THE UNIVERSITY OF ADELAIDE

DEPARTMENT OF ELECTRICAL ENGINEERING

THE USE OF MATHEMATICAL OPTIMIZATION TECHNIQUES IN THE DESIGN OF DIGITAL FILTERS

by

NEIL INNES SMITH, B.Sc., B.E.(Hons.)

A thesis submitted to the Faculty of Engineering of the University of Adelaide for the Degree of Doctor of Philosophy.

Adelaide. September, 1981.

TABLE OF CONTENTS

-o-o-o-o-o-o-o-o-o-

LIST OF FIGURES

LIST OF TABLES

SUMMARY

Several aspects of the use of mathematical optimization techniques in the design of digital filters are studied. In the sense employed in this thesis "mathematical optimization techniques" are those methods which have been developed for finding minima of general functions of several variables. The case in which the variables are constrained to take only integer values is among those considered. "Digital filters" include those designed to satisfy a frequency domain specification, and those generated as mathematical models of time-evolving dynamic processes.

The literature relating to previous such uses of optimization is reviewed. It is shown that many of these contributions may be placed in a consistent, unifying framework based on general features of the mathematical formulation. A number of generally-applicable algebraic relations are derived. Several minor new techniques which streamline the application of the general minimizing methods to digital filter design are introduced.

A great many methods are available for finding unconstrained (local) minima of general functions. Extensive numerical experiments are performed with a selection of these to study their relative effectiveness when applied to representative digital filter design examples. The examples include approximation problems in the frequency domain (both magnitude and group delay responses) and in the time domain. Optimization methods tested include those of second-derivative, Gauss-Newton, quasi-Newton and conjugate gradient type.

This examination of the performance of optimization methods is relevant to (*inter alia*) their use in speed-critical on-line applications. In a follow-up study, an optimization algorithm has been implemented on fast, cheap, short-wordlength digital hardware. The suitability of such a system for on-line system identification is studied. The work reported includes the development of the computer itself as well as the algorithms.

The *discrete* optimization problem associated with choosing digital filter coefficients *of a given wordlength* (and so, quantized) is considered. A novel optimization procedure is introduced and shown by numerical experiment to have superior efficiency to many previously-suggested methods, at least for certain classes of problem.

Chapter one serves as a more detailed introduction to the contents of this thesis.

Statement of Originality

This thesis contains no material which has been accepted for the award of any degree or diploma in any University. To the best of the author's knowledge and belief, this thesis contains no material previously published or written by any other person, except when due reference is made in the text.

(N. I. SMITH)

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Professor R. E. Bogner for his constant encouragement, and for many helpful suggestions and much constructive criticism of this work. Thanks are due also to Dr. B. R. Davis, who provided guidance during a period when Professor Bogner was absent from Adelaide on Study Leave.

Acknowledgement is due to Dr. J. A. V. Rogers, Dr. B. D. Ackland, and particularly to Mr. D. S. Fensom for their contributions to the design and commissioning of the GASP computer. The considerable assistance of several of the technical and office staff of the Department of Electrical Engineering, in particular that of Mr. N. R. Blockley and Mrs. M. T. Drake, is also much appreciated with regard to the GASP project. Thanks are due to the Subroutine Librarian, AERE Harwell (U.K.), for providing subroutine MA29B, without charge.

The typing of this thesis was carried out by Mrs. H. J. Koennecke, Mrs. M. T. Drake and Mrs. E. M. Doubtfire. The skill and particularly the cheerful manner with which they all tacked this quite intricate task is gratefully acknowledged.

CHAPTER ONE

1. SCOPE AND ORGANIZATION OF THIS THESIS

1.1 Organization

In this thesis an exploration is made of several aspects of the use of mathematical optimization techniques in the design of digital filters.

Three essentially separate studies have been carried out. For this reason the organization of the thesis is perhaps slightly unconventional; in particular, no "central problem" can be immediately introduced; nor is there a final chapter reporting "overall conclusions". Each of the studies is reported on in its own large chapter, with an introduction and a section on "conclusions" included. The twin common threads running through the entire work are digital filtering and the use of mathematical optimization techniques.

The chapters referred to above are numbered five, six and seven. The earlier chapters serve principally to introduce terminology and notation and to review the literature relevant to the topics to be discussed. Chapter four, however, additionally contains some original material whose presentation in that chapter has a certain logical justification.

Each of the studies has involved the generation of a large amount of computer code, in three programming languages and for three different machines. Inclusion of the complete printed listings would have more than trebled the size of the thesis without serving any especially useful purpose. Accordingly, virtually all of this material has been omitted; the computer programs are merely described where necessary per medium of flow-

charts, algebraic relations and ordinary text. The one exception is a short FORTRAN subroutine presented as an appendix. This, an implementation of what is thought to be a novel algorithm, serves as an example of several of the programming techniques.

Some of the material of chapters six and seven has been the subject of prior publication. It seems to make little sense to paraphrase these papers, and yet undesirable to break continuity by merely including reprints in an appendix. Accordingly, copies of these published papers are bound in at the appropriate place in the body of the text.

1.2 Comparison of Optimization Techniques

A survey of the literature shows that many investigators in the digital signal processing field have made use of what may be termed "mathematical optimization techniques", taken (somewhat restrictively) here to mean techniques for finding the (constrained or unconstrained) minimum value of a single function of several variables. Such techniques are many and varied, and they are of such great generality that their detailed development could be (and was) largely carried out by mathematicians who did not need to consider expressly the details of any particular user's application.

It is rare in the signal processing literature to find a work comparing even a couple out of the wealth of optimization techniques which applied mathematicians have made available. Most authors have been content to use standard versions of general minimization algorithms available at most large scientific computer sites. This approach is justifiable because the majority of uses of the techniques have been in the computer aided design of digital filters. Although the filters are *used* in high-speed

real-time applications they are *designed* in advance, off-line, and considerations such as

a) the amount of computer time used in the design,

and b) whether the iterative procedure would have converged from a different starting point,

are not of central importance. Obviously, though, a method which offered a factor of 2 improvement in execution time would be worthwhile if the design program were to be used extensively.

Perhaps more importantly, many of the same methods are applicable to situations where digital filters are "designed" in real time, that is to say, the filter coefficients are required to "adapt" to changes in the characteristics of the signal being processed. With the ever-increasing availability of computing power (including faster logic together with more parallelism of computation, at fast-reducing cost) the use of optimization techniques is becoming feasible for more and more demanding applications. It seems desirable to compare some of them in representative applications, from the point of view of reliability and relative execution speed.

Chapter five of this thesis reports upon work done to compare the performance of several mathematical optimization methods in determining the parameters of digital filters. The "filters" considered include those designed to meet a pre-assigned frequency-domain "specification" and those produced as mathematical models of time-evolving processes.

As preliminaries to this material, chapters two and three discuss optimization fundamentals and the mathematical basis of some optimization methods. The requisite terminology and notation from the digital filtering field is introduced in chapter four.

Extensive numerical comparisons of optimization techniques are rare even in the general mathematical literature. Second-derivative methods have been particularly neglected. The results of chapter five may go some way towards remedying this deficiency. Although all the test examples are (deliberately) taken from the digital filtering field it is likely that some of the conclusions regarding performance would be much more generally applicable.

1.3 General Algebraic Development of Optimization in Digital Filter Design

As stated above, chapter four serves to introduce the fundamentals of digital filtering. The literature relating to the uses of optimization in this field is also reviewed. This literature survey is "taxonomic" in nature; it is shown that much of the work may be classified according to general features of the mathematical formulation. Accordingly, a certain amount of algebraic development is possible without fixing attention on one particular filter design problem. A number of general equations are derived. This approach of proceeding from the general to the more particular is believed to be original and of some use in unifying a fragmented field.

Several minor new techniques which generally facilitate the application of optimization to digital filter design are also described. In particular, a method for reviewing the pairing of real poles and zeros as the computation proceeds greatly improves reliability; this is because it eliminates many sub-optimal "solutions" to which the procedure could otherwise converge.

Efficient filter structures are derived for the calculation of the first and second derivatives required when treating approximation problems in the time domain.

1.4 Finite-Wordlength Considerations

The results of an optimization process of the type considered thus far is a vector of optimal parameters whose values are essentially known with infinite precision (although of course this is limited in practice by the wordlength of the computer used). Digital filters, however, are often implemented as routines on 16-bit minicomputers, 8-bit microprocessors, or special-purpose pipelined hardware, and their coefficients are restricted to take only discrete values, the "quantization" being significant because of the short wordlength.

Chapter six is devoted to the matter of coefficient quantization. The literature in this area treats three main topics: filter structures, statistical analysis of wordlength requirements, and discrete parameter optimization. The first two of these topics are reviewed for completeness, but in keeping with the overall theme of the thesis, the main emphasis is on the third. Rounding the infinite-precision "ideal" coefficients to the nearest permissible values usually does not produce the optimal filter for a given wordlength, and the design may be improved by applying some kind of "discrete optimization" procedure. Many alternative methods have been suggested; the relevant literature is surveyed. The techniques mostly fall into two categories:

(a) Those which are in theory capable of seeking a true optimum, but which generally require vast amounts of computer time, and

(b) Heuristic procedures with little theoretical justification and little chance of finding the true optimum (although some may find good sub-optimal solutions quite efficiently).

A new method is proposed which is based on a completely novel approach. The procedure (involving a random search) is

guided by substantial theoretical considerations and there is a good probability of finding the true optimum (although if it is found there is no indication of optimality). Tests are performed which indicate good efficiency in finding a variety of useful sub-optimal filters as well. In this regard the new method is probably superior to most other methods at least for certain classes of problems.

1.5 A High-Speed Implementation of System Identification by Optimization Techniques

Chapter seven deals with a study of the implementation on fast, cheap, short-wordlength hardware of "system identification" by a gradient-based mathematical optimization method.

The work reported includes the development (with others) of the "General Arithmetic Signal Processor" (GASP) - a very fast computer with architectural features which suit it to a wide variety of signal processing tasks.

The design of system identification software to run on GASP is discussed. This includes homomorphic (cepstral) processing as well as an optimization scheme for parameter determination. Attention is focussed on "speech-like" signals, with the aim of identifying the parameters of a pole-zero cascade-form model.

Results of tests of the system identification algorithm are presented. Although some success is achieved, this approach to system identification has serious shortcomings which will probably restrict its applicability. These are discussed and possible remedies are suggested.

Finally, the suitability of the machine GASP for such uses is reviewed. The usefulness of many of its design features

is confirmed. Several additions are suggested which would improve the machine whilst retaining its original concept.

CHAPTER TWO

2. OPTIMIZATION FUNDAMENTALS

2.1 General Nature of Optimization

A large body of mathematical literature has been built up relating to the problem of finding the minimum value of a general function of N variables, and the particular values of the variables which produce such a minimum. The amount of effort which has gone into such studies is justified by the fact that a great number of problems in *engineering design* and in the estimation of the parameters of *mathematical models* of physical processes may be cast in this form.

In the case of *design*, the function to be minimized may be the monetary cost of an item or be a measure of some other undesirable attribute such as energy loss, or, usually, a suitably weighted combination of several such "undesirables". The N variables are the free parameters which the designer is able to vary. In the case of mathematical *modelling*, a model whose *form* has been determined by some process of art, or intuition, is left with N free numerical parameters. These are then found by minimizing a function of them which in some way quantifies the "closeness of fit" of the model to the experimentally observed data.

In most cases of practical interest, all or at least some of the "free" variables are restricted by some physical consideration to lie above or below some particular value, or within certain limits. For example, the values of all inductors, capacitors and resistors in an electrical circuit must be positive. Such *constraints* do not necessarily apply to the variables independently of one another. As an example of a more complicated constraint, a second-order digital filter having a transfer function expressed as

$$H(z) = \frac{1}{1 + a\,z^{-1} + bz^{-2}}$$

must be stable to be useful; this restricts the allowable values of parameters a and b to a triangular region of the a-b plane.

Much more general classes of problems have been studied under the heading "optimization", in particular the optimization of *functionals* in which we seek to determine not only the values of a set of parameters but also the *form* of a function which will minimize some quantity. However, in the field of digital filter design much use has been made of the (simpler) minization of a given function of N variables, and this thesis will be confined to an exploration of "optimization" in that sense.

2.2 Definitions and Terminology

The following, while not purporting to be mathematically rigorous, is a review of general optimization terminology. Some of the more standard mathematical terms used in the thesis and not defined here are given in appendix A.

The recognition of the importance of constraints has led to the formulation of what is usually termed the *mathematical programming problem*, formally stated as:

$$\begin{aligned} &\text{Minimize} && F(\mathbf{X}) \\ &\text{subject to} && Q_i(\mathbf{X}) \geqslant 0 \qquad i = 1, 2 \ldots M \end{aligned} \tag{2.1}$$

where $\mathbf{X}$ is the N-dimensional column vector of parameters (arranged in some arbitrary order) and the M inequalities express the constraints. The function F is termed the *objective function*.

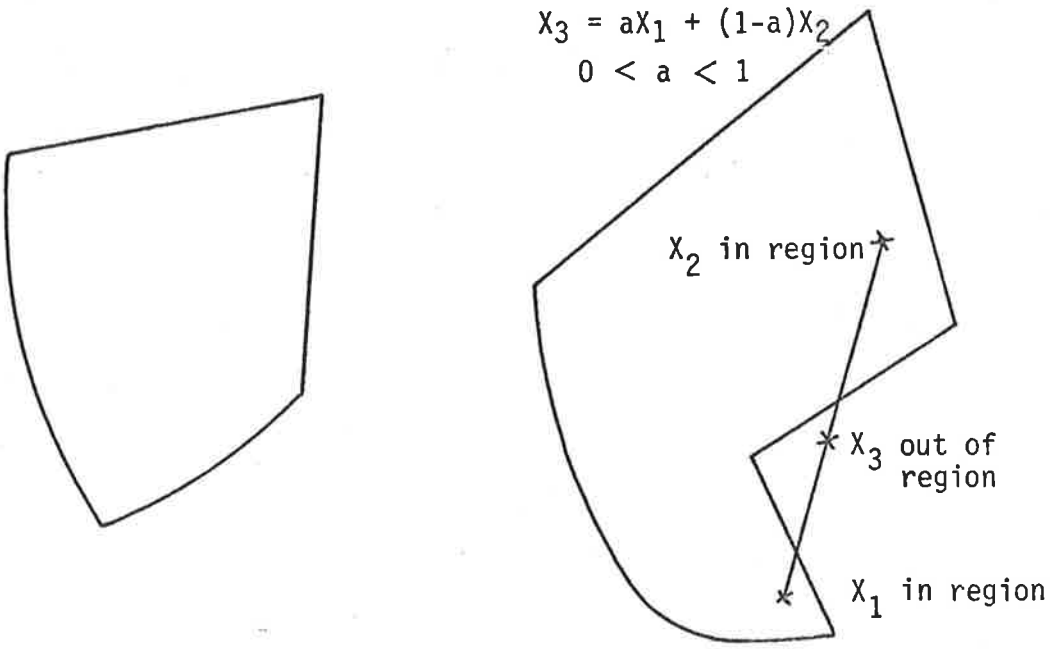
If the constraints are not mutually contradictory, they define a region of parameter space in which a solution, if it exists, must lie. This region, which may be of finite or infinite extent, is the

feasible region and any parameter vector within it is a *feasible solution*. If every convex linear combination of every two feasible solutions is itself a feasible solution, the feasible region is *convex*, a concept illustrated in figure 2.1.

Turning now to the objective function (and ignoring the constraints) a *local minimum* of the function is any parameter vector having a neighbourhood within which every parameter vector yields a function value greater than or equal to that at the local minimum itself. If equality is excluded the point is a *strong local minimum*. A *unimodal path* is a directed curve in parameter space along which the function value decreases monotonically. Such a path must either *terminate* at a local minimum or at least one component of the parameter vector must diverge to infinity. If *all* unimodal paths terminate at the same point the function is called *unimodal*, and necessarily has only one local minimum.

Unimodal functions are high desirable from the point of view of minimization because most techniques seek local minima by some kind of exploratory procedure. For a unimodal function such a point must be the *global* (unconstrained) minimum. However, the presence of constraints complicates the matter enormously. A *constrained local minimum* is a feasible solution with function value F, say, such that there exists a neighbourhood within which every parameter vector whose function value is less than F is infeasible. In other words, it is either a (feasible) local minimum or it is a point at which one or more constraints are *binding* and from which all unimodal paths extend out of the feasible region. Figure 2.2 shows that even if the objective function is unimodal there may be more than one constrained local minimum. However, if the function is unimodal and the (one) local minimum is feasible, then it is also the global constrained minimum.

In many practical cases, unimodality is virtually impossible to



(a) A Convex Region (b) A Non-Convex Region

FIGURE 2.1 Convex and Non-Convex Regions

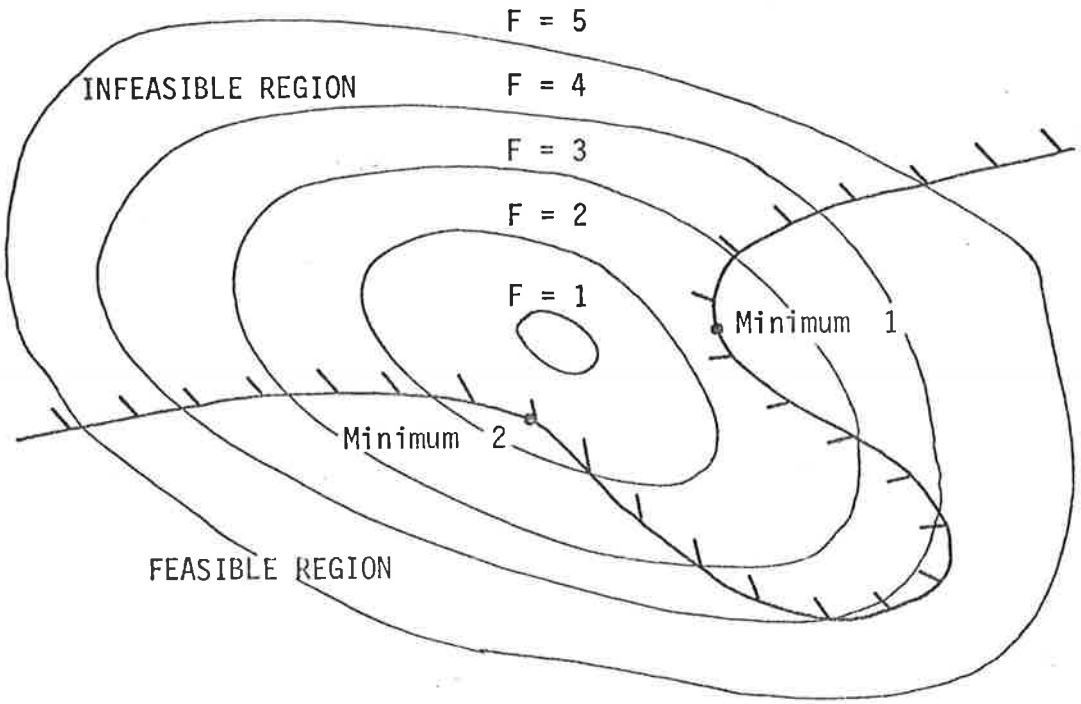


FIGURE 2.2 Multiple Constrained Minima of Unimodal Function

prove, and so if the user seeks the global minimum he should attempt to find a variety of local minima, or constrained local minima, and choose the best among them. Of course, it may not be important to find the global minimum because a suboptimal solution to the problem may be good enough, but this will depend on the application.

A more restrictive condition on a function than unimodality is that of *convexity*, meaning that the function value at a point which is a convex linear combination of any two points in parameter space, is less than or equal to the weighted arithmetic mean of the corresponding function values. In symbols,

$$F(a\mathbf{x}_1 + (1 - a)\,\mathbf{x}_2) \leq a\,F(\mathbf{x}_1) + (1 - a)\,F(\mathbf{x}_2) \qquad (2.2)$$

$$\text{for } 0 < a < 1, \text{ and any } \mathbf{x}_1 \text{ and } \mathbf{x}_2.$$

A convex function is necessarily unimodal, but not vice versa. The useful thing about convexity is that if both the function and the feasible region are convex, there is only one constrained local minimum, i.e. the global solution.

The above serves mainly to introduce some of the general optimization terms to be used from time to time in this thesis. While unimodality and more particularly convexity are of some theoretical importance, in most practical cases we are forced to work with functions which are neither. However, one conclusion to be drawn from such a survey of optimization fundamentals is that exploratory (local-minimum-seeking) techniques are likely to be more successful if the objective function does not have a large number of such minima. Furthermore, if the user has any freedom in the way in which the objective function is defined, multiple minima should not be deliberately introduced.

2.3 Types of Optimization Techniques

2.3.1 Linear programming

When both the objective function and all the constraint formulae (the Q_i of (2.1)) are linear in the variable parameters the problem is amenable to solution by the well-studied methods of linear programming.

A linear function (apart from a constant) if defined for all values of the parameter vector $\mathbf{x}$ is not bounded below, and so the existence of a finite optimum (minimum) depends basically on the constraints. The optimum in such circumstances always lies on the boundary of the feasible region, which is a convex polyhedron in N-dimensional space.

Formulation of a problem as a linear program is desirable because the methods of solution are highly standardised and efficient. If a global optimum exists it is always found in a finite number of steps, and very large numbers (thousands) of independent variables may be handled. Some types of digital filter design problems have been so treated and are mentioned in chapter four. This is mainly from the point of view of completeness of the literature survey because the remainder of this thesis will concentrate on problems not admitting of a linear program formulation.

2.3.2 Unconstrained optimization

Whilst the solution of a linear program exists only because of the constraints, most classes of objective function will have a minimum (or several local minima) in their own right. A large number of techniques for finding such minima have been developed, and these are the type to be considered in most of this thesis. Chapter three reviews such methods. It is sufficient here to mention that most are

iterative methods, proceding step-by-step through parameter space in such a way that the objective function is decreased at each stage, in the hope of converging eventually to a local minimum.

Unconstrained optimization is important not because truly unconstrained problems are common but because various modifications allow the methods to be used for many practical constrained problems also. The principal types of modification are introduced in section 2.3.3.

There remains an important type of problem in which the constraints are expressed by *strict* inequalities (rather than the nonstrict inequalities of (2.1)). In such a case the only acceptable solutions are *interior* points of the feasible region and so are true unconstrained local minima of the objective function. Once such a minimum has been approached sufficiently closely, the constraints have no further bearing on the progress of the optimization algorithm.

This is exemplified by some of the digital filter design problems to be considered. All the poles of the filter must lie strictly inside the unit circle so that the free response actually decays with time. The only type of optimum of interest is a true local minimum of the objective function (at a point interior to the feasible region) and this may be found without considering the constraints, provided that we begin our search procedure sufficiently close to the wanted solution (or are lucky).

This approach to constraints, of ignoring them when there is a reasonable probability that the procedure will converge to a solution which satisfies them anyhow, will be called a *relaxation* approach. The computational algorithm should include safeguards to prevent the search procedure uselessly wandering around in the infeasible region, but this may be a device as simple as re-starting from a new feasible point as soon as a constraint is violated. More generally, some opportunity may

be given for a *return* to feasibility before such a restart.

2.3.3 Constrained Nonlinear Optimization

When physical considerations do *not* prevent inequality constraints from becoming equalities, there may be local optima on the boundaries of the feasible region as well as in its interior. Such problems may be tackled by two basic general methods, the feasible-direction and penalty-function techniques.

With a feasible-direction technique, an unconstrained method is used but if the search procedure encounters a constraint the ordinary predicted (infeasible) step is not followed; rather some *useable feasible direction* (one which allows a decrease of objective function value while maintaining feasibility) is computed. Several such techniques have been suggested, a survey being given by Gottfried and Weisman (1973), chapter 5.

In the case of linear constraints, the "gradient projection" method is effective (Rosen, 1960). Any unconstrained method may be used, starting from an interior feasible point. If the search causes a constraint (hyperplane) to be contacted, exploration is made *along* the constraint in the direction specified by the projection of the negative gradient onto the hyperplane. If further constraints are encountered the gradient is projected onto the intersection of all binding constraints, and so on, as long as the negative gradient direction itself remains infeasible. The search will terminate at a constrained local minimum, either a vertex of the polyhedron formed by the constraint hyperplanes or a point where the gradient is normal to the intersection of the binding constraints.

The gradient projection method may be used when the constraints are nonlinear, by substituting projections on the hyperplane tangent to a constraint. However, the resulting direction may be infeasible, and

although there is a correction technique for returning to the feasible region after such a step, the method is not very successful for non-linear constraints (Gottfried and Weisman, 1973).

Even when the minima sought are interior points of the feasible region, gradient projection may be useful in avoiding some of the forced re-starts of the simple relaxation method of section 2.3.2.

Penalty-function techniques are of two kinds, both allowing the use of an unmodified unconstrained algorithm. In an "exterior-point" algorithm the objective function is artificially augmented by a "penalty" term whenever a constraint is violated, and the further the point is into the infeasible region the higher the penalty. The penalty is generated from a suitable simple formula satisfying this condition and also allowing the steepness of the penalty "wall" to be controlled by a parameter. The sequence of unconstrained optima found with increasing values of the penalty parameter, all lying *outside* the feasible region, will tend in the limit to the true constrained minimum. Such a "sequential" mode of operation is normally necessary because the immediate use of a very severe penalty produces a sharp valley in the augmented objective function. Many unconstrained search techniques have difficulties in following such valleys.

In "interior-point" penalty function algorithms the search must be started at a feasible point, and is prevented from reaching a constraint by augmenting the objective function with a barrier term which increases without limit as the constraint is approached. Again a sequence of unconstrained minima are found, in this case being *inside* the feasible region. In successive unconstrained sub-problems the barrier is made steeper and more localized (closer to the constraint). In such a way the true constrained local minimum is eventually approached.

Penalty-function techniques are discussed in detail in a text

by Fiacco and McCormick (1968).

There are other methods which are applicable to certain types of nonlinear programming problems. These include several linearization techniques which allow the problem to be solved as a sequence of linear programs (Gottfried and Weisman, 1973). They are not usually recommended when the nonlinearities are severe and are not considered further in this thesis.

2.3.4 The uses of calculus

The classical calculus is of course an optimization technique in the sense that for unconstrained differentiable functions $F(\mathbf{x})$, all local minima will be among the solutions of the simultaneous algebraic equations

$$\frac{\partial F}{\partial x_i} = 0 \qquad i = 1, 2 \ldots N. \qquad (2.3)$$

(2.3) represents only the necessary conditions for a local minimum, other possible solutions being local maxima, saddle points, and horizontal inflection points, and so it is necessary in seeking a minimum by classical methods to explore the neighbourhood of each of the candidate points. A particularly useful result is that if the Hessian matrix

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_N} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & & \\ \vdots & & & \\ \frac{\partial^2 F}{\partial x_N \partial x_1} & & \cdots & \frac{\partial^2 F}{\partial x_N^2} \end{bmatrix} \qquad (2.4)$$

is positive definite at such a point, the point is a strong local minimum.

Unfortunately, the complexity of the equations (2.3) is such that the classical calculus is not a practical method for minimizing most realistic objective functions. It does, however, provide most of the insights which have guided the formulation of the practical minimization methods of chapter three. In particular, although there is no fundamental requirement that an objective function be differentiable at the minimum (or elsewhere), differentiability greatly aids in minimization and most successful algorithms employ gradient information explicitly. Algorithms employing also the second derivative (Hessian) matrix are usually even more efficient.

The presence of constraints greatly increases the amount of work necessary in the classical approach, because the function must be minimized on the boundaries of the feasible region as well as within it in order to find all possible minima. Separate problems must be solved with each constraint alone assumed binding (acting as an equality), then each pair of constraints, and so on, up to each combination of N constraints. (The imposition of more than N equality constraints simultaneously leads to an overdetermined set of equations; there is no feasible solution.) These equality-constrained problems may be tackled in two ways:

(a) If a constraint equation may be solved for one variable in terms of the others, the resulting expression may be substituted into the definition of F, resulting in a problem of *reduced* dimensionality but usually increased complexity.

(b) The alternative is the elegant method of *Lagrange multipliers* which always *increases* the dimensionality of the problem but usually simplifies the solution.

