)
C     THIS SUBROUTINE COMPUTES ALL THE EIGENVALUES OF A
C     SYMMETRIC POSITIVE DEFINITE, DIAGONALLY DOMINANT
C     PERIODIC TRIDIAGONAL MATRIX OF ORDER N, BY THE
C     PERIODIC QUOTIENT-DIFFERENCE METHOD AS GIVEN IN
C     SECTION (7.2). ON ENTRY...B,A ARE VECTORS HOLDING
C     THE DIAGONAL AND SUB-DIAGONAL ELEMENTS OF THE
C     GIVEN MATRIX. L,U,L2,U2 ARE PAIRS OF WORKING SPACE
C     VECTORS USED IN THE CYCLIC FACTORISATION PROCESS.
C     WKSP1,WKSP2 ARE WORKING SPACE VECTORS. EP IS THE
C     ABSOLUTE ERROR TOLERANCE FOR THE COMPUTE  EIGEN-
C     VALUES. ON EXIT EV HOLDS THE N EIGENVALUES.
      REAL L ,L2
      DIMENSION A(N),B(N),L(N),L2(N),U(N),U2(N),WKSP1(N
     1),WKSP2(N),EV(N)
      PHI=0.5
      NN=N
      KFL=0
C     USING A SIMPLE SIMILARITY TRANSFORMATION CONVERT
C     A SYMMETRIC PERIODIC TRIDIAGONAL MATRIX TO AN
C     ASSYMMETRIC ONE WITH SUPER-DIAGONALS ALL SET TO 1.
      DO 5 I =1,N
    5 A(I)= A(I)*A(I)
C     COMPUTE L(1) BY A CALL TO THE CF2 SUBROUTINE
C     WHICH EVALUTES A PERIODIC CONTINUED FRACTION.
      NF =0
      JFLG=0
       Z=0
      Y=0.
      EPS =1.0 E-12
      CALL CF2(N,B,A,L,U,VL,WKSP1,WKSP2,JFLG, NF,Y,EPS)
      IF(NF.EQ.0) GOTO 15
      WRITE(2,10)
   10 FORMAT(5X,'CF2 FAILS;NO CONVERGENCE IN CONTD FR;')
      RETURN
   15 L(1) = VL
      U(1) = B(1)-L(1)
      DO 20 I=2,N
      L(I) =A(I)/U(I-1)
   20 U(I) =B(I)-L(I)
      ITER =1
C     PERFORM THE S TH STEP OF THE P.Q.D. FOR S=2,3...
   27 K=0
   28 PHII =PHI
      IC=0
      UTEMP= U(N)+L(1)
   30 Y=PHI*UTEMP
C     SET UP A SUITABLE SHIFT BY AN AD HOC METHOD TO
C     ACCELERATE CONVERGENCE OF D.Q.D SCHEME.
      IF(KFL.EQ.1)GOTO 50
   35 NF=0
      JFLG=1
      CALL CF2(N,B,A,L,U,VL,WKSP1,WKSP2,JFLG,NF,Y,EPS)
      IF(NF.EQ.0)GOTO 45
C     REDUCE SIZE OF SHIFT IF CURRENT SHIFT UNSUCCESSFUL.
```

```
      IF(IC.GT.5)GOTO 40
      PHI = PHI* 0.5
      IC =IC+1
      GOTO 30
C     AFTER 5 UNSUCCESSFUL AD-HOC SHIFT TRIALS SET SHIFT
C     TO 0 AND RESET PHI FOR FUTURE SHIFT TRIALS.
   40 Y=0.
      PHI = 0.5*PHII
      GOTO 35
   45 L2(1)= VL
      IF(L2(1).LT.EPS) KFL=1
      GOTO 70
   50 L2(1)= L(1)
   65 Y=PHI*UTEMP
   70 CONTINUE
      U2(1)= L(2)*U(1)-L2(1)-Y
      DO 75 I =2,N
      II=I+1
      IF(II.GT.N)II=1
      L2(I)=U(I)*L(I)/U2(I-1)
   75 U2(I)=L(II)*U(I)-L2(I)-Y
C     TEST TO ENSURE THAT THE POSITIVENESS OF P.Q.D IS
C     PRESERVED.
      DO 80 I=1,N
   80 IF(U2(I).LE.0.0)GOTO 90
C     Z IS USED TO ACCUMULATE THE SHIFTS.