If we wish to minimize $F(x_1, x_2 \ldots x_N)$ subject to equality constraints

$$Q_i(x_1, x_2, \ldots x_N) = 0 \qquad i = 1, 2 \ldots M$$

we form the *Lagrangian function*

$$\mathcal{L} = F(x_1, x_2 \ldots x_N) + \sum_{i=1}^{M} \mu_i Q_i(x_1, x_2 \ldots x_N) \tag{2.5}$$

and proceed to find the stationary points of $\mathcal{L}$ in the (N + M)-dimensional space of all the x and μ variables. A point so found is a constrained local minimum of F if $\mathcal{L}$ exhibits a minimum with respect to each x variable and a maximum with respect to each μ, that is, a saddle point.

2.3.5 Integer programming

In many problems of practical interest the parameters are not permitted to take on a continuum of values but are in some way quantized. The digital filter design problems considered in this thesis are of this type, and this matter is treated in some detail in chapter six. The purpose of this section is to point out that there are some general methods for locating such quantized optima.

In the majority of cases the allowed points are equally spaced and some suitable scaling will cause the quantization to be expressed as an *integer* requirement on some or all of the variables. There is the obvious division into *all-integer* and *mixed-integer* programming problems, both types being of practical significance.

The case of the *linear* programming problem with added integer constraints is the most highly developed, and two general methods are available. The "cutting plane" method of Gomory (1958) does not readily generalize to nonlinear problems and even in the linear case appears inferior to the "branch-and-bound" technique of Land and Doig (1960) and modified by Dakin (1966). The latter technique is not intimately bound up with linear programming methods and may be used for nonlinear

problems, although it is not guaranteed to find the optimum unless the objective function is convex (Gottfried and Weisman, 1973, chapter 6).

The essence of the branch-and-bound approach is shown by two-dimensional example in figure 2.3. The problem is first solved ignoring the integer constraints, producing a solution at the point $x_1 = 3.5$, $x_2 = 3.3$ with function value (say) 20.0. Since both x_1 and x_2 must be integers at the final solution, we can exclude the region $3 < x_1 < 4$ from consideration and re-solve the problem in the two disjoint feasible regions $x_1 \leqslant 3$ and $x_1 \geqslant 4$. The optimal solutions to these two problems will have function values greater than 20.0 (or else the unconstrained minimum would not have been where it was), and, if the objective function is convex the minima will lie *on the region boundaries*, thus satisfying the integer constraint on x_1. We assume the function values to be 25.0 (point J_1) and 30.0 (point J_2). If only x_1 were constrained to be an integer we would simply choose the smaller of these two values as the final solution. With x_2 also constrained, we proceed to split each semi-infinite feasible region into two again (the shaded regions of figure 2.3, labelled A, B, C and D). Suppose we elect next to minimize the function over the region C (reasonable, since this region is "nearest" to the smaller of the two known minima so far). If the function is reasonably well-behaved we have a significant probability of obtaining as this minimum the "corner" point, labelled K_1, which satisfies the integer constraints on both x_1 and x_2. Suppose this happens, and that the function value is 28.0. We are then spared the trouble of searching the regions A and D, because neither could yield a function value smaller than 30.0, and K_1 is already feasible and a better solution than this.

Minimizing over region B, we obtain point J_3 with value 26.8. This point has a fractional value for x_1 and so is not a feasible solution. But since the function value is below 28.0, region B may

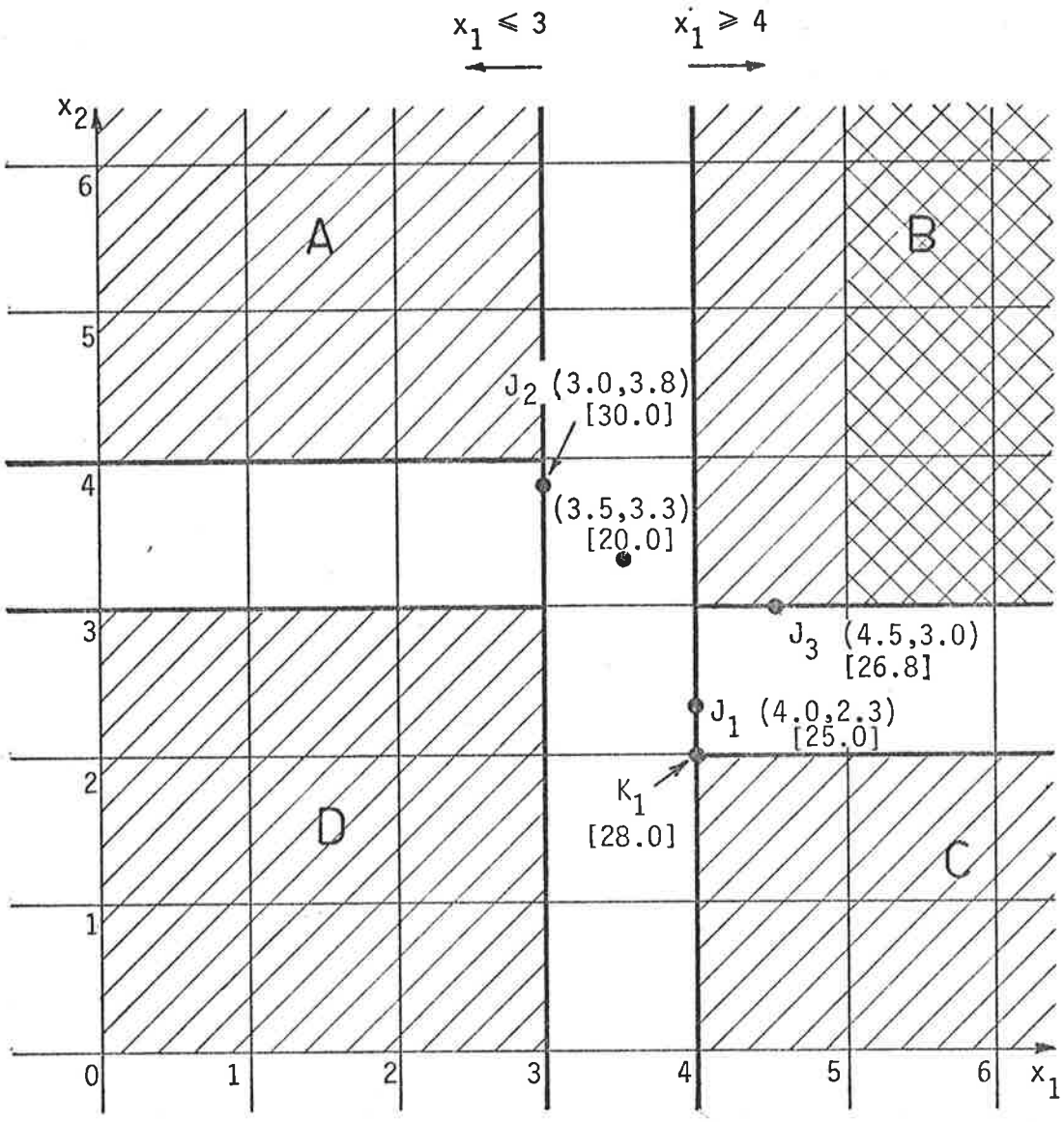


FIGURE 2.3 Example of Integer Programming Solution by Branch-and-Bound Technique

coefficient values shown thus (x_1,x_2)

function values shown thus [F]

still contain a better solution than point K_1, and so it is again split into two subregions, $x_1 \geqslant 5$ (shown cross-hatched) and $x_1 \leqslant 4$ (which, owing to the prior bound $x_1 \geqslant 4$, degenerates to the *line* $x_1 = 4$, $x_2 \geqslant 3$.

The branch-and-bound approach is thus one of tree-search. It is also a relaxation method, in the sense that the integer constraints never enter explicitly into the minimization sub-problems, but are replaced by ordinary inequality constraints which lead to problems with a high probability of an integer solution. The possible feasible minima obtainable from a given branch of the tree are always bounded in function value by the solution of a less-constrained problem associated with that branch, and so whole tree branches may be abandoned when some other feasible solution is better.

The method works best with convex objective functions (and particularly with linear programs) but may be applied to non-convex functions with the proviso that the global optimum will not necessarily be found. If the feasible grid is of fine spacing relative to the size of the irregularities in the function contours (as with many digital filter problems) this theoretical deficiency is not important, the function being effectively "convex" over the entire region of interest. As with most optimization methods, however, the requirement in computer time increases dramatically with dimensionality. In chapter six of this thesis a new method is introduced which, while lacking the theoretical elegance of branch-and-bound, appears to be much more practical for the types of problems considered.

CHAPTER THREE

3. TECHNIQUES FOR UNCONSTRAINED NONLINEAR OPTIMIZATION

3.1. General

We turn now to a review of the various types of algorithms which have been proposed for finding unconstrained local minima of nonlinear functions, and a description in some detail of methods to be compared experimentally in chapter five of this thesis. The utility of such methods is not limited to problems characterized by *truly* unconstrained functions; in chapter two it was pointed out that constrained problems may often be treated as a sequence of unconstrained sub-problems by the penalty-function technique, and there is also the possibility of *transforming* the independent variables so that the new variables are unconstrained (Box, 1966; Powell, 1972). Additionally, there is a class of methods based on Lagrange multipliers (Powell, 1972).

Most techniques for the unconstrained problem are *sequential* in the sense that they start at some arbitrary point in parameter space, $\mathbf{x}^{(0)}$, and proceed by a series of *iterations* to generate points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ... having successively smaller values of the objective functions $F(\mathbf{x})$. They thus seek *local* minima and unless the function is known to be unimodal there is no guarantee of optimality. In practical cases, confidence in the optimality of the final solution may usually be increased by starting the procedure from a variety of different points. Another approach to the problem involves generating trial points at random, with or without some capacity for "learning". (Gottfried and Weisman , 1973, Section 3.4). Such methods are not

sequential and can in principle find global optima - however the amount of computer time is usually prohibitive even when the dimensionality is low. Systematic grid-search procedures are defeated even more rapidly by the problem of dimensionality.

3.2 Classification of Sequential Methods

In general, the more well-behaved a function is, the more sophisticated can be the methods for minimizing it. We denote the class of functions with continuous rth derivatives as C^r. If $F \in C'$, a method which makes use of gradient information may be used, while if $F \in C^2$ there are useful methods employing second derivatives. There are also those methods which employ function values only, and which may be used with non-differentiable functions, and in some cases with functions which are not even continuous. These three types in fact constitute the primary classifications of sequential unconstrained minimization techniques.

It is of course possible to use a "value only" technique on a differentiable function, and this *may* be desirable if the gradient evaluation would be very time-consuming or if the analytic differentiation is simply too complicated an algebraic task. In such cases it may be desirable to use function evaluations specifically to produce finite-difference approximations to the gradient components, and the "value only" methods are further divided into those which do and do not do this.

In a similar way, second derivatives may be approximated by differences of first derivatives. In this thesis all methods employing at least first derivative information are classified as "gradient methods" and are discussed at some length in Section 3.4. The following section is devoted to "value only" methods.

3.3 Methods Employing Function Values Only

3.3.1 Gradient Approximation

Of those methods which generate finite-difference approximations to the gradient components, the best known is that of Stewart (1967). The obvious formula for the ith gradient components is

$$g_i \approx \frac{1}{h} [F(\mathbf{x} + h \ \mathbf{e}_i) - F(\mathbf{x})] \qquad (3.1)$$

where $\mathbf{e}_i$ is the ith unit vector. The main thrust of the Stewart paper is to select the step length h optimally, so that the (Taylor series) truncation error implied by (3.1) just balances the numerical cancellation error (caused by forming the difference of two nearly equal qualities). Otherwise, the method has the characteristics of the gradient method to which it is coupled (in Stewart's case, the Fletcher-Powell method described in Section 3.4.14).

3.3.2 Conjugate-Direction Methods

Another class of methods is based on the properties of quadratic functions. A set of non-zero N-vectors $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)} \ \ldots \ \mathbf{p}^{(N)}$ are called *conjugate* with respect to a given positive-definite matrix $\mathbf{A}$ (or "$\mathbf{A}$ - conjugate") if

$$\mathbf{p}^{(i)^T} \ \mathbf{A} \ \mathbf{p}^{(j)} = 0 \quad \text{for all } i \neq j \qquad (3.2)$$

Such vectors are necessarily linearly independent. In the case of a quadratic objective function $F(\mathbf{x})$ the Hessian matrix $\mathbf{H}$ is constant. If such a function is successively minimized along several (say k) $\mathbf{H}$ - conjugate directions, starting from an arbitrary point $\mathbf{x}^{(0)}$, then the final point found ($\mathbf{x}^{(k)}$) is

the minimum of the function in the subspace defined by the initial point and the k conjugate directions. A proof of this is given by Fletcher (1972a).

This theorem implies, in particular, that *the minimum* value of a quadratic function is found after exact linear searches in all N conjugate directions.

Conjugate directions can be generated in a number of ways without explicit knowledge of the Hessian matrix. A method employing first derivatives is described in Section 3.4.13. However, there is another technique which requires no gradient evaluations and which forms the basis of the method of Powell (1964). If S_1 and S_2 are two "parallel subspaces" generated by a set of linearly independent vectors $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)}$, ... $\mathbf{p}^{(k)}$ and points $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{x}_1 \notin S_2$ (and $\mathbf{x}_2 \notin S_1$) and $\mathbf{z}_1$ and $\mathbf{z}_2$ are the points minimizing a quadratic function in S_1 and S_2 respectively, then $\mathbf{z}_2 - \mathbf{z}_1$ is $\mathbf{H}$-conjugate to each of the $\mathbf{p}$'s. This is the "parallel subspace property" and is illustrated in two dimensions in figure 3.1.

Powell's (1964) method obtains conjugate directions for a quadratic function, and hence finite termination, by virtue of the parallel subspace property. But it is also capable of easy extension beyond N conjugate line searches and so tends to converge to the minimum of a general function. Consider that at the start of a given iteration (say, the kth) a set of N linearly-independent vectors is defined. This set may initially be *any* independent set, such as the unit vectors. It is updated by discarding one vector and calculating a new one at each iteration in such a way that it eventually tends to a conjugate set with respect to the Hessian evaluated at the minimum. We denote the current set of vectors by $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)}$, ... $\mathbf{p}^{(N)}$,

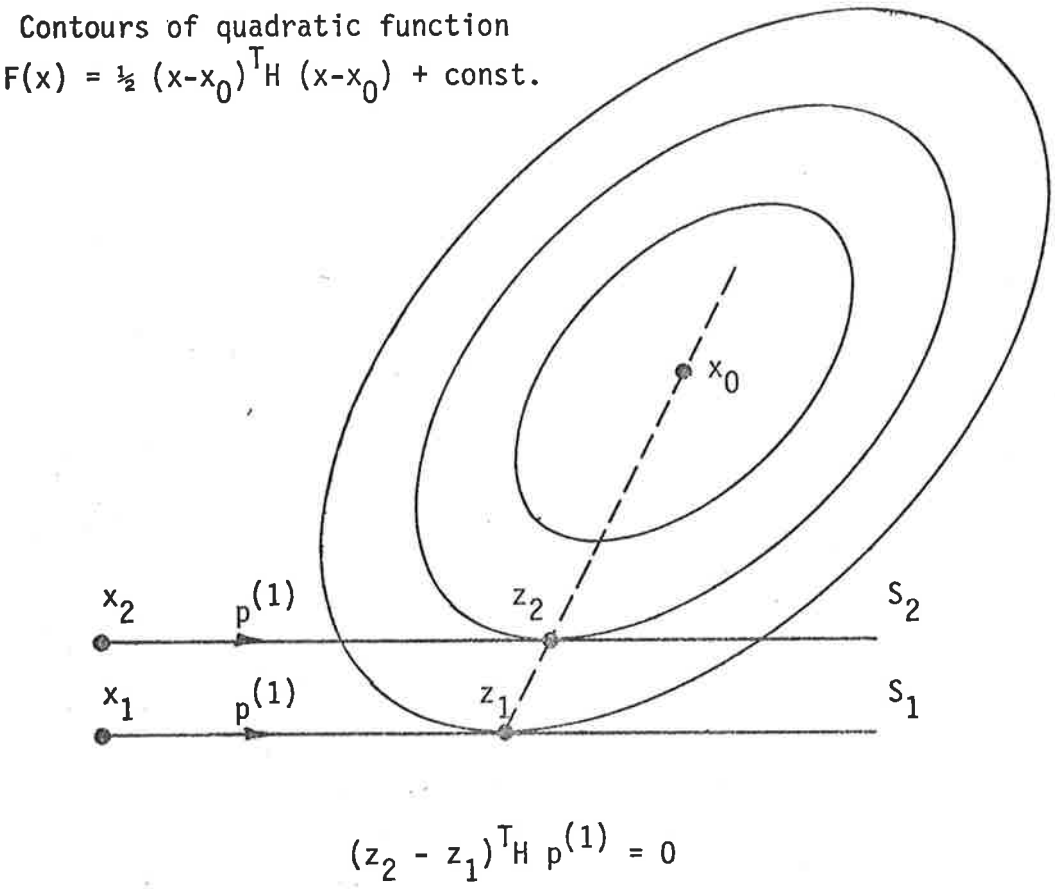


FIGURE 3.1 Illustrating the "Parallel Subspace" Conjugacy Property

and assume that the current parameter vector $\mathbf{x}^{(k-1)}$ has resulted from a line search (minimization) along $\mathbf{p}^{(N)}$. (To start the process such a search is performed as a preliminary step). The general kth iteration consists of the following procedure. A point $\mathbf{z}^{(k)}$ is generated by starting from point $\mathbf{x}^{(k-1)}$ and making line searches (one-dimensional minimizations) along each of the directions $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)}$ ... $\mathbf{p}^{(N)}$ in turn. The vector $\mathbf{p}^{(1)}$ is then discarded, being replaced by $\mathbf{p}^{(2)}$, $\mathbf{p}^{(2)}$ is replaced by $\mathbf{p}^{(3)}$, and so on, and $\mathbf{p}^{(N)}$ is replaced by the new vector $\mathbf{z}^{(k)} - \mathbf{x}^{(k-1)}$. A line search is then performed along the new $\mathbf{p}^{(N)}$, generating point $\mathbf{x}^{(k)}$ and completing the iteration.

This method represents a philosophy which has often been fruitful - methods which theoretically minimize a quadratic form in a finite number of steps have been generalized so that general functions can be handled. Convergence properties are often excellent.

There is one difficulty with the original form of Powell's algorithm. The set of search vectors sometimes tends to become linearly dependent, preventing the true minimum being reached. Powell's suggestion is to omit the replacement of a vector by $\mathbf{z}^{(k)} - \mathbf{x}^{(k-1)}$ if this would make the set "more" dependent. There have also been other remedies suggested, for example by Zangwill (1967).

Powell's method (among others) is critically dependent on the method used for the one-dimensional line searches, a matter which is taken up in Section 3.5.

3.3.3 Direct Search Methods

The methods of Sections 3.3.1 and 3.3.2 employ only function values but in ways which depend on the properties of differentiable functions. A third class of methods is based on the determination of a good direction in which to search merely by comparing the function values at a set of points. None of these methods has the theoretical elegance of the gradient-based algorithms, but they do have the advantages of simplicity and applicability to functions which are not differentiable, or which are the result of some physical measurement and therefore subject to random error.

An obvious approach is to minimize with respect to each of the independent variables in turn. The immediate objection, as seen from figure 3.2, is that the method will proceed in ever-decreasing steps along a "valley" (such as the major axis of an elliptical contour system) unless such a valley happens to be aligned with a coordinate direction. The practical direct-search algorithms are based on various methods for aligning the direction of progress with such local valleys.

In the *pattern search* method of Hooke and Jeeves (1961) a series of exploratory moves are made about the present *base point* by increasing or decreasing each of the variables in turn by a small amount (which need not be the same for each variable). Any step which gives an improvement (decrease) in function value is accepted, thus the method is sensitive to the order in which the coordinates are enumerated. When all variables have been so treated, an attempt is made to *extrapolate* to a forward point which is twice as far from the base point as the point found by exploration, it being assumed that a good direction has been

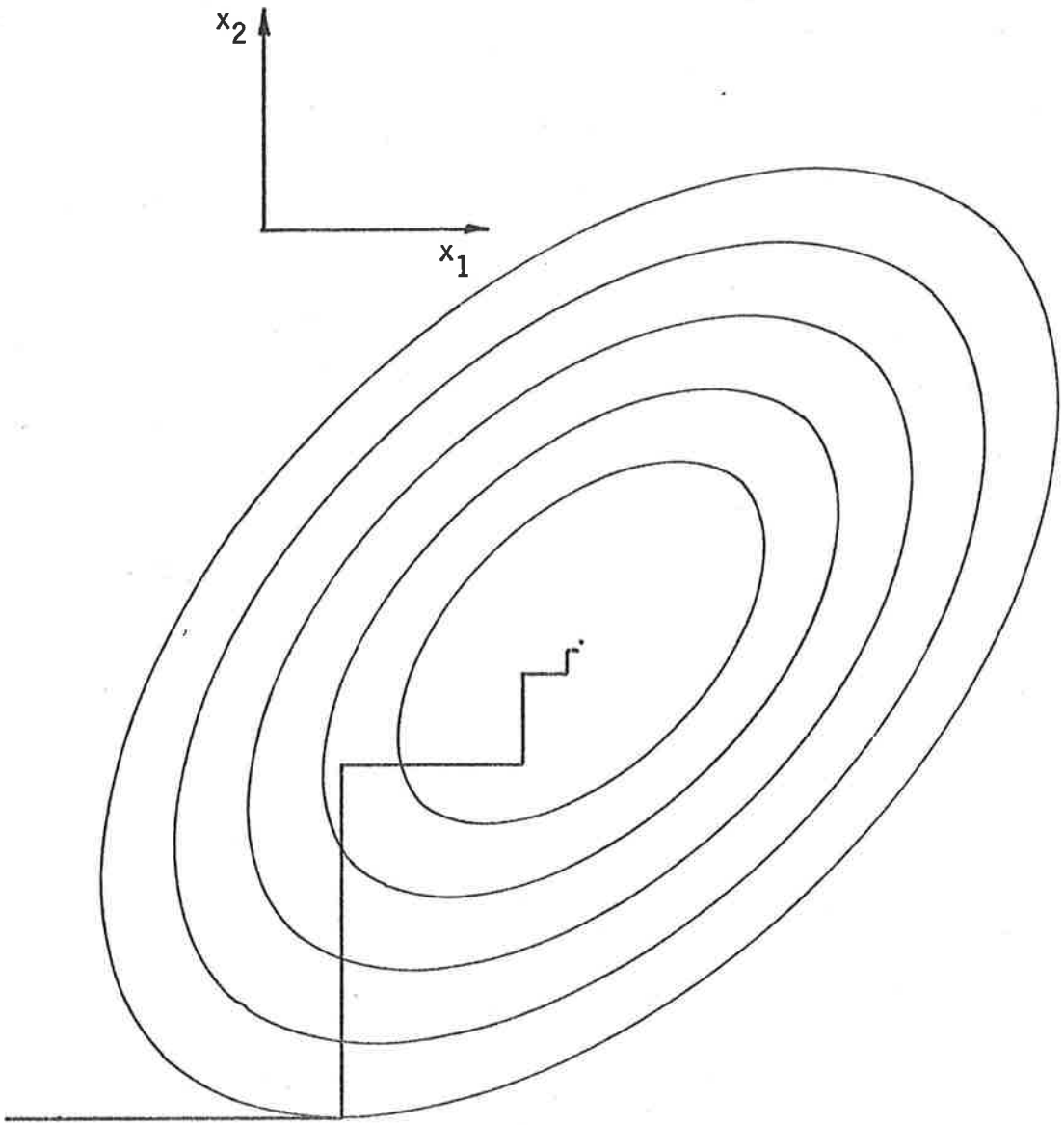


FIGURE 3.2 Illustrating the "Successive Variables" Minimization Scheme

roughly localized by the exploration. Hooke and Jeeves do not require that this extrapolation itself succeed in decreasing the function value, but allow a further series of exploratory moves from the forward point in an attempt to do this. If successful, the new point becomes the *base point*, and another attempt is made to extrapolate from the previous base point through the current one. Such *pattern moves*, if they are continually successful, have the property of growing in size, and the method is very efficient at following long, straight valleys.

As soon as a pattern move fails, the forward point of the exploration phase around the base point becomes the new base point. When no move in any coordinate direction causes a function decrease (and so no new base point can be generated), the size of the exploration steps is reduced. The entire algorithm terminates when this step becomes smaller than some preset size.

The method of Rosenbrock (1960) also proceeds by exploratory moves in a set of N mutually orthogonal directions, but these are not the coordinate directions (except on the first iteration). Exploration can proceed in each of the directions a number of times, a success for a given direction resulting in a larger step to be taken next time. When all directions fail, the overall direction of net progress becomes the first direction to be used on the next iteration, and the remaining N - 1 directions are calculated by the Gram-Schmidt orthonormalization process.

The conceptually simple method suggested by Spendley, Hext and Himsworth (1962) uses the properties of a regular simplex (a set of N + 1 equidistant points in N-dimensional space). A new simplex can be formed on any face of the old by the addition of only one new point, that which is the reflection of any vertex

through the centroid of the remaining N. At each stage of the process the *worst* vertex is abandoned and replaced by its reflection, except that if the most recent point is the worst, the next worst is used instead (to prevent continuous oscillation between two simplexes). Eventually, with one vertex in the vicinity of a local minimum, the successive simplexes will tend to rotate about this one "permanent" point - a signal to reduce the simplex size. The algorithm terminates when this size reaches a preset minimum, or the function value is acceptably small at all vertices.

In the method of Nelder and Mead (1965), the regularity of the simplexes is abandoned so that acceleration steps (similar to Hooke-Jeeves pattern moves) can be introduced. The simple reflection is only accepted if the resulting point is not a new best or worst point. If a best point, the simplex is expanded in the supposed good direction and the distant vertex used if it is better again. If the "reflection" point is the *worst*, a contraction of the simplex is made (affecting only the new point). If the "contracted" vertex is *still* the worst, only the best point of the existing simplex is retained and an overall halving of simplex size is made. Convergence is assumed when the variation in function value is sufficiently small over all points in the simplex.

3.3.4 Special Methods for Sums of Squares

In many cases of practical interest the objective function has the special form

$$F(x) = \sum_{m=1}^{M} f_m^2(x) \qquad (3.3)$$

as in cases of least-squares curve fitting. Section 3.4.12 treats several methods using first derivatives of the individual functions $f_m(x)$ which exploit this special form. There exist also methods which estimate these derivatives from values of the functions only. Obviously, a formula such as

$$\frac{\partial f_m(x)}{\partial x_i} = \frac{f_m(x + a\ e_i) - f_m(x)}{a} \qquad (3.4)$$

where e_i is the ith unit vector, could be used, but there are better ways which rely on the function values obtained on previous iterations. Once found, the approximate derivatives may be used in several ways, as will be outlined for *analytic* derivatives in Section 3.4.12.

The best known algorithms in this class are those of Powell (1965), Barnes (1965), Broyden (1965), Peckham (1970) and Powell (1970).

3.4 Gradient Methods

3.4.1 Introduction

A large class of methods for unconstrained minimization (in fact, the majority of them) employ iterations which may be expressed by the equations:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}\ p^{(k)} \qquad (3.5)$$

and

$$A^{(k)}\ p^{(k)} = -\ g^{(k)} \qquad (3.6)$$

where $g^{(k)}$ is the vector of first derivatives of the objective function F, evaluated at $x = x^{(k)}$, that is

$$\mathbf{g}^{(k)T} = \left| \frac{\partial F}{\partial x_1} \frac{\partial F}{\partial x_2} \cdots \frac{\partial F}{\partial x_N} \right| \tag{3.7}$$

and $\mathbf{A}^{(k)}$ is some N x N matrix. The superscript (k) refers to values preceding or used on the kth iteration. In what follows, it will be omitted whenever the connection with a particular iteration is immaterial.

Equation (3.6) defines a *search direction* $\mathbf{p}$ in terms of the gradient vector $\mathbf{g}$ and the matrix $\mathbf{A}$, and (3.5) implies that the next estimate for the minimizing parameter vector $\mathbf{x}^{(k+1)}$ is determined from $\mathbf{x}^{(k)}$ by taking a step in the direction of $\mathbf{p}^{(k)}$ and of length determined by the (scalar) *steplength parameter* $\alpha^{(k)}$.

In this thesis the term "gradient method" will be taken to include any method whose iterations may be expressed by (3.5) and (3.6) regardless of whether the computational scheme itself actually involves equations of this form. The methods differ in the ways in which the matrix $\mathbf{A}$ and the steplength α are chosen. Both first and second-derivative methods are included (second derivatives can enter via the matrix $\mathbf{A}$) and, indeed, a "function value only" method could qualify if the values were used expressly to compute finite-difference approximations to derivatives to be used in (3.6).

3.4.2 Descent Directions

In using a gradient method, one aims to cause a reduction in the value of the objective function F at each step in the hope that the sequence of iterates $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ... will thereby be forced to converge to a local minimum of F. Most methods restrict the matrix $\mathbf{A}$ to be positive definite, which ensures that a function reduction may be achieved for some

positive value of α, that is, that $\mathbf{p}$ is a *descent direction*. The proof of this is straightforward - since $\mathbf{A}$ is positive definite, its inverse $\mathbf{A}^{-1}$ exists and is itself positive definite. Therefore, from (3.6),

$$\mathbf{p} = - \mathbf{A}^{-1} \mathbf{g} . \tag{3.8}$$

Consider the (vector) Taylor series expansion of F for a small displacement $\Delta\mathbf{X}$ from $\mathbf{X}$, i.e.

$$F(\mathbf{X} + \Delta\mathbf{X}) = F(\mathbf{X}) + \mathbf{g}^T \Delta\mathbf{X} + \tfrac{1}{2} \Delta\mathbf{X}^T \mathbf{H} \Delta\mathbf{X} + O (\| \Delta\mathbf{X} \|^3 \tag{3.9}$$

where $\mathbf{g}$ and $\mathbf{H}$ are the gradient vector and Hessian matrix. If the displacement is in the direction $\mathbf{p}$, i.e.

$$\Delta\mathbf{X} = \alpha \mathbf{p}$$

then this becomes

$$F(\mathbf{X} + \alpha\mathbf{p}) - F(\mathbf{X}) = \alpha \mathbf{g}^T \mathbf{p} + \tfrac{1}{2} \alpha^2 \mathbf{p}^T \mathbf{H} \mathbf{p} + O(\alpha^3) \tag{3.10}$$

For sufficiently small α, the first term on the right-hand-side predominates, and substituting for $\mathbf{p}$ from (3.8),

$$F(\mathbf{X} + \alpha\mathbf{p}) - F(\mathbf{X}) \approx - \alpha \mathbf{g}^T \mathbf{A}^{-1} \mathbf{g} \tag{3.11}$$

Since $\mathbf{A}^{-1}$ is positive definite, the right-hand-side is negative for positive α, proving that a function reduction is possible if α is small enough. (Provided that $\mathbf{g}$ is not the zero vector,

in which case (3.6) would fail to generate a search direction at all).

3.4.3 Steepest Descent

The simplest gradient method takes **A** to be the identity matrix **I** , that is, **p** is simply equal to the negative gradient **-g** . In one variant of steepest descent, an initial guess is made for α, and the function evaluated. If the function is not decreased, then α is halved (or multiplied by some other factor less than 1) and the process repeated. Ultimate success is guaranteed (by the theorem of Section 3.4.2), and the point just found is taken as the start point for the next iteration. In another version, called "optimal" steepest descent by Gottfried and Weisman (1973), the value of α which minimizes F along the direction **-g** is actually found to a fair degree of precision.

At first sight a steepest descent method would seem to be the most efficient way of achieving a decrease in the function value, since (as the terminology implies) the function decreases most rapidly in this direction. However, this optimality holds only in a local (infinitesimal) sense, and the method is a very poor global strategy for minimizing most functions of practical importance. The principal reason is that the search usually becomes trapped in a relatively steep-sided "valley" whose floor has a gradual slope towards one end (to use the two-dimensional contour analogy). The steepest descent direction from a point on the wall of such a valley is almost perpendicular to the axis of the valley, and so the iterations will zig-zag or "hemstitch" from one side to the other with very little forward progress (down the axis of the valley).

The "optimal" approach (locating the valley floor on each iteration) is only a partial remedy (although it would appear effective in a simple, straight, two-dimensional valley). For one thing, the one-dimensional minimization is seldom carried out to extreme accuracy (to avoid waste of computer time better spent on more iterations), causing over or under-shoot with the consequences that the next steepest descent direction still has a large component across the valley. More importantly, however, even if the dimensionality of the problem is only moderately high, the opportunity for curvature of the contours in many directions simultaneously forces most of the minimizing steps to be very short. The end result is the same; very slow convergence.

The existence of such valleys is not at all uncommon; in fact any local minimum of a function will lie in one, except in the most unusual case that the function value depends almost equally on each variable. Virtually every author who has tried the method reports the same disappointing convergence, for example Marquardt (1963) and (for a digital filter design problem) Cadzow (1976). The method is not much used alone because of this very serious deficiency, although it is capable of producing large function decreases in the early iterations far from the optimum, and is sometimes resorted to as a "re-start" step when another algorithm has bogged down for some reason.

3.4.4 The Newton (or Newton-Raphson) Method

The classical Newton method takes **A** as the Hessian matrix of the objective function **H** (the matrix of second partial derivatives). It should be noted that **p** is not necessarily a descent direction since **H** is not restricted to be positive

definite. Straightforward application of Newton's method may then require a search in the negative direction of **p** to secure a function decrease. However, in the neighbourhood of a strong local minimum **H** *is* positive definite, and under such conditions Newton's method is an extremely efficient (quadratically convergent) process for finding the minimum. As such, it is the prototype for all second-derivative (and most other) gradient methods, which attempt to set **A** equal to **H** (or as close an approximation to **H** as possible) once this region of positive-definiteness has been located.

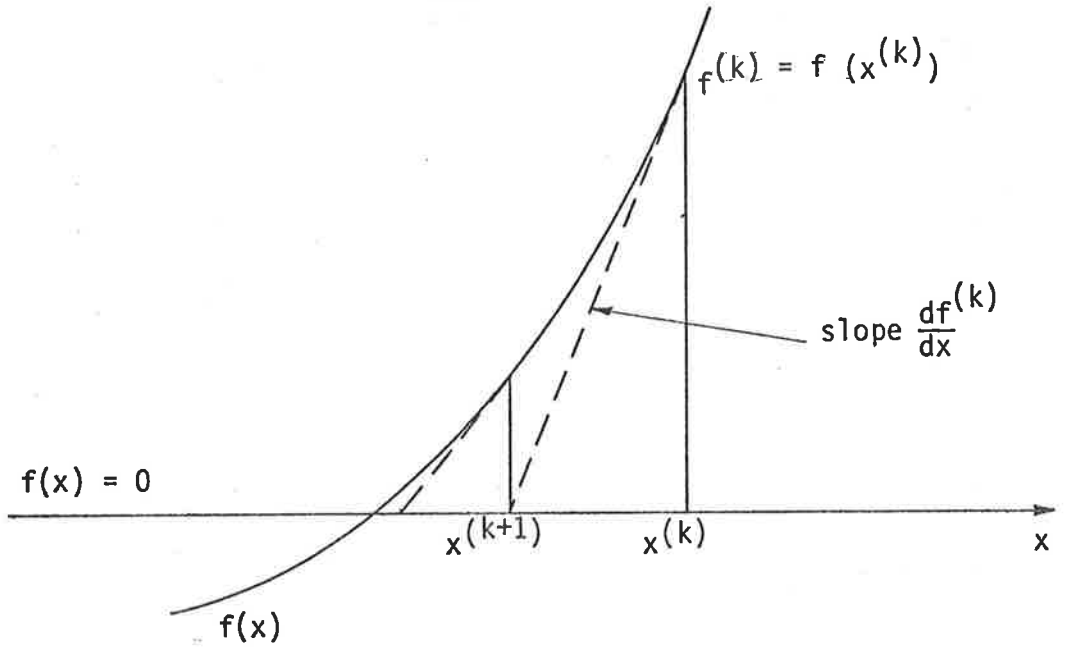
Newton's iteration may be derived from several viewpoints; two being given below:

(i) Vector Generalization of Newton's Method For Finding a Zero of a Function

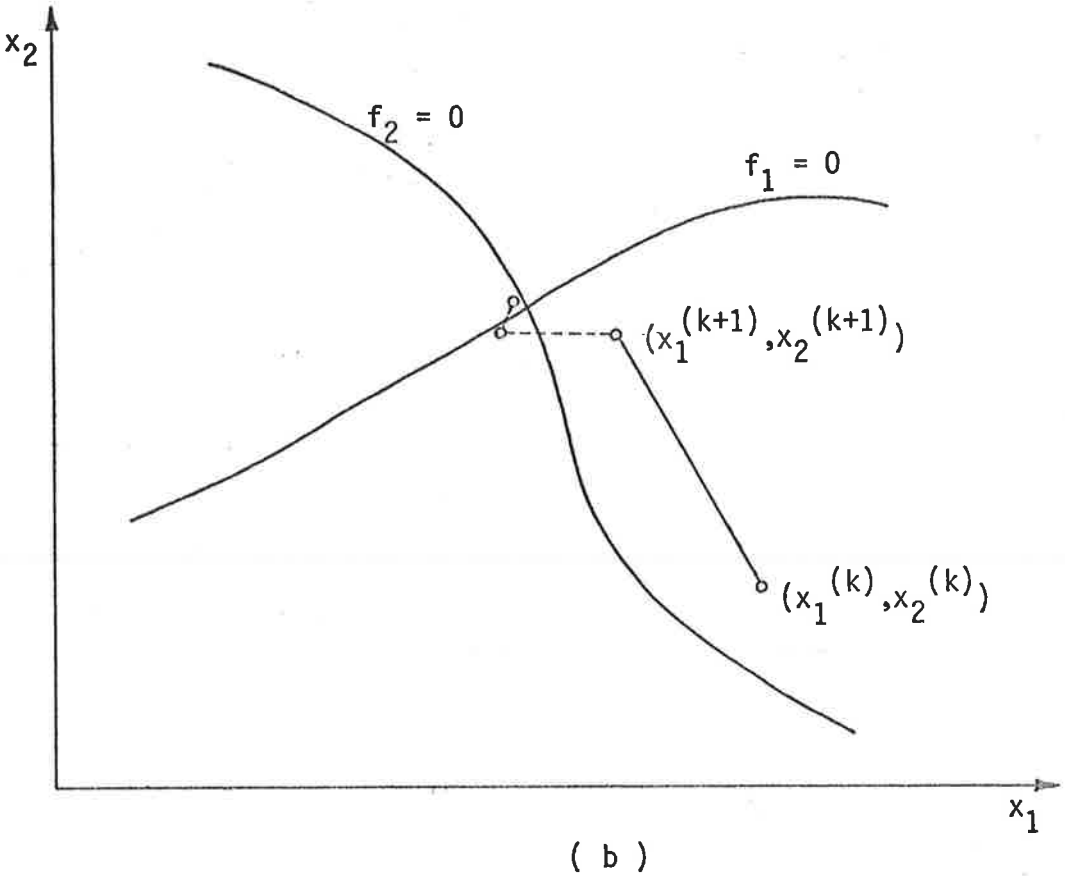
The well-known Newton-Raphson method for the iterative solution of $f(x) = 0$ is illustrated in figure 3.3 (a). The function value and gradient at $x = x^{(k)}$ define the new iterate according to the formula

$$x^{(k+1)} = x^{(k)} - \left[\frac{df}{dx}^{-1}\right]^{(k)} f(x^{(k)}) \qquad (3.12)$$

The rapid convergence which characterizes this iteration is obvious in the geometrical illustration. Figure 3.3 (b) illustrates a generalization where we are trying to find the values of two independent variables x_1 and x_2 which produce simultaneous zeros of two functions $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. In other words, we want the intersection point of the two loci $f_1 = 0$ and $f_2 = 0$ in the (x_1, x_2) plane. Starting at an arbitrary point $(x_1^{(k)}, x_2^{(k)})$ we generate a new point $(x_1^{(k+1)}, x_2^{(k+1)})$ which would zero both functions if the



(a)



(b)

FIGURE 3.3 Newton-Raphson Method for a Zero of a Function

(a) One dimension (b) Two dimensions

partial derivatives $\frac{\partial f_1}{\partial x_1}$, $\frac{\partial f_1}{\partial x_2}$, $\frac{\partial f_2}{\partial x_1}$ and $\frac{\partial f_2}{\partial x_2}$ remained constant at their $x^{(k)}$ values (note that this is also the process involved in the one dimensional case, figure 3.3 (a)). The equations to be satisfied are

$$f_1^{(k)} + (x_1^{(k+1)} - x_1^{(k)}) \frac{\partial f_1}{\partial x_1}^{(k)} + (x_2^{(k+1)} - x_2^{(k)}) \frac{\partial f_1}{\partial x_2}^{(k)} = 0 \qquad (3.13)$$

and

$$f_2^{(k)} + (x_1^{(k+1)} - x_1^{(k)}) \frac{\partial f_2}{\partial x_1}^{(k)} + (x_2^{(k+1)} - x_2^{(k)}) \frac{\partial f_2}{\partial x_2}^{(k)} = 0 \qquad (3.14)$$

which lead to

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+1)} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k)} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}^{-1(k)} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}^{(k)} \qquad (3.15)$$

provided that the matrix inverse indicated exists. This has the same form as (3.12), with vectors replacing scalars and a matrix inversion replacing the reciprocal operation.

A necessary condition for a local minimum of a function is that all gradient components are simultaneously zero. Identifying the functions f_1 and f_2 of the previous discussion with the gradient components $\frac{\partial F}{\partial x_1}$ and $\frac{\partial F}{\partial x_2}$, (3.15) becomes

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+1)} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k)} - \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_2 \partial x_1} \\ \frac{\partial^2 F}{\partial x_1 \partial x_2} & \frac{\partial^2 F}{\partial x_2^2} \end{bmatrix}^{-1(k)} \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \end{bmatrix}^{(k)} \qquad (3.16)$$

or, in vector-matrix terms

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{H}^{-1}{}^{(k)}\,\mathbf{g}^{(k)} \qquad (3.17)$$

which is precisely the same as (3.5) and (3.6) with $\mathbf{A}^{(k)} = \mathbf{H}^{(k)}$ and $\alpha^{(k)} = 1$. Analogous results hold for N dimensions. Of course, the order of partial differentiation is immaterial for "ordinary" functions and so the Hessian matrix is symmetric. The vanishing of the gradient vector is a necessary but not sufficient condition for a local minimum - the uncritical use of (3.17) may lead to convergence to a saddle point, or to a local maximum (or divergence). However, if $\mathbf{H}$ is positive definite at $\mathbf{x}^{(k)}$, the second-degree Taylor series approximation to $F(\mathbf{x})$ in the neighbourhood of $\mathbf{x}^{(k)}$ is a quadratic form with a minimum, and (3.17) finds this minimum.

(ii) Behaviour of the Newton Iteration on a Positive-Definite Quadratic Form

That Newton's iteration will find the stationary point of a quadratic function in one step is obvious from the above analysis, because for such a function the partial derivatives in (3.13) and (3.14) *are* constant for all $\mathbf{x}$. However, it is interesting to consider the function F explicitly defined as

$$F(\mathbf{x}) = b + \tfrac{1}{2}(\mathbf{x} - \mathbf{r})^{T}\,\mathbf{H}\,(\mathbf{x} - \mathbf{r}) \qquad (3.18)$$

which, if $\mathbf{H}$ is positive definite, has a minimum value of b at $\mathbf{x} = \mathbf{r}$. The gradient is given by

$$\mathbf{g}^{(k)} = \mathbf{H}\,(\mathbf{x}^{(k)} - \mathbf{r}) \qquad (3.19)$$

and so from (3.8)

$$\mathbf{p}^{(k)} = - \mathbf{H}^{-1} \mathbf{g}^{(k)} = - \mathbf{x}^{(k)} + \mathbf{r} \tag{3.20}$$

and

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{p}^{(k)} = \mathbf{r} \quad \text{if } \alpha^{(k)} = 1 \tag{3.21}$$

The stationary point $\mathbf{x} = \mathbf{r}$ is thus found in one step. Examining the change in the function value given by (3.10), where all terms above second order now vanish, we have, as a function of α,

$$\begin{aligned} \Delta F(\alpha) &= \alpha \, \mathbf{g}^T \mathbf{p} + \tfrac{1}{2} \alpha^2 \, \mathbf{p}^T \mathbf{H} \, \mathbf{p} \\ &= - \alpha \, \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} + \tfrac{1}{2} \alpha^2 \mathbf{g}^T \mathbf{H}^{-T} \mathbf{H} \, \mathbf{H}^{-1} \mathbf{g} \\ &= - (\alpha - \tfrac{1}{2} \alpha^2) \, \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \end{aligned} \tag{3.22}$$

If $\mathbf{H}$ (and so $\mathbf{H}^{-1}$) is positive definite, the change in the function is a *reduction* for all $0 < \alpha < 2$ and is maximized when $\frac{d}{d\alpha} (\Delta F) = 0$, that is when $\alpha = 1$, in which case the function reduction is

$$\Delta F = - \tfrac{1}{2} \, \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \tag{3.23}$$

Substituting for $\mathbf{g}$ from (3.19) gives

$$\begin{aligned} F &= - \tfrac{1}{2} (\mathbf{x} - \mathbf{r})^T \mathbf{H}^T \mathbf{H}^{-1} \mathbf{H} (\mathbf{x} - \mathbf{r}) \\ &= - \tfrac{1}{2} (\mathbf{x} - \mathbf{r})^T \mathbf{H} (\mathbf{x} - \mathbf{r}) \end{aligned} \tag{3.24}$$

and so from (3.18), the function is reduced to b, its minimum value.

In practical cases of importance, the objective function is not, of course, quadratic. However, close to a local minimum a quadratic is a good approximation to "normal" functions, and any minimization method which converges slowly when applied to a quadratic function cannot be expected to do better on a general function. For this reason, tests of algorithms using quadratic functions are of considerable interest. Newton's method, with its one-step quadratic termination, provides the "benchmark", and another method cannot be rated "excellent" unless it can minimize a quadratic form with an amount of labour similar to one Newton iteration.

3.4.5 Practical Implementation of Newton's Method

Because of its quadratic convergence when the iterate is near the minimum, Newton's method is extremely powerful and is recommended by most authors (e.g. Murray (1972a)) in cases when the required second derivatives can be calculated without too much trouble. However, the following questions must be addressed when considering an implementation:

(i) What should be done when the Hessian matrix is not positive definite? The various answers to this question form the basis of the other second-derivative methods, discussed below. The two cases of singularity (or near singularity, causing numerical ill-conditioning even if $\mathbf{H}$ is actually positive definite), and indefiniteness, may or may not be considered separately. One possibility for nonsingular but indefinite $\mathbf{H}$ is to search the negative direction of $\mathbf{p}$ if the positive fails. This usually works, but there is little theoretical justification for it and better procedures are available. If $\mathbf{H}$ is near singular, steepest descent may be used (as indeed it may be for indefinite $\mathbf{H}$).

(ii) What should be done when the gradient is close to zero? If **H** is positive definite the answer is: stop, the problem is solved. If **H** is indefinite and **g** exactly zero, then (3.8) fails to generate a search direction - the process is trapped at a saddle point. The likelihood of this happening seems extremely small, and some such strategy as a small random perturbation should cure it. However, some authors refer to it, for example Gill and Murray, (1974). They point out that the availability of second derivative information allows identification of a direction of *negative curvature* along which the function will eventually decrease.

(iii) When **H** is positive definite, how should the steplength be determined? Equation (3.17) indicates that α should be equal to unity, but the positive definiteness of **H** does not guarantee that a function decrease will be achieved for this value of α. The actual behaviour of the function F may differ significantly from that of the implied local quadratic approximation. However, as the local minimum is approached the value $\alpha = 1$ becomes very nearly optimal, and so the algorithm should be coded so as to try $\alpha = 1$ first and then proceed to smaller values of α if necessary.

3.4.6 Survey of Modified Newton Algorithms

Many authors have discussed schemes for the modification of Newton's method to handle cases where **H** is indefinite or singular. Goldstein and Price (1967) and Dixon and Biggs (1970) ignore the information in the Hessian in such cases and use a steepest-descent step. Goldfeld, Quandt, and Trotter (1966) base a method on the minimization (although in fact their paper deals with the equivalent maximization problem) of the local quadratic

approximation over a restricted region within which the approximation is known to be adequate. Their method uses a (hyper) spherical region centred on the current iterate and demands a complete eigensystem analysis of $\mathbf{H}$. A similar method of Fletcher (1972) uses a hypercube region and requires solution of a quadratic program at each iteration.

An interesting method of Botsaris and Jacobson (1976) uses in place of (3.5) and (3.6) the iteration

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{p}^{(k)}\left(\mathbf{x}^{(k)}, t^*\right) \tag{3.25}$$

where t^* is selected to minimize $F(\mathbf{x})$ along the *curve* given by

$$\mathbf{x} = \mathbf{x}^{(k)} + \mathbf{p}^{(k)}\left(\mathbf{x}^{(k)}, t\right) \tag{3.26}$$

where

$$\mathbf{p}^{(k)}\left(\mathbf{x}^{(k)}, t\right) = \left[\sum_{i=1}^{N} \frac{e^{-\lambda_i^{(k)} t} - 1}{\lambda_i^{(k)}}\, \mathbf{u}_i^{(k)}\, \mathbf{u}_i^{T(k)}\right] \mathbf{g}^{(k)} \quad . \tag{3.27}$$

$\lambda_i^{(k)}$ and $\mathbf{u}_i^{(k)}$ are the ith eigenvalue and ith normalized eigenvector of $\mathbf{H}^{(k)}$, respectively. In the case of any $\lambda_i = 0$, the coefficient of $\mathbf{u}_i\ \mathbf{u}_i^T$ is replaced by $-t$, its limit as $\lambda_i \to 0$. For $t = \infty$, the iteration is just Newton's method, and the authors try this particular value first, proceeding with the minimization over t only if the function value is not decreased for $t = \infty$. As t tends toward zero the step $\mathbf{p}\left(\mathbf{x}^{(k)}, t\right)$ tends to a small step in the steepest-descent direction, and so a function reduction *will* be obtained for some suitably small positive t. In this respect the method is similar to that of Marquardt (Section 3.4.8) which also generates a sequence of

of "interpolations" between the Newton and steepest-descent directions. When $\mathbf{H}^{(k)}$ is nonsingular but indefinite (i.e. $\lambda_i^{(k)} < 0$ for some i), the value $t = \infty$ is inappropriate in (3.27), and Botsaris and Jacobson have used an "undamped Greenstadt" trial ((3.27) with $t = \infty$ and all negative eigenvalues replaced by their absolute values) before undertaking the minimization over t.

None of the methods discussed above formally fit the model of equations (3.5) and (3.6) in the sense that a search direction $\mathbf{p}^{(k)}$ is first determined, followed by selection of a suitable $\alpha^{(k)}$ which determines a point in *this direction*. Rather, α is always unity and both the direction and length of $\mathbf{p}$ are found by a univariate search over some parameter. Some methods which *do* fit these equations are discussed in the sections which follow.

It should be remarked that eigensystem analysis is undesirable since it is relatively time-consuming and the storage of eigenvectors requires a full N x N array. To be competitive, such a method would need to be considerably more efficient (as regards number of iterations required) than those methods not requiring such analysis. This superior efficiency does not seem to be achievable (this thesis, Chapter five).

3.4.7 Greenstadt's Method

Greenstadt (1967) considered the case when $\mathbf{H}$ is positive definite but one eigenvalue λ_i is much smaller than the rest (all eigenvalues of a positive definite matrix are positive). The function then has a "valley" closely parallel to $\mathbf{u}_i$, the eigenvector corresponding to λ_i. Then, provided that the gradient

g has at least a small component *along* the valley, the Newton step is substantially along the valley, as desired, even when **x** is some distance from the valley floor. However, if a small perturbation shifts the small eigenvalue to be negative (leaving the others *relatively* untouched even though changed by comparable absolute amounts) the Newton step is *reversed*, although the best minimum-seeking direction is unchanged. Such reasoning led Greenstadt to suggest that negative eigenvalues should be replaced by their absolute values in the determination of a search direction, i.e.

$$\mathbf{A}^{(k)} = \sum_{i=1}^{N} |\lambda_i^{(k)}| \; \mathbf{u}_i^{(k)} \; \mathbf{u}_i^{(k)\,T} \quad . \tag{3.28}$$

The method requires the eigensystem analysis of **H** at each iteration, but the inversion of **A** to obtain **p** from equation (3.8) is simple because the eigenvalues of $\mathbf{A}^{-1}$ are the reciprocals of those of **A**, whereas the eigenvectors are the same. **p** can therefore be computed from

$$\mathbf{p} = -\left[\sum_{i=1}^{N} \frac{1}{|\lambda_i|} \; \mathbf{u}_i \; \mathbf{u}_i^{T} \right] \mathbf{g} \; . \tag{3.29}$$

The Greenstadt iteration is completed by some suitable line search procedure (Section 3.5) to determine an acceptable α along **p**.

Murray (1972a) suggests that if **H** is singular (that is, has some $\lambda_i = 0$, so that (3.29) is not computable) then the value $2^{t/2}$ where t is the "relative machine precision" be used in place of $|\lambda_i|^{-1}$. He also suggests a scheme to dispense with the eigensystem analysis and so save computer time when **H** is

positive definite. In such a case the system of linear equations (3.6), with $\mathbf{A} = \mathbf{H}$, can be solved by the Cholesky decomposition of $\mathbf{H}$ into the form

$$\mathbf{H} = \mathbf{L}\,\mathbf{D}\,\mathbf{L}^T \tag{3.30}$$

where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{D}$ is a diagonal matrix. This is followed by two back-substitution steps and a simple scaling by the diagonal elements of $\mathbf{D}$ to obtain $\mathbf{p}$. This procedure is regarded (Wilkinson (1965)) as being the best method for solving symmetric systems when the matrix is positive definite. The method is numerically unstable when the matrix is indefinite (and the Cholesky factors may not even exist) but indefiniteness is revealed by the occurrence of a negative diagonal element of $\mathbf{D}$ during the attempted factorization. Murray's suggestion, then, is to try the Cholesky decomposition of $\mathbf{H}$ and proceed with the back-substitution steps if it succeeds. Otherwise, as soon as indefiniteness is apparent, the eigensystem analysis is performed and (3.29) applied. With the eigensystem analysis taking about 18 times the computer time necessary for Cholesky's method ($3N^3$ as against $1/6\ N^3 + O(N^2)$), there is a significant saving for large N if $\mathbf{H}$ is positive definite a reasonable proportion of the time, and in the worst case a degradation of only 6%.

3.4.8 The Marquardt or Levenberg Method

Consider an N by N symmetric matrix $\mathbf{A}$, having eigenvalues λ_i and corresponding eigenvectors $\mathbf{u}_i$, for i = 1, 2 ... N. Then, by definition,

$$\mathbf{A}\,\mathbf{u}_i = \lambda_i\,\mathbf{u}_i \tag{3.31}$$

Then, adding a multiple β of $\mathbf{u}_i$ to both sides, we get

$$(\,\mathbf{A} + \beta\,\mathbf{I}\,)\;\mathbf{u}_i = (\lambda_i + \beta)\;\mathbf{u}_i \tag{3.32}$$

from which it may be seen that the matrix $\mathbf{A} + \beta\,\mathbf{I}$ has the same eigenvectors as $\mathbf{A}$, and all eigenvalues shifted by an amount β. Hence, a positive definite matrix may be constructed from *any* symmetric matrix by adding a positive amount β, greater than the absolutely largest negative eigenvalue, to each of the diagonal elements. This forms the basis of several distinct minimization algorithms.