      Z =Z+Y
      IF(Y.NE.0)PHI =(1+PHI)* 0.5
      GOTO 100
C     DROP SHIFT TO ENSURE THAT U IS POSITIVE
   90 Y=0.
      IF(KFL.EQ.0) GOTO 35
      L2(1)=L(1)
      GOTO 70
  100 CONTINUE
      ITER =ITER+1
      IF(K.GT.100) GOTO 160
      IF(L2(N).GT.EP)GOTO 120
  105 EV(N)= Z +U2(N)
C     CURRENT SMALLEST EIGENVALUE IS FOUND. DEFLATE MAT-
C     RIX SYSTEM BY REDUCING ITS SIZE BY 1.
      N=N-1
      IF(N.EQ.0)GOTO 135
      IF(N.EQ.1)GOTO 105
      PHI=0.5
      DO 115 I=1,N
      U(I)=U2(I)
  115 L(I)=L2(I)
      K=K+1
      GOTO 27
  120 DO 125 I=1,N
      U(I)=U2(I)
  125 L(I)=L2(I)
      GOTO 28
  135 N=NN
      RETURN
  160 WRITE(2,161)ITER
  161 FORMAT(10X,'NO CONVERGENCE AFTER',I3,'ITERATIONS'//)
      RETURN
      END
```

```
      SUBROUTINE CF2(N,B,A,L,U,VL,ALP,BETA,JFLG,NF,Y,EPS)
C     THIS SUBROUTINE EVALUATES AN INFINITE PERIODIC
C     CONTINUED FRACTION OF THE FORM IN (7.2.10) OR (7:
C     2:22) OBTAIN IN THE CYCLIC FACTORISATION OF AN
C     ASYEMMETRIC PERIODIC TRID. MATRIX WITH UNIT SUPER-
C     DIAGONALS; THIS BEING REQD. IN THE PQD ALGORITHM;
C     ON ENTRY...B AND A HOLD THE DIAGONAL  AND SUB-DIA-
C     GONAL ELEMENTS OF INPUT MATRIX RESPECTIVELY. ALP,
C     BETA,L,U ARE WORKING SPACE VECTORS. Y IS CURRENT
C     SHIFT APPLIED TO SPEED-UP CONVERGENCE IN THE CALL-
C     ING PROGRAM. EPS IS TRUNCATION ERROR TOLERANCE IN
C     CONTINED FRACTION EVALUATION. NF;JFLG ARE FLAGS
C     TO INDICATE CONVERGENCE OR OTHERWISE;AND TO DIFF-
C     ERENTIATE THE 1ST AND SUBSEQUENT CALLS TO THE CF2
C     ROUTINE RESPECTIVELY. ON EXIT,THE VALUE OF CONT-
C     INUED FRACTION IS HELD IN VL.
      REAL L
      DIMENSION B(N),A(N),L(N),U(N),ALP(N),BETA(N)
      IF(JFLG.EQ.1) GOTO 15
      DO 10 I=1,N
      II=N-I+1
      JJ= N-I+2
      IF(JJ.GT.N)JJ=1
      ALP(I)=A(JJ)
      BETA(I)= B(II)-Y
   10 CONTINUE
      GOTO 20
   15 DO 20 I=1,N
      II=N-I+1
      JJ=N-I+2
      IF(JJ.GT.N)JJ=1
      ALP(I)=L(JJ)* U(JJ)
      BETA(I)=L(JJ)*U(II)-Y
      IF(BETA(I).GT.0) GOTO 20
      NF=1
      RETURN
   20 CONTINUE
C     COMPUTE APPROXIMANTS OF INFINITE CONTINUED FRACT-
      ION BY A FORWARD RECURRENCE RELATION.
      E0=0.
      F0=1.0
      E1=ALP(1)
      F1=BETA(1)
      DO 50 I=2,N
      E2 =BETA(I)*E1- ALP(I)*E0
      F2 =BETA(I)*F1-ALP(I)*F0
      ALP REUSED TO STORE AWAY PARTIAL APPROXIMANTS OF
      CONTINUED FRACTION.
      ALP(I-1)=E1/F1
      ALP(I)=E2/F2
      K=I
      IF(ABS(ALP(I)- ALP(I-1)).LE.EPS)GOTO 55
       IF(I.EQ.N)GOTO 50
      E0 =E1
      E1 =F2
      F0 =F1
      F1 =F2
   50 CONTINUE
```

```
   55 CONTINUE
      SOLVE QUADRATIC EQN; F1*W**2+(F2-E1)*W-E1=0.