Firstly, a descent direction $\mathbf{p}$ may be determined from an indefinite or singular Hessian $\mathbf{H}$ by adding some amount β to the diagonal elements which is sufficient to ensure positive definiteness (and a reasonable condition number), that is, $\mathbf{p}$ is determined by solving the linear equation system

$$(\,\mathbf{H} + \beta\,\mathbf{I}\,)\;\mathbf{p} = -\mathbf{g} \tag{3.33}$$

The main difficulty is that a suitable value of β is not known a priori. However, we may proceed as Murray (1972a) suggested for Greenstadt's method, that is, attempt first the Cholesky decomposition of $\mathbf{H}$ so that the Newton direction may be used if $\mathbf{H}$ is positive definite. If this fails, we then try to perform the Cholesky factorization of $(\,\mathbf{H} + \beta\,\mathbf{I}\,)$ for some trial value of β. If this succeeds, a direction $\mathbf{p}$ is found (by back-substitution) and the trial β decreased by a factor of (say) 2 so that a smaller value will be tried on the next iteration.

If the factorization fails, β is increased by some factor (say 2 again) and the process repeated (with the exception that the trial value passed to the next iteration is halved only if the factorization succeeds the *first* time). It has been found in practice (this thesis) that more than three trials of β are seldom required except perhaps on the first few iterations when β has not yet matched itself to the eigenvalue spectrum of $\mathbf{H}$.

When a descent direction $\mathbf{p}$ has been determined, a line search (Section 3.5) is performed to find a point which reduces the function value. This method will be referred to as "Marquardt with line search." It appears not to have been investigated before, although experiments to be reported in this thesis indicate that it performs better than most alternatives.

The method described above has the objection that if the value of β tried happened to be only marginally greater than the absolute value of the most negative eigenvalue, the matrix $\mathbf{A} + \beta \mathbf{I}$ could be so ill-conditioned that $\mathbf{p}$ could fail to be a descent direction (due to numerical roundoff error). However, this situation is easily avoided by insisting that the diagonal elements of the $\mathbf{D}$ matrix be not only positive but exceed some suitable small positive quantity.

A method which is related to "Marquardt with line search", and much more widely appreciated, involves making the parameter β so large that not only is the matrix $(\mathbf{H} + \beta \mathbf{I})$ positive definite, but also that a function reduction is achieved at the point $\mathbf{x}^{(k+1)}$ found from

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{H}^{(k)} + \beta^{(k)} \mathbf{I})^{-1} \mathbf{g}^{(k)} \quad . \qquad (3.34)$$

In this way no line search is needed, but for every trial value of β which makes $(\mathbf{H} + \beta \mathbf{I})$ positive definite, we need to solve a new system of N linear equations (by Cholesky, or otherwise) and evaluate the function to see whether it is reduced. We can, as described above for the method with line search, use a "reasonable" first trial value passed from the preceding iteration. This method will be called the "true Marquardt" method, although the original work by Marquardt (1963) added β not to the diagonal elements of $\mathbf{H}$ but to those of an "approximate $\mathbf{H}$" which may be derived from first derivative information only in cases where the objective function F is a sum of squares. This is considered in Section 3.4.12. The method in fact dates from 1944, when Levenberg was led to the same procedure by different reasoning.

The addition of an amount β to each of the diagonal elements of $\mathbf{H}$, which at first may seem a rather arbitrary procedure, has at least the theoretical justification that as β tends to infinity, the matrix $(\mathbf{H} + \beta \mathbf{I})$ tends to a multiple of the identity matrix and so the displacement $\mathbf{p}^{(k)}$ $(= \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, as found from (3.34)) becomes a small step in the steepest descent direction. Marquardt's method thus shares with that of Botsaris and Jacobson (Section 3.4.6) the characteristic that the line search is replaced by a search for a function decrease along a *curve* in parameter space. The curve in each case has the property that it begins at the "Newton" point and spirals in towards the "base" point where it is tangential to the steepest descent direction. A further similar method was published by Jones (1970) in which the curve is given an arbitrary (but plausible) analytic form to enable successive trial points to be generated by vector

addition rather than by repeatedly solving systems of linear equations.

The necessity to solve a system of equations for each value of β can be dispensed with in Marquadt's method, too, as was pointed out both by Jones (1970) and by Bard (1970). However, the alternative is a complete eigensystem analysis of $\mathbf{H}$, in order to calculate a trial displacement $\mathbf{p}^{(k)}$ from

$$\mathbf{p}^{(k)}(\beta) = - \left[\sum_{i=1}^{N} (\lambda_i^{(k)} + \beta)^{-1} \mathbf{u}_i^{(k)} \mathbf{u}_i^{(k)T} \right] \mathbf{g}^{(k)} \quad . \tag{3.35}$$

Murray (1972a) points out that about 20 different values of β would be necessary for the method to represent a time saving, and Marquardt's method normally yields an acceptable point much sooner than this.

Marquardt's method both with and without line search may also be modified in another way. In most cases the value of the objective function will be much more sensitive to some components of the parameter vector than to others, and the components of the gradient vector and elements of the Hessian matrix will accordingly vary greatly in magnitude. Adding the *same* amount to each diagonal element of $\mathbf{H}$ could conceivably "swamp" some of the "true" elements whilst making only minimal changes to others. Some elements of the displacement vector could then be poorly determined by the resulting equations. Accordingly, it is often recommended either that the quantity added to diagonal element h_{jj} be not β but $\beta|h_{jj}|$, or that the problem first be *re-scaled* so that $\mathbf{H}$ has all diagonal elements of magnitude unity. The latter course involves dividing

the jth row and column of **H** by $\sqrt{|h_{jj}|}$, simultaneous scaling of both rows and columns being necessary to preserve symmetry. Dividing the jth element of **g** by the same quantity compensates for the row scaling, and column scaling is compensated for (eventually) by dividing the jth element of **p**, as found, by the same amount. Both of these approaches are equivalent in the sense that the "search spiral" is no longer tangent to the steepest descent direction as $\beta \to \infty$, but to a vector whose jth component is $-g_j|h_{jj}|^{-1}$. This essential difference in global characteristics over the unmodified Marquardt method may or may not be desirable, and in Chapter Five of this thesis the matter is investigated numerically for certain problems.

A possible advantage to be gained from scaling is that the numerical conditioning of the equations is likely to be better because the elements vary less in magnitude. The eigensystem analysis used in Greenstadt's method should also benefit from scaling in that accuracy will be gained, although the sequence of search directions will again be changed by the process. This matter also is investigated in Chapter Five.

3.4.9 The Method of Gill and Murray

A method originally proposed by Murray (1972a) and later explained in more detail by Gill and Murray (1974) takes the **A** matrix of equation (3.6) to be equal to **H** + **E**, where **E** is a diagonal matrix having non-negative elements, such that the matrix **A** is positive definite. When **H** itself is positive definite (and not nearly singular) then all elements of **E** are zero. The method resembles that of Marquardt in that the *diagonal* elements of **H** are perturbed but the method of achieving this is quite different and does not require "trial" values of a parameter.

As was mentioned in Section 3.4.7, a positive-definite symmetric matrix $\mathbf{H}$ admits of a Cholesky decomposition (or factorization)

$$\mathbf{H} = \mathbf{L}\,\mathbf{D}\,\mathbf{L}^T \tag{3.36}$$

where $\mathbf{L}$ is a lower triangular matrix with unit diagonal elements, and $\mathbf{D}$ is a diagonal matrix with positive diagonal elements. Because the diagonal elements of $\mathbf{D}$ are positive, the factorization could equally well be written

$$\mathbf{H} = \hat{\mathbf{L}}\,\hat{\mathbf{L}}^T \tag{3.37}$$

where

$$\hat{\mathbf{L}} = \mathbf{L}\,\mathbf{D}^{\frac{1}{2}} .$$

Denoting the elements of $\hat{\mathbf{L}}$ as l_{ij} and those of $\mathbf{H}$ as h_{ij}, and equating elements in the kth column of (3.37) we have

$$\sum_{i=1}^{k} l_{ki}^2 = h_{kk} \tag{3.38}$$

and

$$\sum_{i=1}^{k} l_{ji}\, l_{ki} = h_{jk} \qquad j = k+1,\ k+2 \ \ldots\ N. \tag{3.39}$$

Thus the elements of the matrix $\hat{\mathbf{L}}$ may be found, one column at a time beginning at k=1, from

$$l_{kk} = \left(h_{kk} - \sum_{i=1}^{k-1} l_{ki}^2\right)^{\frac{1}{2}} \tag{3.40}$$

and

$$l_{jk} = (h_{jk} - \sum_{i=1}^{k-1} l_{ji}\, l_{ki})/l_{kk} \quad , j = k+1, \ldots N \qquad (3.41)$$

which are rearrangements of (3.38) and (3.39). If $\mathbf{H}$ is *not* positive definite, the parethesized quantity in (3.40) will, for some k, become negative, and the process cannot proceed. However, if this does *not* happen, the factorization is numerically stable because the elements of $\hat{\mathbf{L}}$ are bounded by the diagonal elements of $\mathbf{H}$, as expressed by (3.38).

In the method of Gill and Murray, a lower triangular matrix $\hat{\mathbf{L}}$ is generated from $\mathbf{H}$ in a manner which is a modification of the above Cholesky process. Deliberate steps are taken at each stage of the process to ensure progress and numerical stability. The radicand in (3.40) is replaced by its absolute value if it is negative, and, further, the value of l_{kk} thus found is increased even more if necessary to ensure that *all* the elements l_{jk} (found from (3.41)) are bounded by a preassigned positive constant β.

In their paper, Gill and Murray show that such "tampering" results in a lower triangular matrix $\hat{\mathbf{L}}$ such that $\hat{\mathbf{L}}\,\hat{\mathbf{L}}^T$ is the true Cholesky factorization of a matrix $\mathbf{H} + \mathbf{E}$, where $\mathbf{E}$ is a diagonal matrix whose elements are bounded. Further, they derive an optimum value for the constant β which minimizes this bound and at the same time ensures that the true Cholesky factors of $\mathbf{H}$ are found (i.e. $\mathbf{E}$ is the zero matrix) whenever $\mathbf{H}$ is sufficiently positive definite.

A search direction $\mathbf{p}$ is then computed from the usual formula (3.6), which becomes

$$\hat{\mathbf{L}}\,\hat{\mathbf{L}}^T \mathbf{p} = -\mathbf{g}$$

and the process is simply one of back-substitution (twice). A line search is then performed, completing the iteration.

Although they suggest that convergence to a saddle point is very unlikely, Gill and Murray also point out that their process allows a direction of *negative curvature* to be identified so that a function reduction may be achieved even in cases when $\|\mathbf{g}\| = 0$, and the formula (3.6) fails. The derivation is given in the original paper. In practice, all that is required is that the index, k, which gives the smallest "radicand" in (3.40) during the process be recorded. If $\mathbf{H}$ is indefinite, then $\mathbf{p}$ found from

$$\hat{\mathbf{L}}^T \mathbf{p} = \mathbf{e}_k \qquad (3.42)$$

is a direction of negative curvature, where $\mathbf{e}_k$ is the vector with kth element unity and all the rest zero. In a practical algorithm, this alternative formula would be used whenever $\mathbf{H}$ were indefinite and $\|\mathbf{g}\| < \varepsilon$, for some preset threshold ε. The direction $-\mathbf{p}$ would then be used in place of $\mathbf{p}$ if $\mathbf{g}^T\mathbf{p} > 0$.

Matthews and Davies (1971) have proposed another method based on the $\mathbf{LU}$ factorization of $\mathbf{H}$ by Gaussian elimination, where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{U}$ an upper triangular matrix. During this process some of the diagonal elements u_{ii} will become negative if $\mathbf{H}$ is indefinite; the authors propose to force them positive and thus generate the factorization of a related positive definite matrix. Later authors (e.g. Murray, 1972a) do not favour the method because it is not numerically stable in general.

3.4.10 The Method of Fletcher and Freeman

Fletcher and Freeman (1977) have proposed a method which, in the case that **H** is indefinite, generates a search direction **p** which is not only a descent direction but also one of negative curvature. In this, they follow a suggestion of Fiacco and McCormick (1968). The rationale is that, if $F(\mathbf{x})$ is bounded below, the negative curvature must eventually change to positive for some sufficiently large value of the steplength α. Fiacco and McCormick argue that such a search is likely to lead, on the next iteration, to a Hessian matrix with one fewer negative eigenvalue. The method of Fiacco and McCormick was based on the Cholesky factorization, which for indefinite matrices can be unstable or even impossible. This objection is removed in the later work, which relies on a stable matrix factorization due to Bunch and Parlett (1971). These authors show that a factorization $\mathbf{L}\,\mathbf{D}\,\mathbf{L}^T$ can be constructed in a numerically stable manner, where **L** is unit lower triangular and **D** is *block diagonal* having blocks which are 1 by 1 and/or 2 by 2 only. The factorization is not of the original symmetric indefinite matrix **H** but of one which has been modified by symmetric row and column interchanges. When 2 by 2 blocks occur they are symmetric and can be restricted to having negative determinants, that is, one eigenvalue of each sign. Fletcher (1976) describes another strategy for generating a factorization of this type which maintains stability but allows a somewhat greater "rate of growth" of error, to gain a substantial reduction in computer time.

In the minimization method of Fletcher and Freeman such a factorization is computed. If all blocks of **D** are 1 by 1 and positive, the Hessian is positive definite and the factorization is used to generate an ordinary Newton search direction. If

there are any 2 by 2 blocks, or negative 1 by 1 blocks, a direction of negative curvature $\mathbf{t}$ is generated from

$$\mathbf{L}^T \mathbf{t} = \mathbf{a} \tag{3.43}$$

where the vector $\mathbf{a}$ has unit elements for those indices for which $\mathbf{D}$ has negative 1 by 1 blocks and zero elements corresponding to positive 1 by 1 blocks. Where (i, i+1) correspond to a 2 by 2 block of $\mathbf{D}$, then $[a_i, a_{i+1}]^T$ is set equal to the normalized eigenvector corresponding to the negative eigenvalue of the $\mathbf{D}$ block. After unravelling the column permutation which occurred during factorization (that is, making a compensating permutation to the elements of $\mathbf{t}$), the scalar product $\mathbf{g}^T\mathbf{t}$ is evaluated and tested, and the search direction $\mathbf{p}$ set to $\mathbf{t}$ or $-\mathbf{t}$ as required to obtain a descent direction.

Fletcher and Freeman recommend not that such a direction be used *whenever* $\mathbf{H}$ is indefinite, but that it should alternate with the use of a Newton-like step restricted to the subspace of directions of positive curvature. This is because the iterates could otherwise become restricted to a subspace and so fail to converge to a minimum. A vector $\mathbf{t}$ within the positive-curvature subspace is found from

$$\mathbf{L}\ \mathbf{D}^*\ \mathbf{L}^T \mathbf{t} = -\mathbf{g} \tag{3.44}$$

where $\mathbf{D}^*$ is $\mathbf{D}$ with negative eigenvalues replaced by zero.

The descent direction $\mathbf{p}$ is found by permuting the elements of $\mathbf{t}$ to unravel the permutation which occurred during factorization.

3.4.11 First-Derivative Methods

The foregoing sections have outlined most of the useful gradient methods which employ second derivative (Hessian matrix) information explicitly. They are, as a class, the most efficient nonlinear minimization methods known because of the excellent convergence properties of the basic underlying Newton-Raphsan iteration. However, in many practical cases, the task of analytically twice differentiating the objective function with respect to each pair of variables, and then generating computer code, is very intricate, time-consuming and error prone, and many methods have been developed which avoid the need. Of course, in cases such as the digital filter design considered in this thesis the objection is largely removed because the differentiation may be done (and carefully checked) once-and-for-all, hence, the fairly complete survey of second-derivative methods in the preceding sections. However, it is conceivable that a first-derivative method (which requires more iterations) could be more efficient than a second-derivative method (which requires more work per iteration).

First-derivative methods fall into three categories:

(a) those which generate some sort of approximation to the Hessian matrix **H** or its inverse at each iteration,

(b) those which employ several iterations (usually N) to simulate the effect of one "Newton" iteration (conjugate gradient methods), and

(c) those which use a matrix **A** which is updated at each iteration and eventually becomes an approximation to the Hessian (or inverse Hessian) i.e. the quasi-Newton or "modification" methods.

The latter two types are considered in Sections 3.4.13 and

3.4.14. Of type (a) there are those which generate approximations to second derivatives by differencing the first derivatives. The idea is obvious - the art to choose a steplength which nicely balances the cancellation and truncation errors. One such algorithm is given by Gill and Murray (1974). There is another important type which falls into category (a) and applies when the objective function is a sum of squares of other functions. This is now considered.

3.4.12 Special Methods for Sums of Squares, or "Gauss-Newton" Methods

Suppose that the objective function has the special form

$$F(\mathbf{x}) = \sum_{m=1}^{M} f_m^2(\mathbf{x}) \tag{3.45}$$

This is applicable to problems of curve fitting, system identification and digital filter design, where we seek to minimize the sum of squared differences between model-derived and experimentally measured quantities, or "achieved" and "desired" values. In such cases M could be greater than N. It is also applicable to the problem of solving simultaneous nonlinear equations, in which case M would be equal to N and the individual f_m would be the expressions required to be zero.

Differentiating (3.45),

$$\frac{\partial F}{\partial x_i} = \sum_{m=1}^{M} 2 f_m \frac{\partial f_m}{\partial x_i} \qquad i = 1, 2 \ldots N \tag{3.46}$$

and

$$\frac{\partial^2 F}{\partial x_i \partial x_j} = \sum_{m=1}^{M} 2 \frac{\partial f_m}{\partial x_i} \frac{\partial f_m}{\partial x_j} + \sum_{m=1}^{M} 2 f_m \frac{\partial^2 f_m}{\partial x_i \partial x_j} \qquad \begin{matrix} i = 1, 2 \ldots N \\ j = 1, 2 \ldots N \end{matrix} \qquad (3.47)$$

which give expressions for the gradient $\mathbf{g}$ and Hessian $\mathbf{H}$ to be used as in Newton's method. The special methods are based on ignoring the second sum in the expression for the elements of the Hessian, in the hope that it will be small compared with the first. This is equivalent to approximating each of the component functions $f_m(\mathbf{x})$ by the first-order terms in its Taylor series expansion about $\mathbf{x}^{(k)}$. The method is especially successful in the values of the "residuals" f_m are zero at the solution (as when solving non-linear equations) because then the second sum *does* tend to zero, and Newton-like convergence is ultimately obtained.

The matrix whose elements are given by the first sum in equation (3.47) may be written as $2\,\mathbf{J}^T\mathbf{J}$, where $\mathbf{J}$ is the M by N Jacobian matrix, given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \\ \vdots & & \vdots & \\ \frac{\partial f_M}{\partial x_1} & & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} \qquad (3.48)$$

The matrix $2\,\mathbf{J}^T\mathbf{J}$ will also be denoted as $\mathbf{R}$. Now considering the quadratic form

$$
\begin{aligned}
Q &= \mathbf{y}^T (\mathbf{J}^T \mathbf{J}) \mathbf{y} \\
&= (\mathbf{y}^T \mathbf{J}^T)(\mathbf{J} \mathbf{y}) \\
&= (\mathbf{J} \mathbf{y})^T (\mathbf{J} \mathbf{y}) \\
&= \sum_{m=1}^{M} (\mathbf{J} \mathbf{y})_m^2
\end{aligned}
$$

which is obviously greater than or equal to zero, it is seen that $\mathbf{R}$ is at least positive semidefinite. It must be positive definite unless $(\mathbf{J} \mathbf{y})_m = 0$ for all m with some non-zero vector $\mathbf{y}$; that is unless the N columns of $\mathbf{J}$ are linearly dependent. Hence, unless $\mathbf{J}$ fails to have full rank (N), the matrix $\mathbf{R}$ is positive definite, and a descent direction may in principle always be generated by choosing the matrix $\mathbf{A}$ of equation (3.6) to be equal to $\mathbf{R}$. However, it does not infrequently happen that $\mathbf{R}$ becomes indefinite due to roundoff error, and practical algorithms must take this into account.

The classical, or undamped, "Gauss-Newton" iteration uses (3.5) and (3.6) with $\mathbf{A}=\mathbf{R}$ and $\alpha = 1$. It is not guaranteed to converge because the approximations involved are not necessarily valid for large step sizes, and the "damped Gauss-Newton" algorithm, employing a line search to guarantee a function reduction, is much more reliable.

If the matrix $\mathbf{R}$ is not positive definite, however, due to roundoff error when $\mathbf{J}$ is close to being rank-deficient, even this may fail. Accordingly, the standard method for least-squares problems has come to be that of Marquardt (1963), already discussed at length in Section 3.4.8. The difference between the present method and the one of that section is that a positive

value β is now added to the diagonal elements of $\mathbf{R}$ rather than to those of the true Hessian $\mathbf{H}$. β is increased to whatever value necessary to make the iteration matrix positive definite *and* produce a function reduction for $\alpha = 1$. The process must succeed for large enough β, because the step tends to a very small one in the steepest descent direction. The value used for β may usefully be passed to the next iteration as a first trial value, as described in Section 3.4.8.

There will be an "optimum" value of β for each iteration in the sense that the greatest reduction in the function value will be achieved. Davies and Whitting (1971) have tried a version of Marquardt's algorithm which seeks to use this value, but conclude that the "damping" introduced is usually too large, and more iterations are required than if the first value of β to decrease the function is simply accepted.

The algorithm of Jones (1970) represents another means of interpolating between the Gauss-Newton and steepest descent directions.

Bard (1970) has conducted numerical experiments which indicate that special methods for sums-of-squares perform better than representatives of the quasi-Newton class, which ignore this special form of the objective function. Nevertheless, except when the residuals are zero at the solution or the component functions are linear, there is no theoretical reason why they should converge at a rapid rate. Theoretical work by Meyer (1970) and McKeown (1975) suggests that the ultimate convergence rate of Gauss-Newton and Marquardt-type algorithms is critically dependent on the ratio τ/μ, where μ is the smallest eigenvalue of $\mathbf{R}$ and

τ the (absolutely) largest eigenvalue of the matrix $\mathbf{S}$ whose elements are given by the *second* sum in (3.47), i.e.

$$S_{ij} = 2 \sum_{m=1}^{M} f_m \frac{\partial^2 f_m}{\partial x_i \partial x_j} \quad . \tag{3.49}$$

Specifically, if $\mathbf{x}^*$ denotes the minimizing parameter vector, it is shown that for an *undamped* Gauss-Newton algorithm

$$\lim_{k \to \infty} \frac{\| \mathbf{x}^{(k)} - \mathbf{x}^* \|}{\| \mathbf{x}^{(k-1)} - \mathbf{x}^* \|} \leqslant \tau/\mu \tag{3.50}$$

If $\tau << \mu$, convergence should be excellent, but if $\tau > \mu$ the undamped algorithm can diverge. McKeown goes on to report computational experience with functions specifically designed to lead to high τ/μ values, and shows that the specialized least-squares algorithms become increasingly inferior to alternative methods as this parameter increases.

Even so, the idea of obtaining partial Hessian information ($\mathbf{R}$) without explicitly calculating second derivatives is attractive, and several workers have proposed schemes for approximating the other ($\mathbf{S}$) part of the $\mathbf{H}$ matrix to improve convergence in the "large-residual" problems. These schemes have involved updating an approximation to the $\mathbf{S}$ matrix from iteration to iteration, in the manner of quasi-Newton methods. The work of Dennis (1973), Biggs (1977) and Betts (1976) is cited.

Davidon (1976) has suggested an interesting idea relating not to the ultimate convergence of a least-squares algorithm

but to its performance in the early iterations, far from the optimum. The point is that not all of the M component functions need to be evaluated in order to determine a useful descent direction, especially if M >> N, and much computer time is wasted in calculating information about the functions far from the optimum, most of which is discarded shortly afterwards. In his algorithm, only one component function is evaluated per iteration although all are used in a complete cycle of M iterations. The latest information is given most weight in determining the search direction to be followed. The iterates ultimately fluctuate around the optimum rather than converging to it; however, if true convergence is required there is no reason why a switch cannot be made to a "full-evaluation" algorithm.

In the remainder of this thesis, the term "Gauss-Newton method" and the abbreviation "GN" will be used to refer to any method (including Marquardt's) which is based on the $\mathbf{R}$ matrix.

3.4.13 Conjugate Gradient Algorithms

The matter of *conjugacy* of vectors was treated in Section 3.3.2 where it was mentioned that for a quadratic function the minimum will be found in just N line searches in mutually $\mathbf{H}$-conjugate directions. When first derivatives are available it is much simpler to generate conjugate directions than when they are not. To show this, we define symbols for the changes to parameter and gradient vectors which occur on the jth iteration, viz.

$$\mathbf{s}^{(j)} = \mathbf{x}^{(j+1)} - \mathbf{x}^{(j)} = \alpha^{(j)}\ \mathbf{p}^{(j)} \qquad (3.51)$$

and

$$\mathbf{y}^{(j)} = \mathbf{g}^{(j+1)} - \mathbf{g}^{(j)} \tag{3.52}$$

For a quadratic objective function, defined as in equation (3.18), i.e.

$$F(\mathbf{x}) = b + \tfrac{1}{2} (\mathbf{x} - \mathbf{r})^T \mathbf{H} (\mathbf{x} - \mathbf{r}) \tag{3.18}$$

we have by differentiation

$$\mathbf{g} = \mathbf{H}(\mathbf{x} - \mathbf{r}) \tag{3.53}$$

for any $\mathbf{x}$, so that

$$\mathbf{y}^{(j)} = \mathbf{H}\,\mathbf{s}^{(j)} = \alpha^{(j)} \mathbf{H}\,\mathbf{p}^{(j)} \tag{3.54}$$

for *any* step $\alpha^{(j)}$ along *any* vector $\mathbf{p}^{(j)}$. We now assume that we want on the kth iteration to use a search vector $\mathbf{p}^{(k)}$ which is $\mathbf{H}$ - conjugate to all of the preceding search vectors $\mathbf{p}^{(j)}$, j=1, 2 ... k-1, (where k is less than or equal to N). That is, we want

$$\mathbf{p}^{(k)\,T} \mathbf{H}\ \mathbf{p}^{(j)} = 0 \qquad j=1, 2 \ldots k-1 \tag{3.55}$$

Substituting for $\mathbf{H}\,\mathbf{p}^{(j)}$ from (3.54), the conjugacy condition becomes

$$\mathbf{p}^{(k)T} \left(\frac{1}{\alpha^{(j)}}\ \mathbf{y}^{(j)}\right) = 0 \qquad j=1, 2 \ldots k-1 \tag{3.56}$$

or since $\alpha^{(j)}$ is a scalar, simply

$$\mathbf{p}^{(k)\,T}\ \mathbf{y}^{(j)} = 0 \qquad j=1,2 \ \ldots\ k-1\ . \tag{3.57}$$

Hence, when dealing with a quadratic function a new direction $\mathbf{p}^{(k)}$ in which to search can be generated merely by ensuring that it is orthogonal to the gradient-difference ($\mathbf{y}$) vectors resulting from the previous k-1 searches.

The conjugate gradient method of Fletcher and Reeves (1964) uses that component of the current negative gradient $-\mathbf{g}^{(k)}$ which satisfies the above condition, thus combining the ideas of steepest descent and conjugacy. Formally,

$$\mathbf{p}^{(k)} = -\mathbf{Q}^{(k)}\ \mathbf{g}^{(k)} \tag{3.58}$$

where $\mathbf{Q}^{(k)}$ is the symmetric orthogonal projection matrix which annihilates $\mathbf{y}^{(1)}$, $\mathbf{y}^{(2)}$ ... $\mathbf{y}^{(k-1)}$ for $k \leqslant N$. Equivalently,

$$\mathbf{Q}^{(k)}\ \mathbf{p}^{(k)} = -\mathbf{g}^{(k)} \tag{3.59}$$

so that the conjugate gradient method satisfies the formal definition (3.6) of a "gradient method". However (3.59) is not used in practice for computing $\mathbf{p}^{(k)}$, rather (as is shown for instance by Fletcher (1972)) it may be determined simply as the sum of two vectors:

$$\mathbf{p}^{(k)} = -\mathbf{g}^{(k)} + \beta^{(k)}\ \mathbf{p}^{(k-1)} \tag{3.60}$$

where $\beta^{(k)}$ is given by either of the formulae

$$\beta^{(k)} = \mathbf{y}^{(k-1)^T} \mathbf{g}^{(k)} / \mathbf{y}^{(k-1)^T} \mathbf{p}^{(k-1)} \qquad (3.61)$$

or

$$\beta^{(k)} = \mathbf{g}^{(k)^T} \mathbf{g}^{(k)} / \mathbf{g}^{(k-1)^T} \mathbf{g}^{(k-1)} \qquad (3.62)$$

$\mathbf{p}^{(1)}$ is taken simply as the steepest descent direction - $\mathbf{g}^{(1)}$. The formulae (3.60) to (3.62) may be used repeatedly, generating more than N directions, so that the method may be used for general (non-quadratic) functions. However, it usually seems preferable to reset $\mathbf{p}$ to the steepest descent direction after N iterations. (Fletcher, 1972). This explains the remark in Section (3.4.11) that the conjugate-gradient methods use N iterations (which would minimize a quadratic function) to simulate the effect of one Newton iteration(which would do likewise). Because each $\mathbf{p}^{(k)}$ depends on the preceding $\mathbf{p}^{(k-1)}$, individual iterations are dependent on information other than that which can be derived at the present point $\mathbf{x}^{(k)}$, a feature shared with the quasi-Newton methods to be discussed in the next section, and which places them apart from the gradient methods so far discussed.

While not usually considered as effective as quasi-Newton methods, the conjugate-gradient algorithm possesses what may be a useful advantage if N is very large - there is no necessity to store any matrices.

3.4.14 Quasi-Newton Methods

In the class of methods known as "quasi-Newton" or "modification" methods the matrix $\mathbf{A}$ of (3.6) is initially set

to some arbitrary positive definite matrix (usually the unit matrix) and *updated* from iteration to iteration in such a way that it tends towards the Hessian matrix **H** as the solution is approached. Only function value and first-derivative information is used in determining the update. Positive definiteness of the **A** matrix is maintained during the procedure, so that a descent direction is produced at each step.

The plausibility of such a procedure is again demonstrated by appealing to the properties of quadratic (constant Hessian) functions. From equation (3.54)

$$\mathbf{y}^{(k)} = \mathbf{H}\,\mathbf{s}^{(k)} \tag{3.63}$$

where $\mathbf{y}^{(k)}$ and $\mathbf{s}^{(k)}$ are the gradient and parameter vector differences for some arbitrary step in parameter space (equations (3.51), (3.52).) Since **A** is supposed to be an approximation to **H** we would like to force **A** to have the property expressed by (3.63), i.e.

$$\mathbf{y}^{(k)} = \mathbf{A}^{(k)}\,\mathbf{s}^{(k)} \tag{3.64}$$

but since $\mathbf{s}^{(k)}$ and $\mathbf{y}^{(k)}$ are not known until *after* the kth step this is not possible. However, the *next* approximation can be forced to have such a property, i.e.

$$\mathbf{y}^{(k)} = \mathbf{A}^{(k+1)}\,\mathbf{s}^{(k)} \tag{3.65}$$

an equation known as the "quasi-Newton condition".

Since $\mathbf{A}^{(k)}$ is already some sort of approximation to $\mathbf{H}$, we don't want $\mathbf{A}^{(k+1)}$ to differ too drastically, and so we compute $\mathbf{A}^{(k+1)}$ from an update formula

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mathbf{C}^{(k)} \tag{3.66}$$

where $\mathbf{C}^{(k)}$ is restricted to being a matrix of low rank (usually 1 or 2) and further restricted so that $\mathbf{A}^{(k+1)}$ satisfies the Quasi-Newton condition. There is still some freedom in the way in which the update matrix $\mathbf{C}^{(k)}$ is chosen, and herein lie the differences between a whole host of quasi-Newton algorithms.

An obvious disadvantage inherent in updating the matrix $\mathbf{A}$ as discussed is that it is necessary to solve a set of linear equations (3.6) at each step in order to determine $\mathbf{p}$. In practice, therefore, the information stored and updated at each iteration is either an approximation to the *inverse* Hessian $\mathbf{H}^{-1}$ or (as long as symmetric updates only are used) to the Cholesky factor $\mathbf{L}^{(k)}$ such that $\mathbf{L}^{(k)}\ \mathbf{L}^{(k)T} = \mathbf{A}^{(k)}$. Appropriate updating formulae analogous to (3.66) may be derived in either case. If the *inverse* is recurred (as is traditional) a matrix-vector multiplication only is required to determine $\mathbf{p}$ and the updating formulae are simpler than those for the factors, but Gill and Murray (1978) believe that the alternative course (requiring two stages of back-substitution to determine $\mathbf{p}$) is preferable because the maintenance of positive-definiteness can be guaranteed in the update. It is possible for roundoff error to cause the approximation to the

inverse to become indefinite, in which case the only remedy would be to reset $\mathbf{A}$ to the identity matrix, wasting useful information.

We come now to a brief survey of particular update formulae. Of those for which $\mathbf{C}^{(k)}$ is of rank 1, Broyden (1965), Barnes (1965) and Pearson (1969) have considered non-symmetric updates. If instead $\mathbf{C}^{(k)}$ is restricted to being symmetric, there is only one possible formula, whose use has been investigated by Davidon (1968), Murtagh and Sargent (1970a) and Bard (1970).

We now introduce a new symbol $\mathbf{T}^{(k)}$ for the kth approximation to the *inverse* Hessian, that is

$$\mathbf{T}^{(k)} = \mathbf{A}^{(k)-1} \tag{3.67}$$

and the formula used in lieu of (3.6) to compute the search vector $\mathbf{p}^{(k)}$ is

$$\mathbf{p}^{(k)} = - \mathbf{T}^{(k)} \mathbf{g}^{(k)} \quad . \tag{3.68}$$

The "symmetric rank one" (SR1) update formula is, in terms of the inverse approximation,

$$\mathbf{T}^{(k+1)} = \mathbf{T}^{(k)} - \frac{\mathbf{v}^{(k)} \mathbf{v}^{(k)T}}{\mathbf{v}^{(k)T} \mathbf{y}^{(k)}} \tag{3.69}$$

where

$$\mathbf{v}^{(k)} = \mathbf{T}^{(k)} \mathbf{y}^{(k)} - \mathbf{s}^{(k)} \tag{3.70}$$

This scheme possesses the highly desirable property of finite termination for quadratic functions without exact line searches (Broyden, 1967) but unfortunately the unmodified formula is unstable since there is no guarantee that successive **A** or **T** matrices remain positive definite. Davidon (1968), Murtagh and Sargent (1970a), and Bard (1970) all propose different strategies for getting around this problem, with the result that the algorithm remains competitive.

Murtagh and Sargent show that $\mathbf{T}^{(k+1)}$ is unlikely to become indefinite if

$$\frac{\mathbf{v}^{(k)T}\,\mathbf{g}^{(k)}}{\mathbf{v}^{(k)T}\,\mathbf{y}^{(k)}} < 0 \tag{3.71}$$

and accordingly evaluate the expression on the left-hand-side of (3.71) and require it to be less than some suitable small negative quantity δ_1, say -10^{-8}. If the test fails, **T** is updated not by the ordinary formula but by

$$\mathbf{T}^{(k+1)} = \mathbf{T}^{(k)} - \frac{\mathbf{v}^{(k)}\,\mathbf{v}^{(k)T}}{\mathbf{v}^{(k)T}\,\mathbf{v}^{(k)}} \tag{3.72}$$

Murtagh and Sargent use this formula also in cases when $\mathbf{v}^{(k)T}\mathbf{y}$ is too close to zero to enable (3.69) to be used in a stable manner. They require that

$$|\mathbf{v}^{(k)T}\,\mathbf{y}^{(k)}| \geq \delta_2\,\mathbf{v}^{(k)T}\,\mathbf{v}^{(k)} \tag{3.73}$$

where δ_2 is again a suitable small quantity, say 10^{-8}. Even when such tests are incorporated it can happen that $\mathbf{p}^{(k+1)}$ fails to be a descent direction, and a practical computer

program must include such a test. The standard procedure in the case of failure is to reset $\mathbf{T}$ to the identity matrix.

Many useful updates of *rank 2* belong to the "single-parameter family" defined by

$$\mathbf{T}^{(k+1)} = \mathbf{T}^{(k)} + \frac{\mathbf{s}^{(k)}\,\mathbf{s}^{(k)T}}{\mathbf{s}^{(k)T}\mathbf{y}^{(k)}} - \frac{\mathbf{T}^{(k)}\,\mathbf{y}^{(k)}\,\mathbf{y}^{(k)T}\,\mathbf{T}^{(k)}}{\mathbf{y}^{(k)T}\mathbf{T}^{(k)}\,\mathbf{y}^{(k)}}$$

$$+ \rho^{(k)}\,\mathbf{v}^{(k)}\,\mathbf{v}^{(k)T} \qquad (3.74)$$

where now

$$\mathbf{v}^{(k)} = \mathbf{T}^{(k)}\,\mathbf{y}^{(k)} - \mathbf{s}^{(k)} \cdot \frac{\mathbf{y}^{(k)T}\,\mathbf{T}^{(k)}\,\mathbf{y}^{(k)}}{\mathbf{s}^{(k)T}\,\mathbf{y}^{(k)}} \qquad (3.75)$$

and $\rho^{(k)}$ is arbitrary. If $\rho^{(k)} \geqslant 0$ such algorithms are theoretically stable if exact line searches are carried out, since Broyden (1970) has proved that if $\mathbf{T}^{(k)}$ is positive definite then so is $\mathbf{T}^{(k+1)}$ under such conditions. The case $\rho^{(k)} = 0$ gives the most famous of all gradient methods, the DFP formula, of Davidon (1959) and Fletcher and Powell (1963). This method has achieved much success, but a later update known as the BFGS formula (after the initials of its four independent discoverers, Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970)) now seems preferable. This is because the line search can usually be carried out with fewer function evaluations (Gill and Murray, 1978). For BFGS, $\rho^{(k)}$ is given by

$$\rho^{(k)} = \frac{1}{\mathbf{y}^{(k)T}\,\mathbf{T}^{(k)}\,\mathbf{y}^{(k)}} \quad . \qquad (3.76)$$

Useful surveys of quasi-Newton methods are given by Broyden (1967), Broyden (1972) and Gill and Murray (1978). The last-named authors survey some more recent work, including the use of an "optimally-conditioned" update (the use of a value of $\rho^{(k)}$ in (3.74) which minimizes a bound on the condition number of $\mathbf{T}^{(k+1)}$) and a method of Davidon (1975) which requires no line searches. They conclude that numerical evidence is not conclusive in favour of these techniques and continue to prefer BFGS on the grounds of simplicity.

3.5 Line Search Techniques

Most of the gradient methods (and the conjugate-direction method of Section 3.3.2) require some form of search along a line in parameter space. This is expressed by equation (3.5) - a search direction $\mathbf{p}^{(k)}$ has somehow been determined and a value of $\alpha^{(k)}$ is required.

All methods require $\alpha^{(k)}$ to be found in such a way that $F(\mathbf{x}^{(k+1)}) \leqslant F(\mathbf{x}^{(k)})$ which ensures that for "reasonable" functions the sequence of iterates will converge to a local minimum. This will be referred to as the "stability requirement". As was shown in Section 3.4.2, some sufficiently small positive value of α will always suffice if the matrix $\mathbf{A}$ used in (3.6) is positive definite. Thus the simplest form of line search technique would consist simply of trying a likely value for α and repeatedly decreasing this by a constant factor until a function reduction was achieved. In the case of Newton methods (when $\mathbf{H}$ is positive definite) and quasi-Newton methods, the theoretical best value (derived by considering a quadratic function) is $\alpha = 1.0$, and this is often the value first tried.

In theory many methods (e.g. the DFP method) require "exact" line searches (that is, minimization of the function along the line), to achieve their performance, and even in other cases it seems intuitively desirable that the function value be reduced as much as possible. Accordingly, practical algorithms often include line minimization subroutines of varying sophistication. The evaluation of these procedures must rely largely on experiment because even for methods theoretically requiring exact searches much success has been had with much cruder (and therefore less time-consuming) searches. This matter is taken up for representative digital filter design applications in Chapter Five.

Although a large number of approaches to the problem of localizing the minimum of a function of one variable (α) have been tried in this context, there are really only two basic methods. The first is capable of refining known bounds on the minimum by comparing function values only. The disposition of trial function evaluations to achieve this with least labour give rise to the "Fibonacci" and "golden-ratio " techniques (Gottfried and Weisman, 1973). In the second method, a simple function with a minimum (such as a quadratic or a cubic) is fitted through known α - F points. Such a fit may also utilize evaluations of the gradient made along the line. This method is usually more effective, and can even be used in an extrapolation sense (i.e. before a "bracket" on the minimum is known). Safeguards must be incorporated to ensure that extrapolations too far outside a reasonable neighbourhood of known points are not made. Such an algorithm is described by Gill and Murray (1978).

Some specific line search techniques are described and computational experience is reported, in Chapter Five.

3.6 Published Comparisons of Unconstrained Minimization Techniques

Extensive comparisons of the performance of minimization techniques are rare in the literature. Most of the original papers proposing a new technique compare it with at least one other method, but the test functions used are rather restricted and usually of low dimensionality. There is the added complication that (especially if results from another's work are quoted) the test conditions may not be uniform, particularly with regard to convergence criteria. Such tests invariably tend to show the "new" method in a good light.

To make matters worse, the criteria for comparing computational effort are by no means easy to define. Comparisons of actual CP time are useless when different computers are used, and execution time in any case will depend on the "cleverness" of the programmer. "Complicated" subroutines written for investigative purposes in particular may suffer from poor coding. Some authors quote "number of function evaluations" whilst others use "equivalent function evaluations" where a gradient evaluation is often assumed to be equivalent to N evaluations of the function. In many problems this will not be even approximately true. Such measures of performance ignore all computational overheads peculiar to the methods themselves, for example, a Greenstadt method requiring an eigensystem analysis at each iteration may be quite inefficient although it appears good with regard to function evaluations.

Some methods (particularly Newton methods) may benefit from a fortuitous choice of starting point, and most comparative surveys have not attempted to average the performance of the algorithms over a variety of starting points.

Hopefully, some of the above objections are answered for a restricted class of objective functions, the digital filter problems, by the analysis of Chapter Five. The remainder of this section is devoted to a brief look at some comparative surveys appearing in the mathematical literature.

Box (1966) looked at eight methods as applied to five "sum-of-squares" problems of 2, 3, 5, 10 and 20 dimensions. Two of the methods are specialized sums-of-squares methods (Powell (1965), requiring no gradient evaluations, and Barnes (1965)). Of the remainder, four do not require derivatives (Powell's 1964) conjugate direction method and three direct-search methods), one is a conjugate gradient method (Fletcher and Reeves, 1964) and one a quasi-Newton method (Fletcher and Powell, 1963). The criterion used for comparison was "equivalent function evaluations" with a gradient determination counting as N function evaluations. Several starting points were used in each case.

Box concluded that the sums-of-squares methods are preferable for this type of problem, but in all cases the residuals at the solution were zero so that the considerations of McKeown (1965), as discussed in Section 3.4.12, do not apply. Of the "general" methods the Fletcher-Powell method was superior but Powell's derivative free (1964) method very competitive. The conjugate-gradient technique was somewhat poorer but better than most direct-search techniques. The Nelder-Mead (1965) algorithm was the best of the direct-search methods tested, being somewhat better than conjugate gradients for problems of low dimensionality, but its performance in 20-dimensions was quite poor.

Bard (1970) again treated sums-of-squares (parameter estimation) problems, employing methods which use analytic first derivatives only. The test problems were more searching than those of Box, some algorithms failing to produce acceptable answeres in some cases. Dimensionality N was 3 to 10. Only one starting point was used for each problem.

Algorithms tested were the damped Gauss-Newton and true Marquardt (which are specifically for sums-of-squares), the Davidon-Fletcher-Powell (DFP) method, and the symmetric rank-one (SR1) quasi-Newton method with a) the Hessian approximation carried and b) the inverse Hessian approximation carried. Several schemes for line search, finding the minimum to various degrees of accuracy, were carefully defined and tested.

Algorithms were ranked firstly in terms of robustness (smallest number of failures) and secondarily in terms of number of "equivalent function evaluations". The conclusion was that the sums-of-squares methods are significantly superior for this type of problem. In the case of the damped Gauss-Newton method, excessive effort to locate the minimum was unjustified but it was beneficial to try an additional value of α rather than just accept the first which met the stability criterion.

The SR1 algorithm was significantly better than DFP, and Bard attributed this to the former's not requiring accurate line searches to achieve quadratic termination. For both algorithms a "moderate" effort in determining a step length was worthwhile (more effort for DFP).

Bard's versions of the SR1 algorithm are rather unusual. In one case he replaces the eigenvalues of the approximate Hessian

with their absolute values to ensure a descent direction following Greenstadt (1967), which could be a very time-consuming procedure. In another version he simply reverses the search direction if necessary, making no effort to retain positive-definiteness of the matrix. However, Murtagh and Sargent (1970b) also compared various (more orthodox) versions of SR1 with DFP and again found it usually superior.

Himmelblau (1972) attempted a "uniform" evaluation of a number of methods on problems of up to five dimensions only. The uniformity is in the sense that convergence criteria (in terms of relative changes to parameters, function value and gradient norm) are the same in each case, and comparison is in terms of computer time (on the same machine) so that matrix algebra and other overheads are included as well as function evaluation time. Himmelblau's preference is for the algorithm of Fletcher (1970) which switches between the DFP and BFGS updates depending on the result of a simple test, and which does not require line searches, but he did not test BFGS alone. The DFP and Broyden (1965) non-symmetric rank one algorithms were ranked "very good" but the accurate line searches used made then considerably slower than Fletcher's method. Powell's (1964) algorithm was the best of those not employing derivatives, ranking considerably higher than Stewart's (1967) and direct-search techniques.

Sargent and Sebastian (1972) studied the performance of some gradient algorithms for "classic" test problems of up to four variables. They found that the performance of the DFP algorithm uniformly improves as the accuracy requirement on the linear search is *relaxed*, at least down to $\varepsilon = 0.1$. However,

replacement by a mere stability test led to *failure* on three out of the five problems. Both BFGS and SR1 performed well with a stability test, and better in this mode than when line minimization was attempted. However, both outperformed DFP even in minimization mode. On their evidence, the BFGS update is marginally better than SR1.

The Fletcher-Reeves (1964) algorithm was also tested by Sargent and Sebastian, who found it inferior to DFP, supposedly due to greater accuracy requirements in the line search ($\varepsilon = 0.01$ seemed optimum) and to loss of information in the periodic re-starts.

The general conclusions to be drawn from the literature seem to be that:

a. Specialised sums-of-squares methods should be used if applicable (but McKeown (1965) shows that functions *can* be constructed for which this conclusion does not hold.)

b. Gradient-based methods are generally superior to direct-search methods. The Powell (1964) and Stewart (1967) algorithms which simulate gradient methods are the best of those not requiring analytic derivatives, but they are not as good as those that do use them.

c. Of methods using first derivatives, quasi-Newton algorithms are superior to conjugate - gradients, and those for which a stability test can safely replace line minimization are the best of the quasi-Newton class.

However, tests have been on restricted classes of problems, mainly of low dimensionality, and the above conclusion may not be general. As for second-derivative methods, which appear to offer significant theoretical benefits, the present author knows of no comprehensive numerical comparisons either with each other or relative to the other classes of algorithm.

CHAPTER FOUR

4. APPLICATION OF OPTIMIZATION METHODS TO DIGITAL FILTER DESIGN

4.1 Introduction

The aim of this chapter is twofold. Firstly there is a survey of the digital signal processing literature to identify areas in which mathematical optimization methods have been used. An attempt is made to achieve some philosophical unification of this field by grouping published works according to general features of their mathematical formulation.

Much of the work reviewed is not dealt with in detail beyond that necessary to place it in the above context. However, other material provides the framework within which various mathematical optimization methods are to be compared (in chapter five). And so the second objective of this chapter is the development, in some detail, of the mathematics involved in these applications. Some originality is claimed for this development. Although the mathematics involved is no more than analytic differentiation and elementary algebra, it is believed that the *generality* of the derivation is new. In contrast, most published work has fixed attention on a *particular* type of filter design problem before embarking upon the mathematical development.

Although some generality is claimed for the mathematical development, it is also necessary to attempt to justify the *restrictiveness* of the types of problem chosen for detailed development here and for computer study in chapter five. In

particular, attention is given mainly to sum-of-squares (SS) objective functions and to digital filters which are realized as a cascade of second-order sections. Of course, some restriction of scope has been necessary because of the limited time available for the study and because this thesis would otherwise be too unwieldy. And both SS functions and cascade filters have been much treated in the literature which may in itself be enough to justify further study. However, I believe that there are some more particular justifications for the continued use of cascade filters and SS objective functions. More details are given later, in particular in sections 4.3.3 and 4.4.2.

An appreciation of the remainder of this thesis requires some familiarity with the fundamentals of digital filtering. Such material is presented in several textbooks, for example, in Bogner and Constantinides (1975), and it is hardly useful to repeat it here. However, section 4.3 deals with elementary material, for the sake of introducing notation and terminology, to justify the emphasis of following sections, and, hopefully, to add to the clarity of the exposition.

The matter of *discrete* optimization of digital filter coefficients (determination of the best, or at least of "good", coefficients of a given wordlength) is of interest in this thesis and a sizeable amount of literature has addressed this problem. However, a survey of this area is deferred to chapter six - in the present chapter and the next we are concerned with determining "ideal" values of coefficients as though they were not subject to quantization.

4.2 Area of applicability of optimization Methods

As mentioned in chapter 2, the technique of mathematical optimization is extremely general and can in principle be applied to the parameter - determination phase of almost any design or system modelling problem. The distinction between "design" and "modelling" is unimportant - the mathematical formulation is similar regardless of whether we want a machine to *do* something (perhaps at minimum cost) or just require a mathematical model (however artificial) which provides a concise *description* of an observed process.

As used in this thesis (and in most hitherto published works), optimization is useful only in determining the most appropriate *values of the parameters* of a design or of a model, once the *form* of that model has been decided upon. The matter of arriving at such a *form*, which will achieve the machine design objectives on the one hand, or enable a "sufficiently accurate" correspondence between the model and the measured process on the other, is still largely an art. Very little progress has apparently been made in automating this phase of the process. However, in the matter of electrical network design, Director and Rohrer (1969a) have extended the concept of parameter optimization in an interesting way. The partial derivatives of the performance criterion (objective function) are calculated with respect to certain *non-existent* circuit elements. If such figures seem to indicate that significant benefits would accrue, the network is allowed to "grow" a new element.

It would seem that a similar approach could be taken in digital filter design. However, the main business of this thesis is the comparison of optimization techniques themselves, and such comparison is readily done within the framework of the more prosaic "conventional" uses of parameter optimization. It is perhaps more correct to speak of "system identification" rather than "design" or "modelling".

The principal requirement for the use of an optimization technique is that we can compute, for any given values of the variable parameters, the value of some objective function that is a suitable measure of the desirability of the design or the closeness of fit of the model.

Many common signal processing tasks fit naturally into this mould; for example inverse filtering (or linear prediction) requires the minimization of the energy in an error signal. However, the general mathematical optimization techniques are iterative in nature (and each iteration usually requires much computation), and so their use has not been much advocated for processing signals such as speech in real time. This situation may change, however, as small digital processors with more speed and/or more parallelism become available. The great generality of the optimization techniques could allow the use of dynamic models whose parameters could not be estimated any other way.

To date, most use of optimization for signal processing has been in the design of digital filters. If a filter is not required to adapt to changes in the characteristics of the signal with time, it may be designed once-and-

for-all, with the help of a general-purpose digital computer. In this case the computation time is not of central importance, and not surprisingly most researchers have simply made use of the standard pre-programmed versions of optimization sub-routines available at most scientific computer sites. But again, if the filter is required to be adaptive, the time efficiency of the method used to evaluate its parameters becomes important. This provides the main motivation for the present comparison of optimization methods in the signal processing context.

4.3. Digital Filters

4.3.1 General

The type of "digital filter" to be considered is the usual linear discrete-time system in which an output sequence of numbers v_i, $i = 0,1,2,\ldots$ is produced element-by-element, by linearly combining past output values and past and present values of an input sequence u_i, $i = 0,1,2\ldots$. In symbols

$$v_i = \alpha_o u_i + \alpha_1 u_{i-1} + \ldots + \alpha_m u_{i-m} - (\beta_1 v_{i-1} + \beta_2 v_{i-2} + \ldots + \beta_n v_{i-n}) \qquad (4.1)$$

and the α's and β's are real numbers. The *z-transform* of the causal real-valued sequence u_i is defined as a function of a complex variable z, viz.

$$U(z) = u_o + u_1 z^{-1} + u_2 z^{-2} + \ldots . \qquad (4.2)$$

With a corresponding definition for the z-transform of the

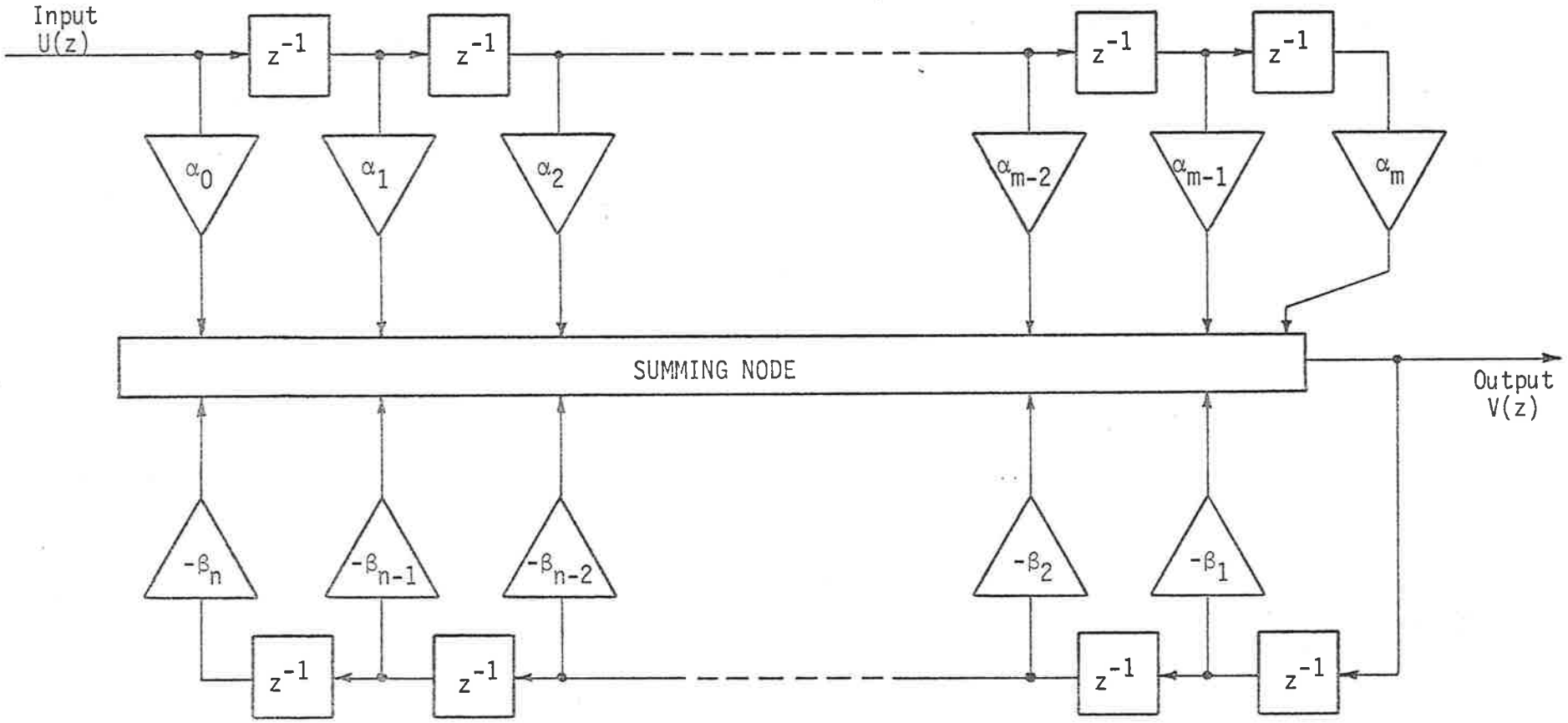


FIGURE 4.1 Direct Realization of Rational Digital Transfer Function

output sequence V(z) it follows from (4.1) that

$$\frac{V(z)}{U(z)} = \frac{\alpha_o + \alpha_1 z^{-1} + \ldots + \alpha_m z^{-m}}{1 + \beta_1 z^{-1} + \ldots + \beta_n z^{-n}} = H(z) \qquad (4.3)$$

The digital filter is completely characterized by this expression for its *transfer function* H(z), and it may be *realized* by the configuration of figure 4.1, which directly implements equation (4.1). As in equation (4.2) the *operator* z^{-1} has the meaning of "unit delay".