      R =F2-E1
      S = 4*F1*E2
      VL= (-R+SQRT(R*R +S))/(2*F1)
      VLL=(-R-SQRT(R*R+S))/(2*F1)
      IF(VL.GT.VLL)GOTO 58
       T=VL
      VL=VLL
      VLL=T
   58 CONTINUE
      IF((VL-VLL).LT.EPS) GOTO 75
      IF(ABS(ALP(K-1)-VLL).LT.ABS(ALP(K-1)-VL))GOTO 60
      GOTO 65
   60 NF=1
      RETURN
   65 DO 70 I=1,K-1
   70 IF(ALP(I).EQ.VLL)GOTO 60
   75 CONTINUE
      RETURN
      END



C       ***********        PROGRAM    19        ***********
      SUBROUTINE SCLH(N,A,B,C,L,U,WKSP1,WKSP2,EP)
C     THIS SUBROUTINE REDUCES A GENERAL PERIODIC TRIDIA-
C     GONAL MATRIX M[A(I),B(I),C(I)] TO A LOWER SPARSE
C     CYCLIC HESSENBERG FORM H[ A'(I),B'(I),1] WHERE
C     A(1) DECREASES MONOTONICALLY UNTIL IT IS .LT. EP,
C     A SMALL ERROR TOLERANCE = 1.0E-08. THE VECTORS B,
C     A,C ARE THE DIAGONAL, SUB-DIAGONAL AND SUPER-DIA-
C     GONAL ELEMENTS OF INPUT MATRIX. L,U,WKSP1,WKSP2
C     ARE WORKING SPACE VECTORS. ON EXIT THE INPUT MAT-
C     RIX IS OF THE FORM M[A'(I),B'(I),1] WITH A'(1).LE.EP.
      REAL L
      DIMENSION A(N),B(N),C(N),L(N),U(N),WKSP1(N),
     1WKSP2(N)
      EPS= 1.0E-10
      SH=0.
      Y=0.
      ITER=0
C     REDUCE PERIODIC TRID. MATRIX M[A,B,C] TO THE FORM
C     M[A',B',1] WITH UNIT SUPER-DIAGONAL ENTRIES.
      A(1)=A(1)*C(N)
      C(1)=1.0
      DO 10 I=2,N
      A(I)=A(I)*C(I-1)
   10 C(I)=1.0
   20 CONTINUE
      NF=0
      JFLG=0
      CALL CF2(N,B,A,L,U,VL,WKSP1,WKSP2,JFLG,NF,Y,EPS)
      IF(NF.EQ.0)GOTO 40
C     IF CONTD. FACTION DOES NOT CONVERGE, ADD A SHIFT
C     TO STRENGHTEN THE DIAGONAL DOMINANCE OF MATRIX.
      Y=0.5* B(1)
      SH= SH+Y
      DO 30 I=1,N
   30 B(I)=B(I)+Y
```

```
      GOTO 20
   40 L(1)= VL
      U(1)= B(1)=L(1)
      DO 45 I=2,N
      L(I)= A(I)/U(I-1)
   45 U(I)= B(I)-L(I)
C     REMULTIPLICATION
      DO 50 I=1,N
      J=I+1
      IF(J.GT.N)J=1
      B(I)=U(I)+L(J)
   50 A(I)=U(I)*L(I)
      ITER=ITER+1
      IF(ITER.GT.400)GOTO 90
      IF(ABS(A(1)).LE.FP)GOTO 60
      GOTO 20
   60 DO 80 I=1,N
   80 B(I)=B(I)-SH
      RETURN
   90 WRITE(2,95) ITER
   95 FORMAT(10X,'NO CONVERGENCE AFTER',I3 'P-Q SIMILAR-
      ITY TRANSFORMATIONS')
      RETURN
      END


C        ***********        PROGRAM    20        ***********
      SUBROUTINE BAIRSTOW(N,A,B,W,WI,EPS,C,D,M,L,E,F)
      THIS SUBROUTINE COMPUTES ALL THE EIGENVALUES OF A
C     SPARSE CYCLIC LOWER HESSENBERG MATRIX OF THE FORM
C     MIA(I),B(I),1) BY BAIRSTOW'S SYNTHETIC DIVISION
C     METHOD WHERE THE INPUT MATRIX IS OBTAINED AFTER
C     P-Q  SIMILARITY TRANSFORMATIONS, USING THE SCLH
C     SUBROUTINE(PROGRAM 19). B(I),A(I) ARE THE DIAGO-
C     NAL AND SUB-DIAGONAL ELEMENTS OF INPUT MATRIX OF
C     ORDER N. C,D,M,L,E,F ARE WORKING SPACE VECTORS.
C     EPS IS ACCURACY DESIRED IN THE DETERMINATION OF
C     QUADRATIC FACTORS. ON EXIT,W AND WI HOLD REAL AND
C     IMAGINARY PARTS OF THE COMPLEX EIGENVALUES.
      REAL M,L,J,K,L0,M0,F0,F0
      DIMENSION A(N),B(N),C(N),D(N),M(N),L(N),E(N),F(N)
     1,W(N),WI(N)
      EPS=1.0E-05
      IFLG=0
      J=1.0
      K=1.0
      C0 =0.0
      D0 =0.0
      DO 10 I=1,N
      C(I)=0.0
   10 D(I)=0.0
      DELTAJ=0.0
      DELTAK =0.0
      I =0
   20 CONTINUE
      IF(I.GT.0)GOTO 25
      L0=0.
      M0=1.0
      E0=0.0
```

```
      F0=0.0
      M(1)=B(1)=C0
      L(1)=-1.0
      E(1)=0.0
      F(1)=0.0
      IF(I.EQ.0)GOTO 30
   25 L(I)=0.0
      M(I)=1.0
      L(I+1)=-1.0
      M(I+1)=B(I+1)=C(I)
      E(I)=0.0
      F(I)=0.0
      E(I+1)=0.0
      F(I+1)=0.0
   30 CONTINUE
      DO 45 IN= I+2,N
      IF ( IN.EQ.2)GOTO 40
      L(IN)=(B(IN)-J)*L(IN-1)-A(IN)*L(IN-2)-M(IN-1)
      M(IN)=B(IN)*M(IN-1)-A(IN)*M(IN-2)-K*L(IN-1)-
     1C(IN-1)
      E(IN)=(B(IN)-J)*E(IN-1)-F(IN-1)-A(IN)*E(IN-2)
      F(IN)=B(IN)*F(IN-1)-K*E(IN-1)-A(IN)*F(IN-2)-
      L(IN-1)
      GOTO 45
   40 L(IN)=(B(IN)-J)*L(IN-1)-A(IN)*L0-M(IN-1)
      M(IN)=B(IN)*M(IN-1)-A(IN)*M0-K*L(IN-1)-C(IN-1)
      F(IN)=(B(IN)-J)*F(IN-1)-F(IN-1)-A(IN)*E0
      F(IN)=B(IN)*F(IN-1)-A(IN)*F0-L(IN-1)
   45 CONTINUE
      M(N)=M(N)+J*L(N)
      IF(I.EQ.0)M(N)=M(N)+(-1)**(N-1)
      F(N)=F(N)+J*E(N)
      ALPHA=K* E(N)+J*F(N)
      BETA =F(N)*F(N)-ALPHA*E(N)
      DELTAJ=(E(N)*M(N)-F(N)*L(N))/BETA
      DELTAK=(ALPHA *L(N)-F(N)*M(N))/BETA
      J=J+DELTAJ
      K=K+DELTAK
      IF(IFLG.NE.1)GOTO 95
C     SOLVE QUADRATIC EQN. TO OBTAIN A PAIR OF E/VALUES.
      IFLG=0
      I=I+1
      DISCR = J*J +4*K
      IF(DISCR.GE.0.0)GOTO 70
      DISCR =-DISCR
      WI(I)=SQRT(DISCR)/2.0
      WI(I+1)=-WI(I)
      W(I)=J/2.0
      W(I+1)=J/2.0
      GOTO 80
   70 DISCR=SQRT(DISCR)/2.0
      W(I)= J/2.0-DISCR
      W(I+1)=J/2.0+ DISCR
      WI(I)=0.0
      WI(I+1)=0.0
   80 CONTINUE
      I=I+1
      IF(N-I .LE.1)GOTO 100
      DO 90 IN =I-2,N
      IF(IN .EQ.0)GOTO 90
```

```
      C(IN)=L(IN)
      D(IN)=M(IN)
   90 CONTINUE
      C0 =L0
      D0 =M0
      J=W(I-1)
      K=W(I)
      GOTO 20
   95 IF(ABS(DELTTAJ).LT.EPS.AND.ABS(DELTAK).LT.EPS)
     1IFLG=1
      GOTO 20
  100 CONTINUE
      IF(N-1.NE.1)GOTO 110
      W(N)= B(N)-L(N-1)
      WI(N)=0.0
  110 RETURN
      END
```

# APPENDIX II

A2.