4.3.2 Finite Impulse Response Filters

There is an important special form of linear digital filter in which there is no feedback of past outputs. The difference equation (4.1) reduces to the convolution sum

$$v_i = \alpha_o u_i + \alpha_1 u_{i-1} + \ldots + \alpha_m u_{i-m} \qquad (4.4)$$

and an obvious implementation is the *transversal* filter of figure 4.2. The impulse response is of finite duration (m + 1) and is given simply by the coefficients themselves.

Simple forms of such finite-impulse-response (FIR) filters have been in use since long before the advent of digital computers, an example being the "moving-average" smoothing of time series data, where m could be, say, 4 and $\alpha_o = \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1/5$. In the frequency domain this is a crude low-pass filter. Since the impulse response of such a digital filter is related to a set of $\frac{(m+1)}{2}$ complex-valued samples in the *frequency domain* via the Discrete

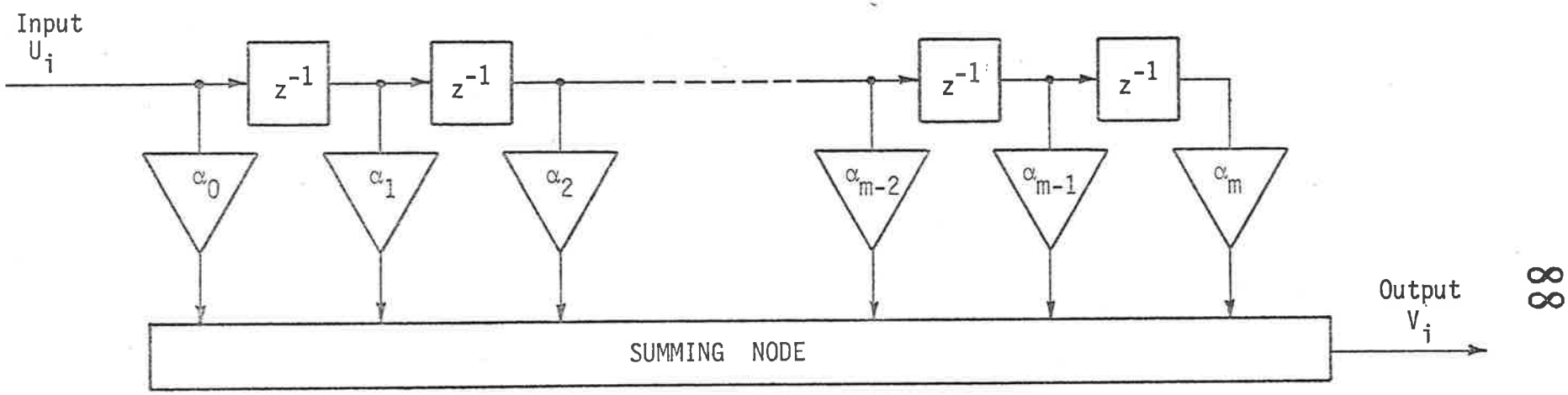


FIGURE 4.2 Transversal Realization of FIR Digital Filter

Fourier Transform (DFT), it follows that arbitrarily complicated frequency domain objectives may be met by such a filter provided that m is made large enough. (In fact if m is large the convolution (4.4) is usually *implemented* by using the DFT; the amount of computation is thereby reduced but the filtering action is exactly equivalent to (4.4)).

The simple specification of a set of frequency samples and use of the inverse DFT to get α_o, α_1 ... α_m is not the whole story in the design of FIR filters, because the frequency response may not be acceptable *between* the samples. There are several methods for improving the design, some involving optimization techniques, and these are mentioned in section 4.5. The straightforward use of general nonlinear optimization methods (in which every filter coefficient is a variable parameter) is inappropriate for FIR filters because m is usually too large (typically several hundred) and because alternative methods are more efficient. Accordingly, the remainder of section 4.3 relates to useful filters for which the number of coefficients is smaller and alternative design procedures less satisfactory, that is the recursive or infinite impulse response (IIR) filters.

4.3.3 Infinite Impulse Response Filters

IIR digital filters are often preferred over FIR filters, principally because they are more efficient for many applications (in terms of number of arithmetic operations per output sample). They may be especially efficient as models of real physical phenomena, such as speech, which are usually considered as the forced response of some system possessing its

own resonances (poles). Although an FIR filter can approximate any impulse response the required length may be excessive, and an IIR filter model (possessing poles) is likely to correspond much more closely with the real physical system.

To offset this fundamental advantage are a number of disadvantages, for instance:

a. IIR filters can be unstable, and any design procedure must include stability checks.

b. Coefficient sensitivity (quantization) problems are much more serious than with FIR filters.

c. Low-level limit cycles and overflow oscillations are possible.

d. Noise introduced by finite wordlength arithmetic is more significant.

e. There is no simple method for ensuring a design with linear phase (if an IIR filter had all its poles *on* the unit circle the phase *would* be linear, but such a filter would not normally fulfil a useful function).

f. There are no special elegant techniques for producing designs to arbitrary specifications. Optimization methods are necessary, and furthermore the design problem is inherently nonlinear, so that linear programming is not applicable in a straightforward way (but it *has* been applied - section 4.6.2).

None of these difficulties is insurmountable and IIR filters are often used. Design may be tackled in the frequency domain (section 4.6) or the time domain (section 4.7).

4.3.4 Realizations of IIR Digital Filters

Equation (4.3) may be manipulated in several ways to yield algebraic forms suggesting realizations of IIR filters apart from the direct form of figure 4.1. Thus, by factorizing the numerator and denominator polynomials in (4.3) into quadratic factors and arbitrarily pairing them off, we have

$$H(z) = A_o \prod_{k=1}^{K} \frac{1 + a_k z^{-1} + b_k z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (4.5)$$

and the corresponding *cascade realization* of figure 4.3. (If $n \neq m$ or if n or m is odd, the expression (4.5) is still valid but some of the coefficients are zero).

By expanding (4.3) as a sum of partial fractions, we have

$$H(z) = \gamma_o + \sum_{k=1}^{K} \frac{\gamma_k + \delta_k z^{-1}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (4.6)$$

and the corresponding *parallel realization* of figure 4.4.

There are other possible realizations based on expansion of (4.3) in continued fractions. If we wished to use complex arithmetic we could also find cascade and parallel forms based on the linear (first degree) factors of the denominator of (4.3).

If the coefficients involved in these filters are specified to very high accuracy, then for the same input sequence all realizations will produce the same output sequence. However in practice the coefficients must be represented as binary numbers (fixed or floating-point) of a certain word-

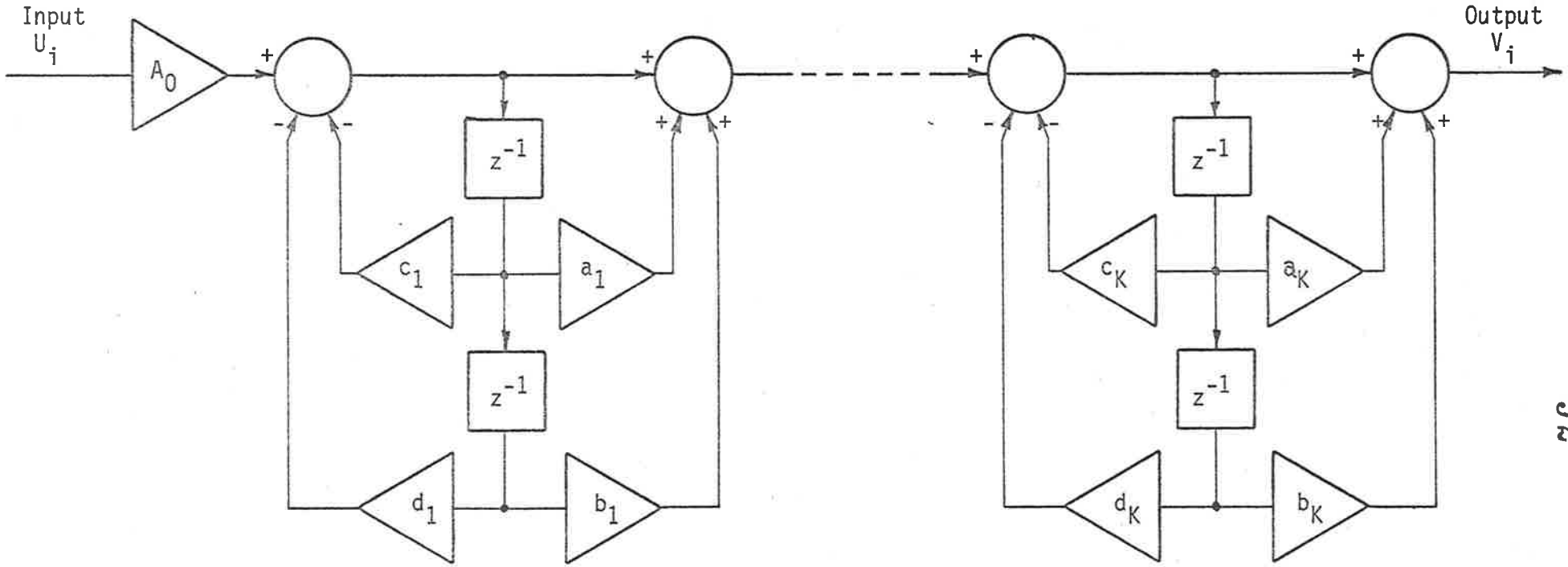


FIGURE 4.3 Cascade Realization of Infinite-Impulse-Response Digital Filter

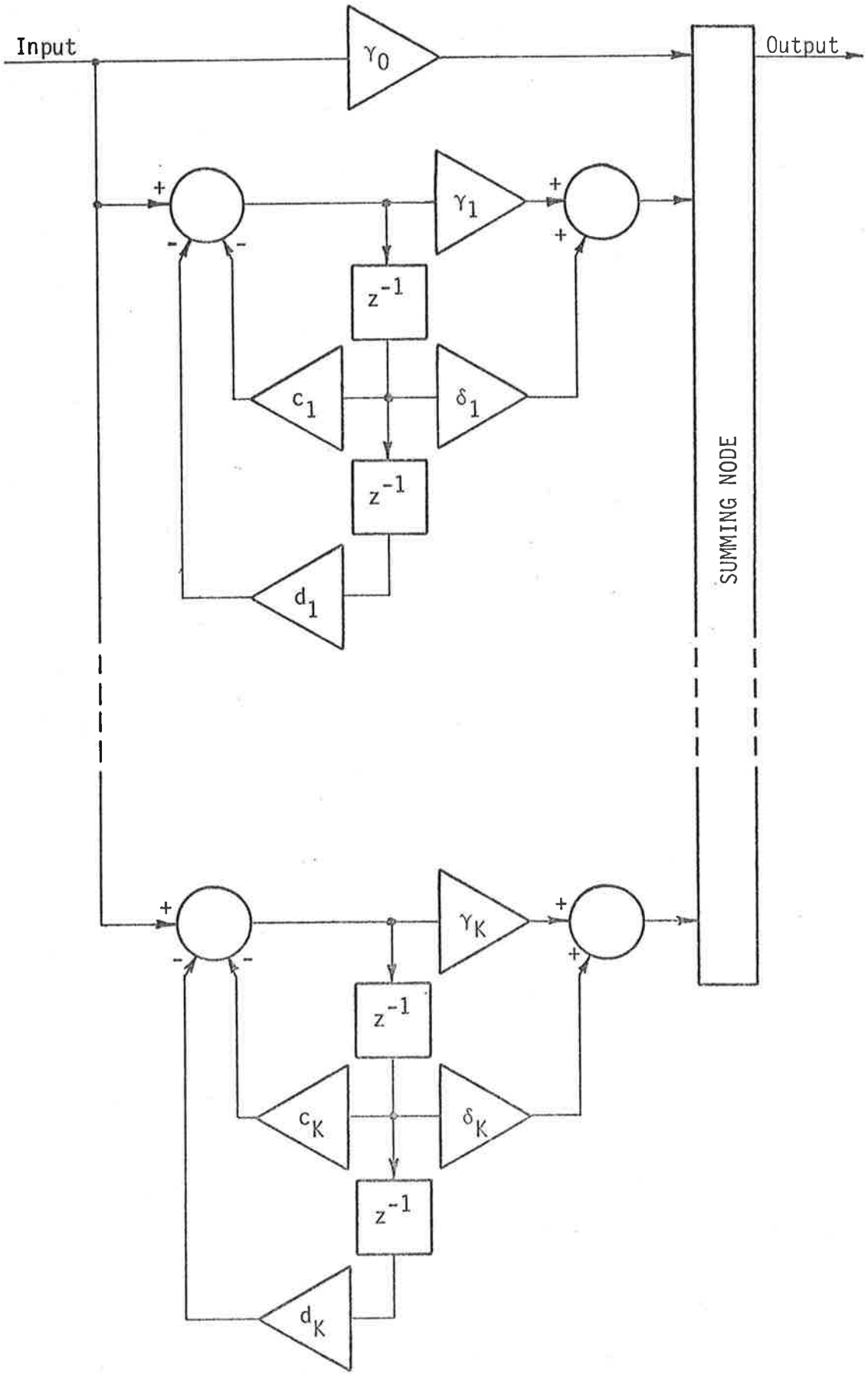


FIGURE 4.4 Parallel Realization of Infinite Impulse Response Digital Filter

length and the "ideal" coefficients cannot be used. Chapter 6 of this thesis addresses the problem of choosing an optimal combination of fixed-wordlength coefficients. In the present context I wish merely to mention that quantization of the coefficients has a greater (deleterious) effect with some realizations than with others.

What is important is the sensitivity of the positions of the *poles* and *zeros* (the zeros of the denominator and numerator of (4.3), respectively) to small changes in the coefficients. In this respect the direct form (figure 4.1, or the *canonic* equivalent shown in figure 4.5, involving the smallest possible number of delays) in which the poles and zeros are determined as zeros of high-order polynomials, can be especially bad. (National Physical Laboratory, 1961, pp 59-60).

In the cascade realization individual poles and zeros depend only on two coefficients, and are only moderately sensitive functions of them. This form is often used for general-purpose digital filtering. In addition, the positions of both poles and zeros are easy to calculate when the coefficients are known, and this can be important during the application of an optimization method of design. Firstly it is easy to check that poles do not wander outside the unit circle $|z| = 1$, resulting in unstable (and so, useless) filters, and secondly, and more subtly, there is the matter of root pairing, to be explored in section 4.9.2.

The parallel form has similar pole coefficient sensitivity properties to the cascade form, and is sometimes used, although probably less often. Although the poles are

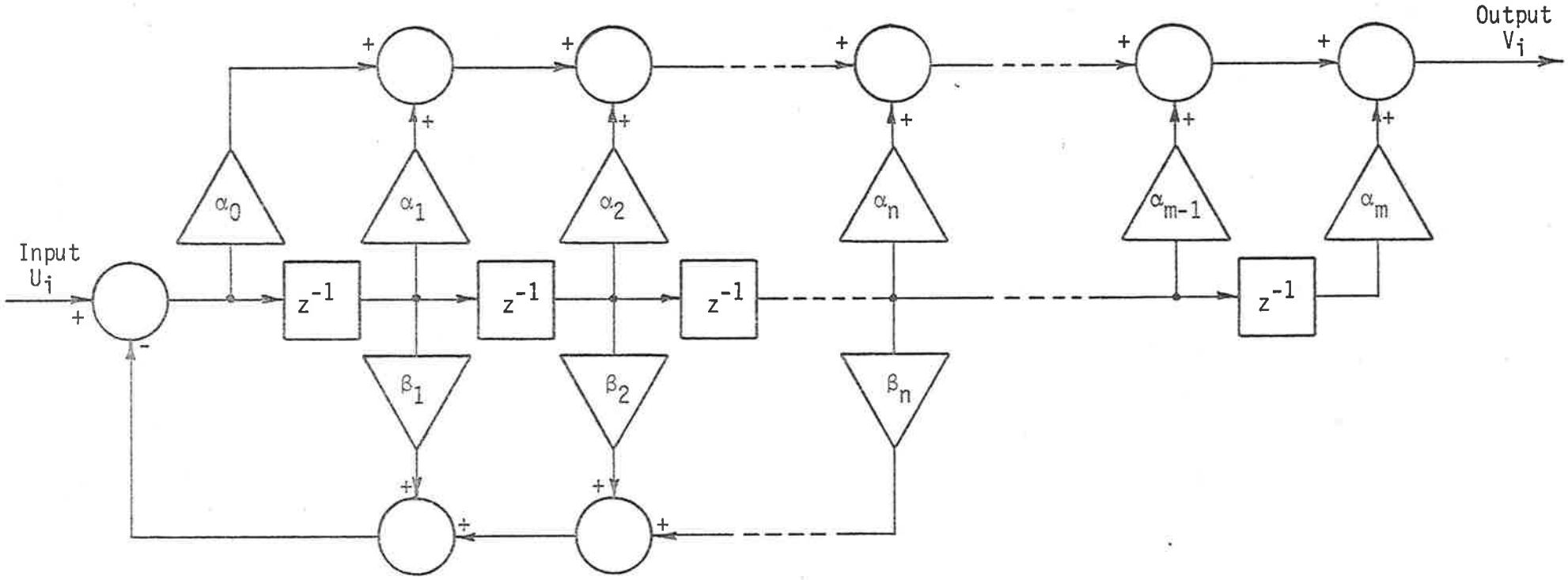


FIGURE 4.5 Direct Canonic Realization of Infinite Impulse Response Digital Filter

easy to calculate, the positions of the zeros are not obvious and direct design by optimization is not as convenient as with the cascade form. Of course, a parallel-programmed filter could be derived from one in cascade form as a second phase of the design procedure, if such a filter were needed.

With any of the foregoing realizations, the sensitivity of pole (or zero) positions to coefficient changes is not independent of the positions of the poles themselves. In other words, for any given coefficient wordlength some regions of the z-plane are richer in possible pole positions than others. Consider a single second-order section, the basic building block for the cascade-form filter of figure 4.3, and a coefficient quantization coarseness $q = 2^{-3}$. All possible pole locations are shown in figure 4.6, from which it is obvious that if we wanted a pole in the vicinity of $z = \pm 1$ we would have to settle for a very poor approximation. These are precisely the regions of interest for narrow-band low-pass and high-pass filters, and several configurations have been suggested which have more appropriate pole distributions. Examples are the coupled form of Rader and Gold (1967) shown with its possible poles ($q = 2^{-3}$) in figure 4.7, and a circuit proposed by Avenhaus (1972) shown in figure 4.8. These (and other) forms are important and would be used in critical applications. However, they are not relevant in the early stages of the design when the problems of wordlength are not being considered. They are used in a cascade configuration and would be designed by a *discrete* optimization procedure (chapter 6) after the ideal coefficients for an "ordinary" cascade realization had been found.

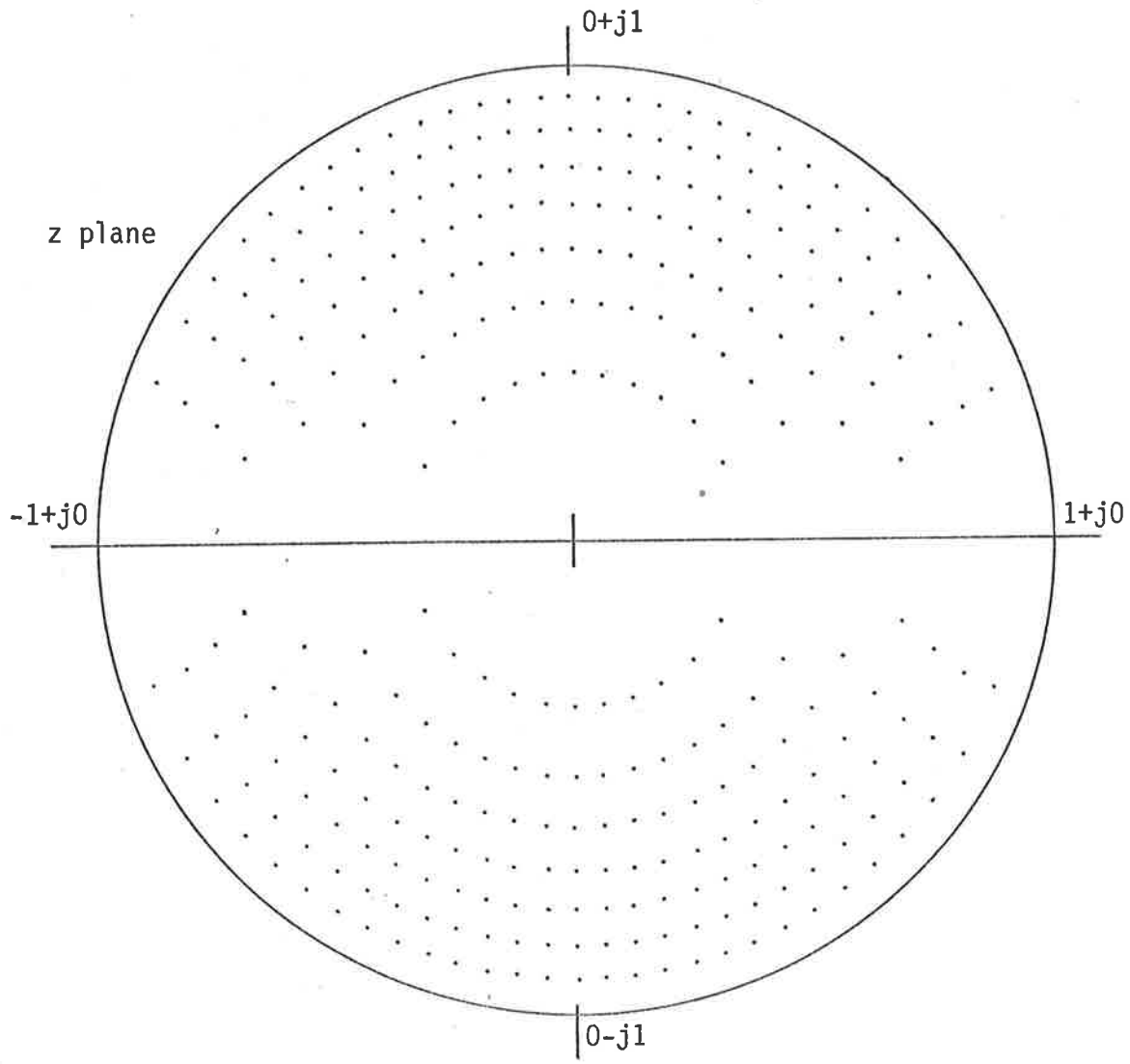
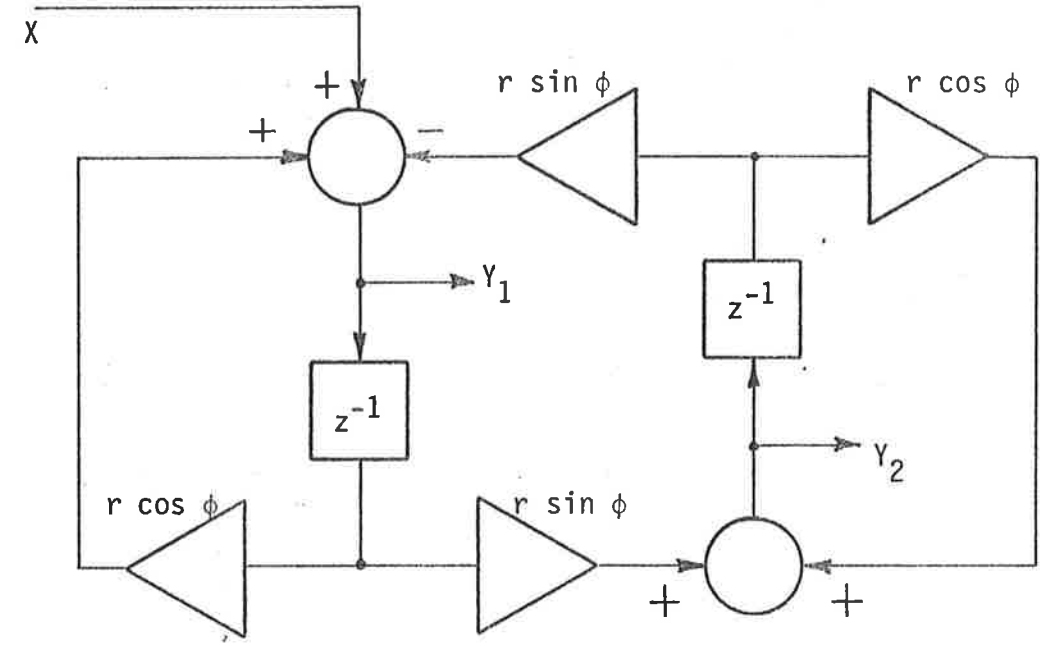
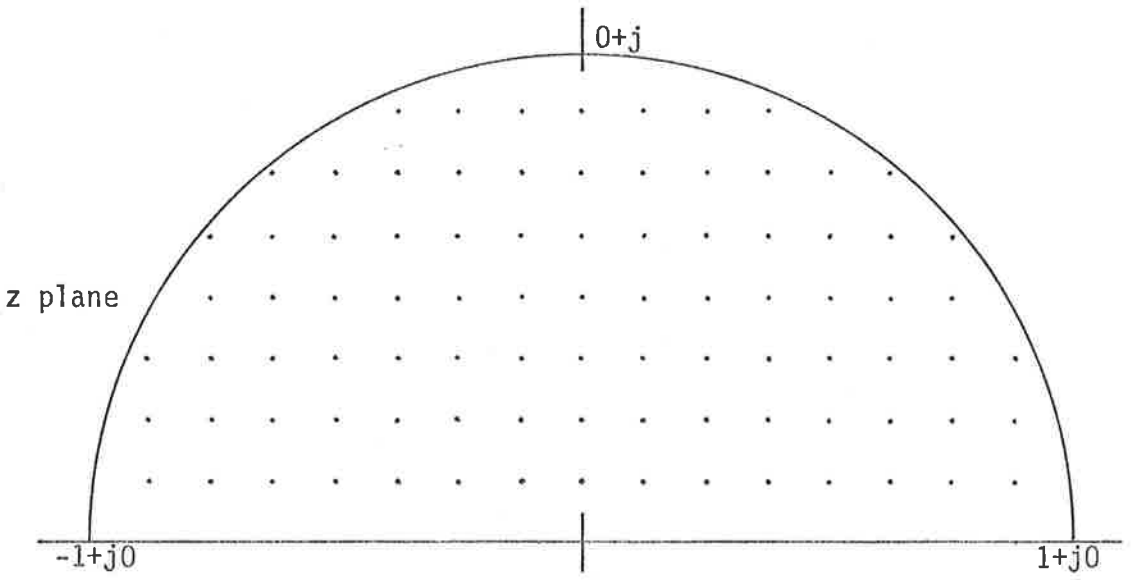


FIGURE 4.6 Direct Realization of 2nd Order Section

Pole Positions, $q = 2^{-3}$



(a)



(b)

FIGURE 4.7 (a) "Coupled-form" Second-Order Resonator

(b) Pole Positions, $q = 2^{-3}$ (upper half plane only)

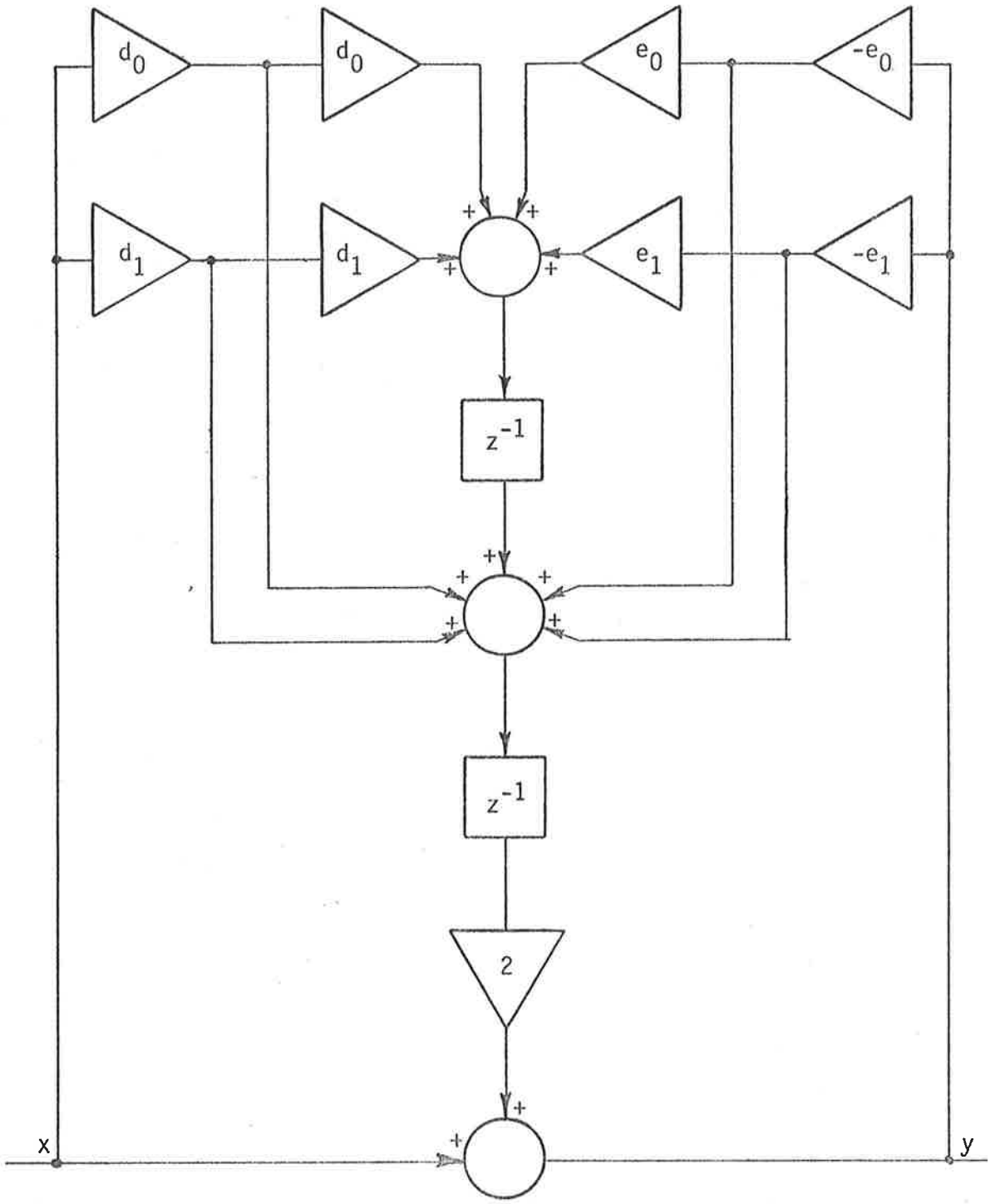


FIGURE 4.8 Avenhaus (1972a) "Circuit C" Second-Order Filter Section

At this stage it is fitting to point out that there are many algebraic schemes, not involving general optimization techniques, for generating digital filters as models of dynamic processes. Examples are the generation of an all-pole model for speech by linear prediction (Makhoul, 1975), Shanks' (1967) system identification method, and the iterative algorithms of Steiglitz and McBride (1965) and Evans and Fischl (1973). Although there are exceptions (notably those dealing with "lattice" filter structures), the majority of these techniques produce the coefficients of a digital filter in *direct* form; consequently if the pole and zero positions are the desired output all computations must be done to high accuracy and, furthermore, an accurate (and iterative and therefore somewhat time-consuming) process will be necessary to find the second-order factors of the high-order polynomials. Both of these requirements are unfortunate, particularly if we want to perform the computations in real time on a short wordlength minicomputer.