The recursive point partitioning (R.P.P.) Algorithms (4.8) and (4.11) for the solution of a general tridiagonal and a general symmetric quindiagonal matrix system respectively were both programmed in parallel (see Programs 12 and 13 of Appendix I) and then run on the Loughborough University Interdata Model 70 dual processor computer. The two processors which we denote as A and B, have private memory of 32kb and a further 32kb of shared core memory. Processor A is known to be approximately 4.6% faster than processor B.

In Table (A2.1) we give typical timing and other relevant results obtained from runs of the two parallel Programs 12 and 13. Calculations based on these results for the estimation of the speed-up and efficiency losses (due to memory clashing and parallel control overheads) of the two parallel programs are also given below.

Timings Used for Estimating Speed-ups and Efficiency Losses of Parallel Programs Run on the Interdata Parallel Computer

| Descriptions of events and/or parameters to be timed or measured | Timings (in secs.) | |
|---|---|---|
| | Tridiagonal System of order n=650 (Program 12) | Quindiagonal System of order n=650 (Program 13) |
| (1) Program running without Parallel Constructs | | |
| Processor B alone | 0.65 | 1.39 |
| (2) Program running with Parallel Constructs | | |
| Processor B alone | 0.67 | 1.42 |
| Processor A alone | 0.66 | 1.40 |
| (3) Program running with both Processors A and B running together in Parallel | | |
| (a) Processor B first (A & B in Parallel) | 0.36,0.35 | 0.74,0.74 |
| FJSHED† | 0.01,0.16 | 0.00,0.17 |
| GETRES+ | 0.00,0.00 | 0.1,0.00 |
| RESCHECKS* | 15,15 | 15,15 |
| (b) Processor A first (A & B in Parallel) | 0.35,0.35 | 0.73,0.73 |
| FJSHED† | 0.15,0.02 | 0.17,0.02 |
| GETRES+ | 0.00,0.00 | 0.00,0.02 |
| RESCHECKS* | 16,16 | 17,17 |

TABLE A2.1

†*denotes time processors are in the scheduler with nothing to do.*

+*denotes time in which there is waiting for parallel control of resources, such as getting/releasing protection mechanism.*

**denotes the number of checks made on resources.*

Typical Calculations Used to Estimate the Performance of Parallel Programs 12 and 13 (Based on timing results in Table A2.1)

Case 1: Tridiagonal matrix of order n=650 (Program 12)

(a) Shared Memory (Static) Overhead

Processor B alone (running in parallel) = 0.67 secs.

Processor A alone (running in parallel) = 0.66 secs.

Mean alone time (running in parallel) = 0.67 (=z say)

The above implies that A is $\frac{0.01}{0.66}\%$ = 1.5% faster than B.

But as A is known to be approximately 4.6% faster than B, this implies a shared memory overhead of (4.6-1.5)% = 3.1%

(b) Parallel Control (Static) Overhead

Processor B alone with parallel control = 0.67 secs.

Processor B alone without parallel control = 0.65 secs.

Parallel control (static) overhead = (0.67-0.65)/0.65% = 3.1%

(c) Speed-up

Running Alone Times

Running alone time for Processor B without parallel = 0.65 secs.

Running alone time for Processor A without parallel = 93.8% of 0.65
= 0.61 secs.

where 93.8% is obtained after taking into account the % overheads in (a) and (b).

∴ Mean alone running time = 0.63 secs. (=x say)

Running Together Times

Running time for Processors A and B together with A first = 0.35 secs.

Running time for Processors A and B together with B first = 0.36 secs.

Mean together time = 0.36 secs (=y say)

Speed-up of the parallel program 12 = $\frac{x \times 100}{y}\% = \frac{0.63 \times 100}{0.36}\%$ = 175%

(d) Parallel Control (Dynamic) Overheads (due to contention for parallel control)

Mean time in GETRES (for two runs) when A and B are in parallel

$= (0.0+0.0+0.0+0.0)/2 = 0.0$ secs.

Mean no. of RESCHECKS (for two runs) $= (15+15+16+16)/2$

$= 31$

1 RESCHECK takes 250 $\mu$ secs.

Mean time for RESCHECKS $= 31\times250\times10^{-6}$

$= .01$ secs.

Total contention time for parallel control $= (0.0+0.01) = 0.01$ secs.

Parallel control (Dynamic) overhead $= \frac{0.01}{z} = \frac{0.01}{0.67} = \underline{1.2\%}$

(e) Shared Memory (Dynamic) Access Overhead

Mean alone time with parallel control $= (0.67+0.66)/2 = 0.67$ secs.

Mean total active time when program is running in parallel

(for two runs) $= (0.36+0.35+0.35+0.35)/2$

$= 0.71$ secs.

Mean time in FJSHED $= (0.01+0.16+0.15+0.02)/2$

$= 0.17$ secs.

Mean time in GETRES $= 0.0$

$\therefore$ Time for shared memory access contention $= 0.71-0.67-0.17-0.0$

$= -0.7$

The negative value implies that within the accuracy of our calculations, the shared memory access contention $\simeq \underline{0.0\%}$

Case 2: Quindiagonal matrix of order n=650 (Program 13)

(a) Shared Memory (Static) Overhead

Processor B alone (running in parallel) = 1.42 secs.

Processor A alone (running in parallel) = 1.40 secs.

Mean alone time = 1.41 secs (=z say)

∴ Processor A is (0.02/1.40)% = 1.4% faster than B.

But A is known to be approximately 4.6% faster, which implies a shared memory (static) overhead of (4.6-1.4)% = 3.2%

(b) Parallel Control (Static) Overhead

Processor B alone with parallel control = 1.42 secs.

Processor B alone without parallel control = 1.39 secs.

Parallel control (static) overhead = $\frac{(1.42-1.39)\%}{1.39}$ = 2.2%

(c) Speed-up

Running Alone Times

Running aline time for Processor B without parallel controls

= 1.39 secs.

Running alone time for Processor A without parallel controls

= 94.5% of 1.39% = 1.31 secs.

where 94.5% is obtained after taking into account the % overheads in (a) and (b).

∴ Mean alone running time = (1.39+1.31)/2 = 1.35 secs. (=x say)

Running Together Times

Running time for Processors A and B together, with A first = 0.73 secs.

Running time for Processors A and B together, with B first = 0.74 secs.

Mean together time = 0.74 (=y say)

Speed-up of parallel program 13 = $\frac{x \times 100_{\%}}{y}$ = $\frac{1.35 \times 100_{\%}}{0.74}$ = 182%

(d) Parallel Control (Dynamic) Overheads (due to contention for parallel control)

Mean time in GETRES (for two runs) when A and B are in parallel

$= (0.10+0.004+0.00+0.02)/2 = 0.02.$

Mean no. of RESCHECKS (for two runs) $= (2+2+17+17)/2 = 19.$

1 RESCHECK takes 250 μ secs.

Mean time for RESCHECK $= 19\times250\times10^{-6} \simeq 0.005.$

Total contention time for parallel control $= (0.02)0.05) = 0.025.$

Parallel control (Dynamic) overhead $= \frac{0.025}{z} = \frac{0.025}{1.41} = \underline{\underline{1.7\%}}$

(e) Shared Memory (Dynamic) Access Overhead

Mean alone time with parallel control $= (1.42+1.40)/2 = 1.41$ secs.

Mean total active time when program is running in parallel

(for two runs) $= (0.74+0.74+0.73+0.73)/2$ secs.

$= 1.47$ secs.

Mean time in FJSHED $= (0.0+0.17+0.17+0.0)/2$ secs.

(for two runs) $= 0.17$ secs.

Mean time in GETRES $= (0.01+0.00+0.00+0.02)/2$ secs.

(for two runs) $= 0.02$ secs.

$\therefore$ Time for shared memory access overhead

$= (1.47-1.41-0.17-0.02)$ secs.

$= -0.13$ secs.

Hence, shared memory (dynamic) overhead $\simeq \underline{\underline{0.0\%}}$

A summary of these calculations for both Case 1 and Case 2 is given in Table (4.1) of Chapter 4.