Thus it may be worth sacrificing some of the elegance of the algebraic techniques and using a "cruder" optimization method to yield a cascade-form model directly, from which pole and zero positions are then easily found.

In summary, the realization of a recursive digital filter as a cascade of second order sections as in equation 4.5 and figure 4.3, is a highly desirable form to consider in conjunction with optimization studies. This is both because it is often used in its own right, and because the locations of both poles and zeros are readily determined from it. Optimization techniques for design may need to include frequent computation of poles and zeros. Special filter con-

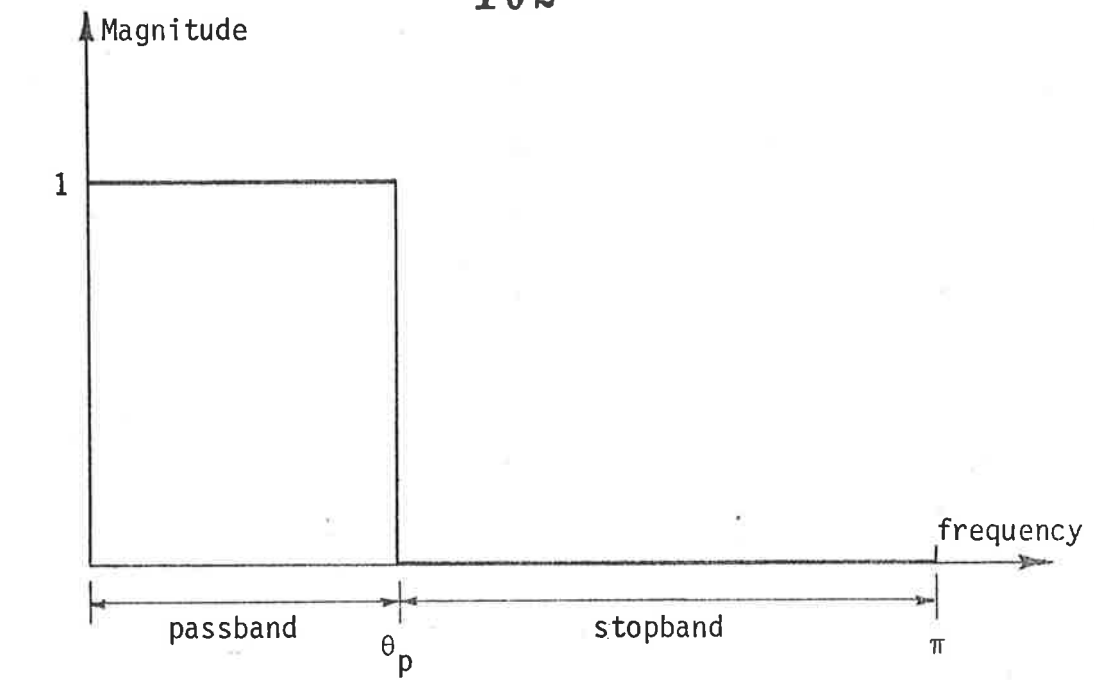
figurations which are good from the point of view of coefficient quantization are readily derived from the cascade form. For these reasons it is this form which is considered in most of this thesis.

4.4 General Formulations of the Approximation Problem

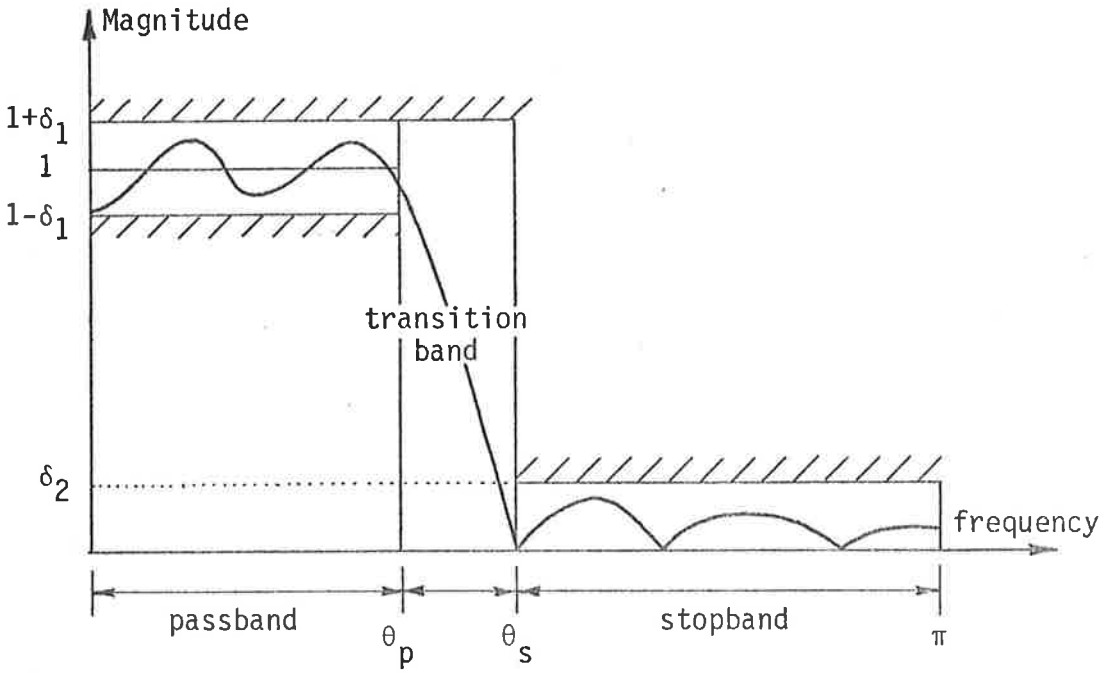
4.4.1 Formulation A - Many Constraints

We consider now how the problem of designing a digital filter to meet a specification can be put into a form suitable for the application of optimization techniques. To be specific we first consider that the specification is on the magnitude of the frequency response, but some of the theory is very general and applies to almost any problem of filter design by optimization.

A type of filter often required is the low-pass filter. The ideal "brickwall" characteristic of figure 4.9 (a) represents what we would like such a filter to do, that is pass unchanged any sinusoidal signal with a frequency less than θ_p and block completely such a signal with a greater frequency. Whether the filter causes phase changes to signals within its passband may or may not be relevant. In most cases it would be, and a particularly sought-after characteristic is that of linear phase - phase lag rising linearly with frequency. Such a filter imposes a constant delay on all signals and so does not distort the wave-shape of any signal within its passband. In the present example phase is ignored. (It may be possible to ensure a desirable phase characteristic by techniques independent of the "magnitude" design). Here as in all parts of this thesis the frequency axis is considered to run from 0 to π and the frequency variable has the symbol θ, and the direct interpretation as the polar angle in



(a)



(b)

FIGURE 4.9 (a) Ideal Lowpass Filter Characteristic

(b) Practical Lowpass Filter Specification

in the z plane. $\theta = 2\pi$ corresponds to the sampling frequency F_s (Hertz).

No real filter can achieve the ideal magnitude characteristic, and more realistic specifications allow variation between certain limits, say $1 - \delta_1$ to $1 + \delta_1$ in the passband and 0 to δ_2 in the stopband. In addition, a transition band is allowed; the stop band edge is above the passband edge. Within the transition band the formal requirement would usually be 0 to $1 + \delta_1$, but in fact most filters can readily ensure a monotonic characteristic within this region. These features are shown in figure 4.9 (b). The values of δ_1 and δ_2 may be equivalently specified in decibels as allowable passband ripple and minimum stop band attenuation.

The maximum number of extrema possible in the magnitude characteristic is dependent on the complexity of the digital filter, and is of the same order as the number of coefficients. Thus by evaluating the response at a sufficiently dense set of frequencies, θ_m, $m = 1,2 \ldots M$ it can be ascertained whether or not a given filter meets the specification without fear that there could be undetected sharp error peaks between the frequency samples chosen.

Let us suppose that a set of M frequencies $\{\theta_m, m = 1,2...M\}$ is chosen such that $\theta_m < \theta_p$ for $m < \ell_1$, $\theta_p < \theta_m < \theta_s$ for $\ell_1 \leq m \leq \ell_2$, and $\theta_m > \theta_s$ for $\ell_2 < m \leq M$. The θ_m are not necessarily evenly spaced but are distributed such that coverage of the frequency scale is "reasonably" uniform. Suppose also that the N "*designable parameters*" of the filter are placed in arbitrary order to form a *parameter vector* $\mathbf{x}$. The form of the filter has not as yet been specified, and the elements of $\mathbf{x}$ could be, among others

(a) the α and β coefficients of a direct-form IIR filter as shown in figure 4.1 ($N = m + n + 1$)

(b) the a, b, c and d coefficients (and possibly A_0) of a cascade form filter, as in figure 4.3 or equation (4.5) ($N = 4k$ or $4k + 1$)

(c) the transversal filter tap gains $\alpha_0 \ldots \alpha_m$ shown in figure 4.2, that is, the values of the impulse response samples of an FIR filter ($N = m + 1$)

(d) some or all of the frequency response samples of an FIR filter, that is, values of the DFT of the impulse response sequence ($N < m + 1$)(If only *some* of the frequency samples are included it is because the others are considered fixed, a priori - see section 4.5).

Whatever the form of the filter, there will be a formula (of greater or less complexity) for the magnitude response at any particular frequency as a function of the elements of $\mathbf{X}$. We use the symbols $B_m(\mathbf{X})$ for the response at $\theta = \theta_m$ and $B(\mathbf{X}, \theta)$ for the response for *any* θ. The requirement "meet the specification" may be expressed by the set of mathematical constraints

$$\begin{aligned} 1 - \delta_1 < B_m(\mathbf{X}) &< 1 + \delta_1 \qquad m < \ell_1 \\ B_m(\mathbf{X}) &< 1 + \delta_1 \qquad \ell_1 \leqslant m \leqslant \ell_2 \\ B_m(\mathbf{X}) &< \delta_2 \qquad m > \ell_2 \end{aligned} \tag{4.7}$$

(There is the very minor point that the specification could still be violated at some frequency between those included in the set $\{\theta_m, m = 1, 2 \ldots M\}$. But since the variation of $B(\mathbf{X}, \theta)$ is "smooth" this will not happen if we set δ_1 and δ_2 slightly below the actual

specified values and use a well-distributed set $\{\theta_m, m = 1, 2...M\}$, and M >> N. This matter will not be considered further.)

The formulation of a set of constraints goes part of the way towards casting the design problem in a form suitable for an optimization technique. But we do not yet have an objective function to minimize. There are a number of possibilities:

(a) In general it will not be known at the outset whether a filter of a particular complexity is capable of meeting the specification at all. However, if the values of δ_1 and δ_2 are relaxed (increased) sufficiently, it certainly will be. A suitable approach is to *augment* the parameter vector to include also δ_1 and δ_2 as additional elements. Constraint equations (4.7) still apply but the δ's are now *variables*. We define the objective function to be a suitable linear combination of δ_1 and δ_2, for example

$$F(\mathbf{X}^*) = \delta_1 + \delta_2 \qquad (4.8)$$

where $\mathbf{X}^*$ is used in place of $\mathbf{X}$ to indicate that the parameter vector includes δ_1 and δ_2. The optimization problem becomes: minimize (4.8) subject to constraints (4.7) and the additional constraints that $\delta_1 \geq 0$ $\delta_2 \geq 0$. Assuming that this can be done, we obtain minimum achievable values for δ_1 and δ_2. If neither is below its specified value the complexity of the filter must be increased. If (say) δ_2 met the spec. with some margin but δ_1 was too large, we could repeat the procedure with δ_1 weighted more heavily than δ_2 in the linear combination (4.8).

(b) This second option is similar to the first. The parameter vector is augmented by δ_1 only, and δ_2 is *defined* as $k\delta_1$ where k is a constant equal to the ratio of the specified values, δ_2/δ_1. The objective function is simply δ_1.

(c) δ_2 (or δ_1) can be held *fixed* at some value less than or equal to its specified value and the remaining δ minimized. In this case there may be *no* solution to the constrained minimization problem. If so, the procedure gives no indication of how closely the spec. was approached and this method is accordingly not as useful in the "exploratory" phase of design as (a) or (b).

(d) If we know that the specification *can* be met, we could think in terms of "improving" the filter in some sense. We could minimize δ_1 for a given δ_2 as in (c) above (that is, obtain minimum passband ripple for a given stopband attenuation) or vice versa. Or we could attempt to decrease the width of the transition band, by keeping δ_1 and δ_2 fixed (at their specified values) and minimizing (say) a linear combination of the $B_m(\mathbf{x})$ values just below the stopband edge, e.g.

$$F(\mathbf{x}) = B_{\ell_2}(\mathbf{x}) + B_{\ell_2-1}(\mathbf{x}) + B_{\ell_2-2}(\mathbf{x}) \qquad (4.9)$$

The methods just discussed in relation to a lowpass filter generalize to an arbitrary magnitude specification of the type shown in figure 4.10, where $Y_L(\theta)$ and $Y_U(\theta)$ represent lower and upper limits. As before we define a set of frequencies θ_m, m = 1, 2.. M, and obtain corresponding limits Y_{Lm} and Y_{Um} for each m. If we now define *target responses* Y_m and *tolerances* δ_m

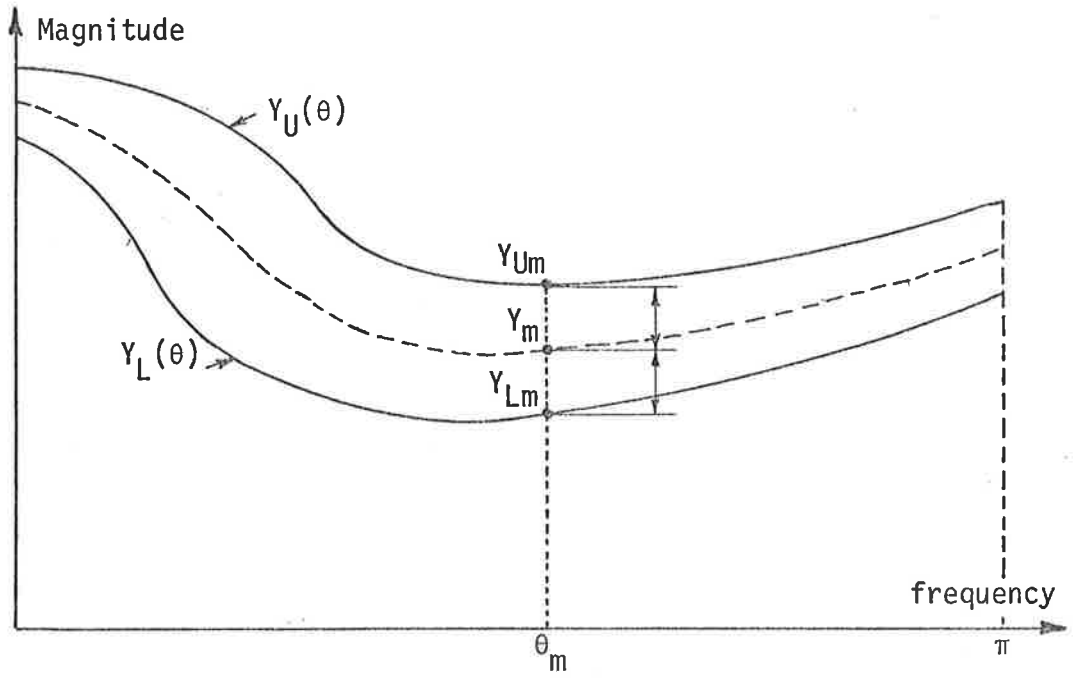


FIGURE 4.10 Arbitrary Magnitude Response Specification

by

$$Y_m = \tfrac{1}{2} (Y_{Lm} + Y_{Um}) \qquad (4.10)$$

and

$$\delta_m = \tfrac{1}{2} (Y_{Um} - Y_{Lm}) \qquad (4.11)$$

we can express the specification by the constraints

$$Y_m - \delta_m < B_m (\mathbf{x}) < Y_m + \delta_m \qquad m = 1, 2..M \qquad (4.12)$$

To obtain a minimization problem we fix all the Y_m values and possibly some of the δ_m, regard the remaining δ's as variables (in addition to the elements of $\mathbf{x}$) and minimize a linear combination of the variable δ's. If the *fixed* δ_m values are realistic there will be a solution, and we have merely to check that the values of the variable δ's thus found are smaller than the corresponding values found from (4.11).

It will now be apparent also that the specification can just as easily relate to some frequency-dependent quantity other than the magnitude response, for example the phase, or the logarithm of the magnitude response. The difference will be in the formula for calculating $B_m(\mathbf{x})$. The abscissa in figure 4.10 does not even need to be frequency - we could be trying to match the first M samples of the impulse response of some system (although in this case the filter being designed would not be an FIR filter, because such an approximation problem would be trivial).

The formulation of the digital filter approximation problem as described above, in terms of generally simple objective functions and a great many (often 2M) constraints will

be referred to as "Formulation A". We have so far assumed tacitly that the solution of such a minimization problem is practical. In fact, the only case when this is ordinarily true is that in which the functions $B_m(\mathbf{x})$ are linear in all the variables x_n, n = 1,2..N and the problem is thus one in linear programming. This is the situation in several approaches to the design of FIR filters and herein lies its main utility (section 4.5). A generalization to IIR filter design of 'Formulation A' is that of Rabiner, Graham and Helms (1974) mentioned further in section 4.6.2.

An obvious advantage in formulating the approximation problem as one in linear programming is that if a solution exists it is globally optimal and standard methods are guaranteed to converge to it in a finite number of operations. In practice the "dual simplex" method is used (in preference to the ordinary simplex method) because there are significantly more constraints than variables. There are also some disadvantages, to be mentioned in connection with the survey of published work in sections 4.5 and 4.6. Certain *nonlinear* formulation A problems of low dimensionality may be amenable to a method of Bandler and Charalambous (1974).

4.4.2 Formulation B - Unconstrained Optimization

For a lowpass filter the specification of upper and lower limits on the variability of the magnitude response via the parameters δ_1 and δ_2 is probably reasonably realistic, since it serves the purpose of isolating the filter design from the likely rather intangible specification for the performance of a signal processing system as a whole. We know

that we want to block high frequencies - the particular value for minimum stopband attenuation may be somewhat arbitrary but at least it allows the application of mathematics to the problem. Such logic would apply also to bandpass, highpass and bandstop filters.

However, for filters with less straightforward characteristics we are unlikely to have rigid specifications for upper and lower limit functions Y_U and $Y_L(\theta)$ as illustrated in figure 4.10. We are much more likely to have a single target response function $Y(\theta)$ or even a set of discrete target responses $\{Y_m(\theta), m = 1, 2...M.\}$ The aim is to make a filter of a specified complexity match such a response "as well as possible". An example of such a requirement is a spectral shaping filter to produce a certain vowel sound in a speech synthesizer. We want to adjust the coefficients of a low-order filter to provide a reasonable fit to a very irregular magnitude spectrum obtained by the spectral analysis of real spoken vowels. The test of acceptability of the design is not whether the difference between achieved and target frequency responses is everywhere less than some particular amount, but is based on listening to the actual sound produced.

Rigid constraints may thus be quite foreign to many filter design (and particularly system modelling) problems. This is fortunate since the constraint formulation (formulation A) was seen above to be mathematically tractable only when the constrained function $B(\mathbf{X}, \theta)$ was linear in the elements of $\mathbf{X}$. We go on now to discuss a much more generally applicable formulation (formulation B) which allows the use of nonlinear unconstrained optimization techniques.

The first step is as before, to select a reasonable set of frequencies $\{\theta_m, m = 1, 2..M\}$ to cover the range $0 < \theta < \pi$. To be general, right from the start, and allow time domain approaches also, the θ_m could (instead of frequencies) be instants of time. If we want our filter to match a target impulse response there is no question of *selection* in the time domain - in this case

$$\theta_m = (m - 1)\, T \tag{4.13}$$

where T is the sampling period ($= 1/F_s$). If we are trying to derive an inverse filter by minimizing an error signal the range of time indices may be different.

The target response at $\theta = \theta_m$ is again written Y_m and the actual response of the filter $B_m(\mathbf{X})$. In principle we want to make all the differences $B_m(\mathbf{X}) - Y_m$ small in magnitude, simultaneously, and think therefore of defining objective functions such as the "sum of squares"

$$F(\mathbf{X}) = \sum_{m=1}^{M} (B_m(\mathbf{X}) - Y_m)^2 \tag{4.14}$$

and the "maximum modulus"

$$F(\mathbf{X}) = \max_m \{|B_m(\mathbf{X}) - Y_m|\} \tag{4.15}$$

and minimizing these with respect to the parameter vector $\mathbf{X}$. The occurrence of an even power (2) in (4.14) and the modulus in (4.15) ensure that both positive and negative deviations from the ideal cause an increase in F.

The "minimax criterion", using (4.15), tends to produce an equiripple type of approximation and is highly compatible with the type of specification considered in section 4.4.1. Unfortunately it does not allow the use of powerful gradient techniques for minimization because the partial derivatives of the objective function (4.15) are not continuous. As $\mathbf{X}$ is varied, $F(\mathbf{X})$ exhibits cusps as the position of maximum $|B_m - Y_m|$ changes from one value of m to another.

The "least squares criterion", using (4.14), has the characteristic that fairly large deviations often result for some m in order to keep the *majority* of deviations small. The ripples in the error function are seldom of equal magnitude so that the criterion is not in tune with rigid "upper and lower" specifications. The least squares criterion is, however, the easiest to handle mathematically, admitting the use of the special techniques of sections 3.3.4 and 3.4.12 as well as ordinary gradient methods and (in many cases) second derivative methods. If the "target response" itself is not accurately determined (as when it is obtained by measurement of a real physical process, which may include noise) then the characteristic of the least squares criterion to ignore isolated points for the sake of an overall smoothing, may be a decided advantage.

A third possibility is a "least pth" criterion, with the objective function defined by

$$F(\mathbf{X}) = \sum_{m=1}^{M} (B_m(\mathbf{X}) - Y_m)^{2p} \qquad (4.16)$$

where p is an integer greater than 1. The even power still ensures increasing contributions from deviations of both

signs. The motivation for using such a function is that large deviations are penalized more heavily than with the least squares criterion, and as p is increased the solution becomes ever closer to the minimax (Deczky (1973), p 76). For a suitably large value of p a good approximation to the minimax solution is obtained while maintaining continuity of the partial derivatives of F and so allowing gradient methods to be used. In practice, minimization is normally carried out for a *sequence* of values of p such as p = 1, 2, 5, 10 because for the higher values the objective function is likely to be well behaved only over a restricted region around the solution and a good starting point will be required.

Unfortunately, although first-derivative gradient methods may be used fairly readily with the least pth criterion, the sums-of-squares and second-derivative methods become quite unwieldy. This matter will be mentioned further in section 4.4.4.

In this thesis the least squares approach is emphasized because of the wide variety of optimization methods which may be used in its solution. Additionally, as discussed above, the resulting filter may for many applications be just as good as that produced by any other method.

4.4.3 Least-Squares Development of Formulation B

In section 4.4.2 we discussed the characteristics of three possible minimization criteria and stated that the least-squares criterion was mathematically the most tractable. In this section it is developed further but still without any restriction on the type of filter response to be optimized.

Equation (4.14) does not represent the most general type of sum-of-squares objective function. Firstly, it is possible

to weight some samples more heavily than others. This may be desirable to force a good fit around sharp peaks in a magnitude spectrum, for example. Thus, making the general definition for the objective function (as in equation 3.45)

$$F(\mathbf{x}) = \sum_{m=1}^{M} f_m^2(\mathbf{x}) \tag{4.17}$$

we may define

$$f_m(\mathbf{x}) = w_m \, (B_m(\mathbf{x}) - Y_m) \tag{4.18}$$

where the weight factors w_m are arbitrary. This will be called "formulation B1". It is directly applicable to time-domain problems, for example to the minimization of the energy in an error sequence (when it becomes even simpler because all $Y_m = 0$). If (say) a cascade-form digital filter is being designed to match a target impulse response, the overall gain constant A_o in equation (4.5) can be set equal to the zero-time target sample, or simpler still to unity, and the target impulse response normalized to have a unit zero-time value. The first sample is then automatically matched and an objective function is defined by (4.17) and (4.18) including only the later samples. A_o does not have to be included in the parameter vector - it is simply ignored.

If we were trying to match a frequency domain magnitude response we could include A_o in the parameter vector and use formulation B1 (4.18). But it is better not to do this, for two reasons. An alternative formulation allows the optimal gain to be calculated analytically, reducing by

one the dimensionality of the parameter vector and thus aiding the optimization algorithm. The second reason for preferring the alternative formulation becomes apparent when we consider discrete optimization of the filter coefficients. The actual value of the gain is irrelevant because we are really interested only in the *shape* of the magnitude response curve. Ad-hoc scaling of data will usually be done at several places in the system via A/D and D/A conversion and (digitally) to maintain precision and to avoid register overflow). If the gain factor is included in the parameter vector and cannot be calculated optimally for any given values of the filter coefficients, it becomes impossible to compare the merits of discrete designs on the basis of *shape*; the objective function (4.18) unreasonably penalizes deviations *from a fixed level*.

Thus in place of (4.18) we make the definition

$$f_m(\mathbf{X}) = w_m(q\, B_m(\mathbf{X}) - Y_m) \qquad (4.19)$$

where q is a factor which scales all the B_m values up or down (together) in such a way that the fit is optimal for any given $\mathbf{X}$. q is thus a function of $\mathbf{X}$ but not of m. Any frequency-independent gain factor in a model (e.g. A_o in (4.5)) can be ignored because it automatically becomes incorporated into q. This will be called "formulation B2". To determine the value of q for a given $\mathbf{X}$ we minimize F (defined by (4.17) and (4.19)) considered as a function of q alone.

$$\frac{dF}{dq} = \sum_{m=1}^{M} 2\, f_m \frac{df_m}{dq}$$

$$= 2 \sum_{m=1}^{M} w_m (q\, B_m - Y_m)\, w_m\, B_m$$

$$= 2\, q \sum_{m=1}^{M} w_m^2 B_m^2 - 2 \sum_{m=1}^{M} w_m^2\, B_m\, Y_m \qquad (4.20)$$

Equating this expression to zero we get

$$q(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m^2 Y_m\, B_m(\mathbf{x})}{\sum_{m=1}^{M} w_m^2 B_m(\mathbf{x})^2} \qquad (4.21)$$

Formulation B2 is applicable whenever the precise value of an overall multiplicative constant is unimportant; for instance it relates as well to a *squared* magnitude response. Further development to obtain formulae for derivatives is necessary if we intend to use a gradient method to minimize F. In what follows the notation will be simplified by using just $\sum$ to mean $\sum_{m=1}^{M}$ and (as in 4.20) by dropping the explicit dependence of B_m and q on $\mathbf{x}$.

The values of the individual f_m are found by first calculating q from (4.21) and substituting this into (4.19). If we want to use a method of the Gauss-Newton type we require also $\frac{\partial f_m}{\partial x_i}$ for all m and for all i (that is, the Jacobian matrix). Differentiating (4.19) we get

$$\frac{\partial f_m}{\partial x_i} = w_m \left(\frac{\partial B_m}{\partial x_i} q + B_m \frac{\partial q}{\partial x_i}\right) \tag{4.22}$$

where $\frac{\partial q}{\partial x_i}$ must of course by found by differentiation of (4.21), that is

$$\frac{\partial q}{\partial x_i} = \frac{(\sum w_m^2 B_m^2)(\sum w_m^2 Y_m \frac{\partial B_m}{\partial x_i}) - 2\,(\sum w_m^2 B_m Y_m)(\sum w_m^2 B_m \frac{\partial B_m}{\partial x_i})}{(\sum w_m^2 B_m^2)^2} \tag{4.23}$$

To use an ordinary first-derivative gradient method we require just the values of $\frac{\partial F}{\partial x_i}$ for all i. We could proceed from formula (3.46) to compute

$$\frac{\partial F}{\partial x_i} = 2 \sum f_m \frac{\partial f_m}{\partial x_i} \tag{3.46}$$

but this requires more work than is necessary. Instead, following Steiglitz (1970), we substitute (4.19) into (4.17) defining a new but equal objective function F* whose dependence on q is explicit, i.e.

$$F^*(x, q) = \sum w_m^2 (q B_m(x) - Y_m)^2 \tag{4.24}$$

By the usual formula for the total derivative

$$\frac{\partial F}{\partial x_i} = \frac{\partial F^*}{\partial x_i} + \frac{\partial F^*}{\partial q} \frac{\partial q}{\partial x_i} \tag{4.25}$$

Because q is selected optimally (by 4.21), we have

$$\frac{\partial F^*}{\partial q} = 0 \qquad \text{and so simply}$$

$$\frac{\partial F}{\partial x_i} = \frac{\partial F^*}{\partial x_i} \tag{4.26}$$

$$= \sum 2\, w_m^2\, (q\, B_m - Y_m)\, q\, \frac{\partial B_m}{\partial x_i} \tag{4.27}$$

$$= 2q^2 \sum w_m^2\, B_m \frac{\partial B_m}{\partial x_i} - 2q \sum w_m^2\, Y_m \frac{\partial B_m}{\partial x_i} \tag{4.28}$$

If we wish to use a second-derivative method we need the complete Hessian matrix $\frac{\partial^2 F}{\partial x_i \partial x_j}$ for all i and j. Again, we could proceed from (3.47) but would need $\frac{\partial^2 f_m}{\partial x_i \partial x_j}$ which in turn requires that (4.23) be differentiated to get $\frac{\partial^2 q}{\partial x_i \partial x_j}$. It is easier to proceed by differentiating (4.25), i.e.

$$\frac{\partial^2 F}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\partial F^*}{\partial x_i} + \frac{\partial F^*}{\partial q}\frac{\partial q}{\partial x_i}\right) + \frac{\partial q}{\partial x_j}\frac{\partial}{\partial q}\left(\frac{\partial F^*}{\partial x_i} + \frac{\partial F^*}{\partial q}\frac{\partial q}{\partial x_i}\right)$$

$$= \frac{\partial^2 F^*}{\partial x_i \partial x_j} + \frac{\partial^2 F^*}{\partial q \partial x_j}\frac{\partial q}{\partial x_i} + \frac{\partial F^*}{\partial q}\frac{\partial^2 q}{\partial x_i \partial x_j}$$

$$+ \frac{\partial^2 F^*}{\partial x_i \partial q}\frac{\partial q}{\partial x_j} + \frac{\partial F^*}{\partial q}\frac{\partial q}{\partial x_j}\frac{\partial^2 q}{\partial x_i \partial x_j} + \frac{\partial^2 F^*}{\partial q^2}\frac{\partial q}{\partial x_j}\frac{\partial q}{\partial x_i} \tag{4.29}$$

Again $\frac{\partial F^*}{\partial q}$ is zero so that the third and fifth terms vanish, and the computation of $\frac{\partial^2 q}{\partial x_i \partial x_j}$ is unnecessary. By differentiating (4.20) we have

$$\frac{\partial^2 F^*}{\partial q^2} = 2 \sum w_m^2 B_m^2 \qquad (4.30)$$

and the other terms in (4.29) are readily calculated as

$$\frac{\partial^2 F^*}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_j} \left(\frac{\partial F^*}{\partial x_i}\right)$$

$$= \frac{\partial}{\partial x_j} \left(2q^2 \sum w_m^2 B_m \frac{\partial B_m}{\partial x_i} - 2q \sum w_m^2 Y_m \frac{\partial B_m}{\partial x_i}\right)$$

$$= 2q^2 \sum w_m^2 \frac{\partial B_m}{\partial x_j} \frac{\partial B_m}{\partial x_i} + 2q^2 \sum w_m^2 B_m \frac{\partial^2 B_m}{\partial x_i \partial x_j}$$

$$- 2q \sum w_m^2 Y_m \frac{\partial^2 B_m}{\partial x_i \partial x_j} \qquad (4.31)$$

and

$$\frac{\partial^2 F^*}{\partial q \partial x_j} = \frac{\partial}{\partial x_j} \left(\frac{\partial F^*}{\partial q}\right)$$

$$= \frac{\partial}{\partial x_j} \left(2 \sum w_m^2 B_m (q B_m - Y_m)\right)$$

$$= \frac{\partial}{\partial x_j} \left(2q \sum w_m^2 B_m^2 - 2 \sum w_m^2 B_m Y_m\right)$$

$$= 4q \sum w_m^2 B_m \frac{\partial B_m}{\partial x_j} - 2 \sum w_m^2 Y_m \frac{\partial B_m}{\partial x_j} \qquad (4.32)$$

Hence, substituting (4.23), (4.30), (4.31) and (4.32) into (4.29) and simplifying gives a final expression for $\frac{\partial^2 F}{\partial x_i \partial x_j}$. To further simplify the notation we introduce the following definitions:

$$
\left.
\begin{aligned}
A_{YY} &= \sum w_m^2 Y_m^2 \\
A_{YB} &= \sum w_m^2 Y_m B_m \\
A_{BB} &= \sum w_m^2 B_m^2 \\
P_{Yi} &= \sum w_m^2 Y_m \frac{\partial B_m}{\partial x_i} \\
P_{Bi} &= \sum w_m^2 B_m \frac{\partial B_m}{\partial x_i} \qquad i = 1, 2..N \\
P_{ij} &= \sum w_m^2 \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} \\
Q_{Yij} &= \sum w_m^2 Y_m \frac{\partial^2 B_m}{\partial x_i \partial x_j} \qquad i = 1, 2..N \\
Q_{Bij} &= \sum w_m^2 B_m \frac{\partial^2 B_m}{\partial x_i \partial x_j} \qquad j = 1, 2..N
\end{aligned}
\right\} \quad (4.33)
$$

We now have the simplified formulae

$$q = A_{YB}/A_{BB} \tag{4.34}$$

and

$$\frac{\partial q}{\partial x_i} = \frac{P_{Yi} - 2q\, P_{Bi}}{A_{BB}} \tag{4.35}$$

which we simplify further to t_i/A_{BB}, defining

$$t_i = P_{Yi} - 2q\, P_{Bi} \tag{4.36}$$

Expressions for the function value and derivatives are now

$$F = A_{YY} - q\, A_{YB} \tag{4.37}$$

$$g_i = \frac{\partial F}{\partial x_i} = 2q^2 P_{Bi} - 2q\, P_{Yi} \tag{4.38}$$

$$h_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j} = 2q^2(Q_{Bij} + P_{ij}) - 2q\, Q_{Yij} - \frac{2t_i t_j}{A_{BB}} \qquad (4.39)$$

If the Gauss-Newton matrix is required, its (i,j)th element is given by

$$r_{ij} = 2q^2 P_{ij} + \frac{2}{A_{BB}} (P_{Yi}P_{Yj} - q(P_{Bi}P_{Yj} + P_{Bj}P_{Yi})) \qquad (4.40)$$

The general approach for evaluating function values and derivatives is now clear, and it is basically independent of the type of response to be optimized. Some or all of the sums in (4.33) must be accumulated for m = 1, 2...M, and for fairly large M (compared with N) this forms the bulk of the computation. Actual evaluation of the elements of the gradient vector and Gauss-Newton or Hessian matrices is then straightforward (equations (4.37) to (4.40))and is independent of M. A "function value only" method requires accumulation of A_{YY}, A_{YB} and A_{BB} only, a first-derivative gradient method requires P_{Yi} and P_{Bi} in addition, a Gauss-Newton method P_{ij} also, and a second derivative method, Q_{Yij} and Q_{Bij} as well as the rest. The type of response being optimized has not entered into any of the above, but of course it determines the form of the functions B_m, $\frac{\partial B_m}{\partial x_i}$ and $\frac{\partial^2 B_m}{\partial x_i \partial x_j}$.

As the optimization procedure approaches the solution the values of the gradient components g_i become small, and there is a problem with numerical cancellation in determining them from (4.38). If the fit of the model to the target response is very good there is a similar problem in calculating F from (4.37). The alternative is to calculate the individual

f_m and $\frac{\partial f_m}{\partial x_i}$ and evaluate the sums in (4.17) and (3.46). However, much more computation is then involved. The experience of the present author (working with 60-bit floating-point arithmetic) is that adequate convergence is usually obtained before cancellation becomes important.

"Formulation B2" as developed above is useful when there is an arbitrary multiplicative gain constant. If, however, the specification had related to the *logarithm* of the magnitude spectrum the arbitrary constant would be *additive*. Another similar situation arises if we are trying to produce a linear phase characteristic. An equivalent requirement is *constant* group delay (phase slope), and this is usually easier to handle because group delay expressions are simpler than those for phase. An additive constant may be introduced (optimally) because it is the *flatness* of the group delay response that is important, not the precise value of delay. This leads to "formulation B3" in which we define

$$f_m(\mathbf{x}) = w_m\,(B_m(\mathbf{x}) - Y_m - q) \tag{4.41}$$

and (4.17) still applies. As before we find q by equating to zero the derivative of the objective function considered as a function of q alone, i.e.

$$F(\mathbf{x}) = F^*(\mathbf{x},q) = \sum_{m=1}^{M} w_m{}^2(B_m(\mathbf{x}) - Y_m - q)^2 \tag{4.42}$$

$$\frac{\partial F^*}{\partial q} = -2 \sum_{m=1}^{M} w_m^2 (B_m(\mathbf{x}) - Y_m - q) = 0$$

$$\therefore\ q(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m^2 (B_m(\mathbf{x}) - Y_m)}{\sum_{m=1}^{M} w_m^2} \tag{4.43}$$

We again drop the limits on the sums and the explicit dependence of q and B_m on $\mathbf{x}$ to simplify the notation, and make the following definitions

$$\left.\begin{aligned}
W &= \sum w_m^2 \\
A_1 &= \sum w_m^2 (B_m - Y_m) \\
A_2 &= \sum w_m^2 (B_m - Y_m)^2 \\
P_{1i} &= \sum w_m^2 \frac{\partial B_m}{\partial x_i} \\
P_{2i} &= \sum w_m^2 (B_m - Y_m) \frac{\partial B_m}{\partial x_i} \qquad \left.\begin{aligned} i &= 1, 2..N \end{aligned}\right\} \\
P_{ij} &= \sum w_m^2 \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} \\
Q_{1ij} &= \sum w_m^2 \frac{\partial^2 B_m}{\partial x_i \partial x_j} \qquad \left.\begin{aligned} i &= 1, 2..N \\ j &= 1, 2..N \end{aligned}\right\} \\
Q_{2ij} &= \sum w_m^2 (B_m - Y_m) \frac{\partial B_m}{\partial x_i \partial x_j}
\end{aligned}\right\} \tag{4.44}$$

Values of $F(\mathbf{x})$ and its first derivatives are readily found in terms of these quantities

$$\begin{aligned}
F &= \sum w_m^2 (B_m - Y_m - q)^2 \\
&= \sum w_m^2 (B_m - Y_m)^2 - 2q \sum w_m^2 (B_m - Y_m) + q^2 \sum w_m^2 \\
&= A_2 - 2q A_1 + q^2 W
\end{aligned} \tag{4.45}$$

but, from (4.43),

$$q = A_1/W \tag{4.46}$$

and so

$$q^2W = q\,A_1 \tag{4.47}$$

and (4.45) simplifies to

$$F = A_2 - q\,A_1 \quad . \tag{4.48}$$

Since we have forced $\frac{\partial F^*}{\partial q} = 0$, we have

$$\frac{\partial F}{\partial x_i} = \frac{\partial F^*}{\partial x_i} = 2 \sum w_m^2 (B_m - Y_m - q) \frac{\partial B_m}{\partial x_i}$$

$$= 2 \sum w_m^2 (B_m - Y_m) \frac{\partial B_m}{\partial x_i} - 2q \sum w_m^2 \frac{\partial B_m}{\partial x_i}$$

$$= 2\,P_{2i} - 2q\,P_{ii} \tag{4.49}$$

If the elements of the Gauss-Newton matrix are required, we proceed as follows

$$f_m = w_m (B_m - Y_m - q) \tag{4.41}$$

$$\therefore \frac{\partial f_m}{\partial x_i} = w_m \left(\frac{\partial B_m}{\partial x_i} - \frac{\partial q}{\partial x_i}\right) \tag{4.50}$$

Now since

$$r_{ij} = 2 \sum \frac{\partial f_m}{\partial x_i} \frac{\partial f_m}{\partial x_j} \tag{4.51}$$

we can expand using (4.50) as

$$r_{ij} = 2 \sum w_m^2 \left(\frac{\partial B_m}{\partial x_i} - \frac{\partial q}{\partial x_i}\right)\left(\frac{\partial B_m}{\partial x_j} - \frac{\partial q}{\partial x_j}\right)$$

$$= 2 \sum w_m^2 \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} - 2 \frac{\partial q}{\partial x_i} \sum w_m^2 \frac{\partial B_m}{\partial x_j}$$

$$- 2 \frac{\partial q}{\partial x_j} \sum w_m^2 \frac{\partial B_m}{\partial x_i} + 2 \frac{\partial q}{\partial x_i} \frac{\partial q}{\partial x_j} \sum w_m^2$$

$$= 2 \left(P_{ij} - \frac{P_{1i}}{W} P_{1j} - \frac{P_{1j}}{W} P_{1i} + \frac{P_{1i}}{W} \frac{P_{1j}}{W} W\right)$$

$$= 2 \left(P_{ij} - \frac{P_{1i} P_{1j}}{W}\right) \tag{4.52}$$

where we have substituted

$$\frac{\partial q}{\partial x_i} = \frac{P_{1i}}{W} \tag{4.53}$$

as is obvious by differentiating (4.43). In this formulation it is easy to obtain an expression for $\frac{\partial^2 q}{\partial x_i \partial x_j}$, by twice differentiating, thus

$$\frac{\partial^2 q}{\partial x_i \partial x_j} = \frac{Q_{1ij}}{W} \tag{4.54}$$

Because of the simplicity of this expression it is straightforward to calculate the full Hessian matrix using (3.47). Thus we augment the terms given by (4.52) by the amount $2 \sum f_m \frac{\partial^2 f_m}{\partial x_i \partial x_j}$, which is equal to

$$2 \sum w_m^2 (B_m - Y_m - q)\left(\frac{\partial^2 B_m}{\partial x_i \partial x_j} - \frac{\partial^2 q}{\partial x_i \partial x_j}\right)$$

$$= 2 \sum w_m^2 (B_m - Y_m) \frac{\partial^2 B_m}{\partial x_i \partial x_j} - 2q \sum w_m^2 \frac{\partial^2 B_m}{\partial x_i \partial x_j}$$

$$- \frac{2Q_{1ij}}{W} \sum w_m^2 (B_m - Y_m) + \frac{2q\, Q_{1ij}}{W} \sum w_m^2$$

$$= 2\, Q_{2ij} - 2q\, Q_{1ij} - \frac{2\, Q_{1ij}\, A_1}{W} + \frac{2q\, Q_{1ij}\, W}{W}$$

$$= 2\,(Q_{2ij} - q\, Q_{1ij}) \qquad (4.55)$$

The final expression for the (i,j)th element of the Hessian, by adding (4.52) and (4.55), is

$$h_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j} = 2\,(P_{ij} - \frac{P_{1i} P_{1j}}{W} + Q_{2ij} - q\, Q_{1ij}) \qquad (4.56)$$

Hence, in the case of an additive constant the procedure is similar to that of a multiplicative constant, that is, sums-of-products are accumulated (equations (4.44)) and the values of functions and derivatives are then found from a set of equations not involving M, i.e. (4.46), (4.48), (4.49), (4.52) and (4.56).

Having developed formulations B2 and B3 are far as possible without specifying the type of response to be optimized, we now return to derive similar expressions for formulation B1. These are much simpler, and we can make use of

some of the shorthand symbols for sums-of-products introduced in (4.44). From (4.18)

$$f_m = w_m(B_m - Y_m) \tag{4.18}$$

so that the function is given by

$$F = \sum w_m^2(B_m - Y_m)^2 = A_2 \tag{4.57}$$

For the gradient components

$$\frac{\partial F}{\partial x_i} = \sum 2\, f_m \frac{\partial f_m}{\partial x_i}$$

$$= \sum 2\, w_m^2(B_m - Y_m) \frac{\partial B_m}{\partial x_i} = 2\, P_{2i} \tag{4.58}$$

for the Gauss-Newton matrix

$$r_{ij} = \sum 2 \frac{\partial f_m}{\partial x_i} \frac{\partial f_m}{\partial x_j}$$

$$= \sum 2\, w_m^2 \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} = 2\, P_{ij} \tag{4.59}$$

and for the Hessian

$$h_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j} = 2\, P_{ij} + \sum 2\, f_m \frac{\partial^2 f_m}{\partial x_i \partial x_j}$$

$$= 2\, P_{ij} + \sum 2\, w_m^2(B_m - Y_m) \frac{\partial^2 B_m}{\partial x_i \partial x_j}$$

$$= 2\, P_{ij} + 2\, Q_{2ij} \tag{4.60}$$

This completesthe general development of the least-squares version of formulation B.

4.4.4 The Least pth Error Criterion

We now return to the least pth error criterion to indicate why it is often more difficult to use than least squares, particularly with Gauss-Newton and second derivative methods.

With the formulation analogous to "formulation B1", that is with no additive or multiplicative gain value q, the situation is still tractable. For simplicity, weight factors are all assumed to be unity. Equation (4.16) may be re-written to set F in the form of a sum-of-squares, i.e.

$$F(\mathbf{x}) = \sum_{m=1}^{M} f_m^2(\mathbf{x}) \tag{4.61}$$

where

$$f_m(\mathbf{x}) = (B_m(\mathbf{x}) - Y_m)^p \tag{4.62}$$

The gradient components are

$$g_i = \frac{\partial F(\mathbf{x})}{\partial x_i} = \sum 2 f_m \frac{\partial f_m}{\partial x_i}$$

$$= \sum 2p \, (B_m - Y_m)^{2p-1} \frac{\partial B_m}{\partial x_i} \quad . \tag{4.63}$$

The element of the Gauss-Newton matrix **R** are

$$r_{ij} = \sum 2 \frac{\partial f_m}{\partial x_i} \frac{\partial f_m}{\partial x_j}$$

$$= \sum 2p^2 \, (B_m - Y_m)^{2p-2} \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} \tag{4.64}$$

and those of the Hessian matrix **H** are

$$h_{ij} = \sum 2f_m \frac{\partial^2 f_m}{\partial x_i \partial x_j} + r_{ij}$$

$$= \sum 2p\,(B_m - Y_m)^{2p-1} \frac{\partial^2 B_m}{\partial x_i \partial x_j}$$

$$+ \sum (4p^2 - 2p)(B_m - Y_m)^{2p-2} \frac{\partial B_m}{\partial x_i} \frac{\partial B_m}{\partial x_j} \quad . \qquad (4.65)$$

The complication arises when an optimal gain term q must be allowed for, as with formulations B2 and B3. Consider the case of an additive gain term, so that the objective function F is given by

$$F(\mathbf{X}) = F^*(\mathbf{X}, q) = \sum (B_m - Y_m - q)^{2p} \qquad (4.66)$$

The optimal value of q is again indicated by setting $\frac{\partial F^*}{\partial q}$ to zero, i.e.

$$-2p \sum (B_m - Y_m - q)^{2p-1} = 0 \qquad (4.67)$$

For p >1, (4.67) cannot be reduced to an explicit expression for q. Rather, it provides a polynomial equation of degree (2p - 1) in q. For example, if p = 2, we have the cubic equation

$$q^3 + aq^2 + bq + c = 0 \qquad (4.68)$$

where

$$\left.\begin{aligned} a &= -\ 3/M \sum (B_m - Y_m) \\ b &= +\ 3/M \sum (B_m - Y_m)^2 \\ c &= -\ 1/M \sum (B_m - Y_m)^3 \end{aligned}\right\} \quad (4.69)$$

There are two methods which may be used to find the value of q. Firstly, the coefficients of the polynomial may be evaluated through equations such as (4.69) and Newton's (iterative) method then used to obtain the solution. Fortunately, an adequate initial estimate for this process is usually obtainable by setting p = 1 and solving (4.67) analytically (the optimal "least squares" value of q.) The second method for obtaining q, which may be considerably more practical when p is large, involves evaluating F for a number of trial values of q. That is, a one-dimensional search is performed to locate the minimum. This search may be of the "golden ratio" or "polynomial interpolation" type, as discussed in section 3.5. The least-squares solution for q again provides a starting point.

When the optimal value of q has been calculated, the components of the gradient vector are readily found, for

$$g_i = \frac{\partial F}{\partial x_i} = \frac{\partial F^*}{\partial x_i} + \frac{\partial F^*}{\partial q}\frac{\partial q}{\partial x_i} \quad (4.70)$$

Again, $\frac{\partial F^*}{\partial q}$ has been forced to vanish, so that

$$g_i = \frac{\partial F^*}{\partial x_i} = \sum 2p\ (B_m - Y_m - q)^{2p-1}\ \frac{\partial B_m}{\partial x_i} \quad . \quad (4.71)$$

First-derivative gradient methods are thus still practical

with the least pth error criterion when there is an additive gain term q. The main complication attending the use of $p > 1$ is the necessity for an iterative procedure to obtain the appropriate value of q. Second-derivative (and Gauss-Newton) methods, however, become vastly more complicated, and increasingly so as p increases. For example, the elements of the Hessian matrix are found by differentiating (4.70), which gives

$$h_{ij} = \frac{\partial^2 F^*}{\partial x_i \partial x_j} + \frac{\partial^2 F^*}{\partial q \partial x_i}\frac{\partial q}{\partial x_j} + \frac{\partial^2 F^*}{\partial q \partial x_j}\frac{\partial q}{\partial x_i} + \frac{\partial^2 F^*}{\partial q^2}\frac{\partial q}{\partial x_i}\frac{\partial q}{\partial x_j} \qquad (4.72)$$

The derivatives of F^* are all easily computable; the trouble is with terms such as $\frac{\partial q}{\partial x_i}$, because equation (4.67) does not provide an explicit formula for q. With much labour, all the values required are available. Again taking the case $p = 2$ as an example, and differentiating (4.68) with respect to x_i :

$$3q^2 \frac{\partial q}{\partial x_i} + 2aq \frac{\partial q}{\partial x_i} + q^2 \frac{\partial a}{\partial x_i} + b \frac{\partial q}{\partial x_i} + q \frac{\partial b}{\partial x_i} + \frac{\partial c}{\partial x_i} = 0 \qquad (4.73)$$

whence

$$\frac{\partial q}{\partial x_i} = - \frac{q^2 \frac{\partial a}{\partial x_i} + q \frac{\partial b}{\partial x_i} + \frac{\partial c}{\partial x_i}}{3q^2 + 2aq + b} \qquad (4.74)$$

where $\frac{\partial a}{\partial x_i}$, $\frac{\partial b}{\partial x_i}$ and $\frac{\partial c}{\partial x_i}$, may in turn be found by differentiating expressions (4.69).

The complexity of all the above expressions and the

attendent computational load would seem to make second-derivative and Gauss-Newton methods unattractive in conjunction with the least pth criterion. This is not to say that experiment would necessarily show that they were not worthwhile. However, in the comparisons reported in this thesis I have preferred to concentrate on the simpler least squares criterion.

4.5 Design of Finite Impulse Response Digital Filters

Optimization techniques, and in particular linear programming, have proven very useful in the design of FIR filters. In this section is a brief survey of the relevant literature.

We wish to design a filter having an impulse response of length N samples, denoted h_0, h_1, ...h_{N-1}, from a frequency-domain specification of magnitude and phase. The use of N in this section is traditional, and not to be confused with its use in this thesis to denote dimensionality of the parameter vector. An obvious procedure is to *sample* the frequency response at N frequencies (uniformly spaced around the unit circle, and symmetric about zero frequency) and obtain an impulse response using the inverse discrete Fourier transform (IDFT). Of course, the real part of the frequency response must be specified as an even function and the imaginary part as an odd function, in order to obtain a real impulse response. Figure 4.11 (after Lockhart, 1975) shows the frequency response of a lowpass filter designed in this manner, with N = 30 and all the passband samples set to the same value and all other

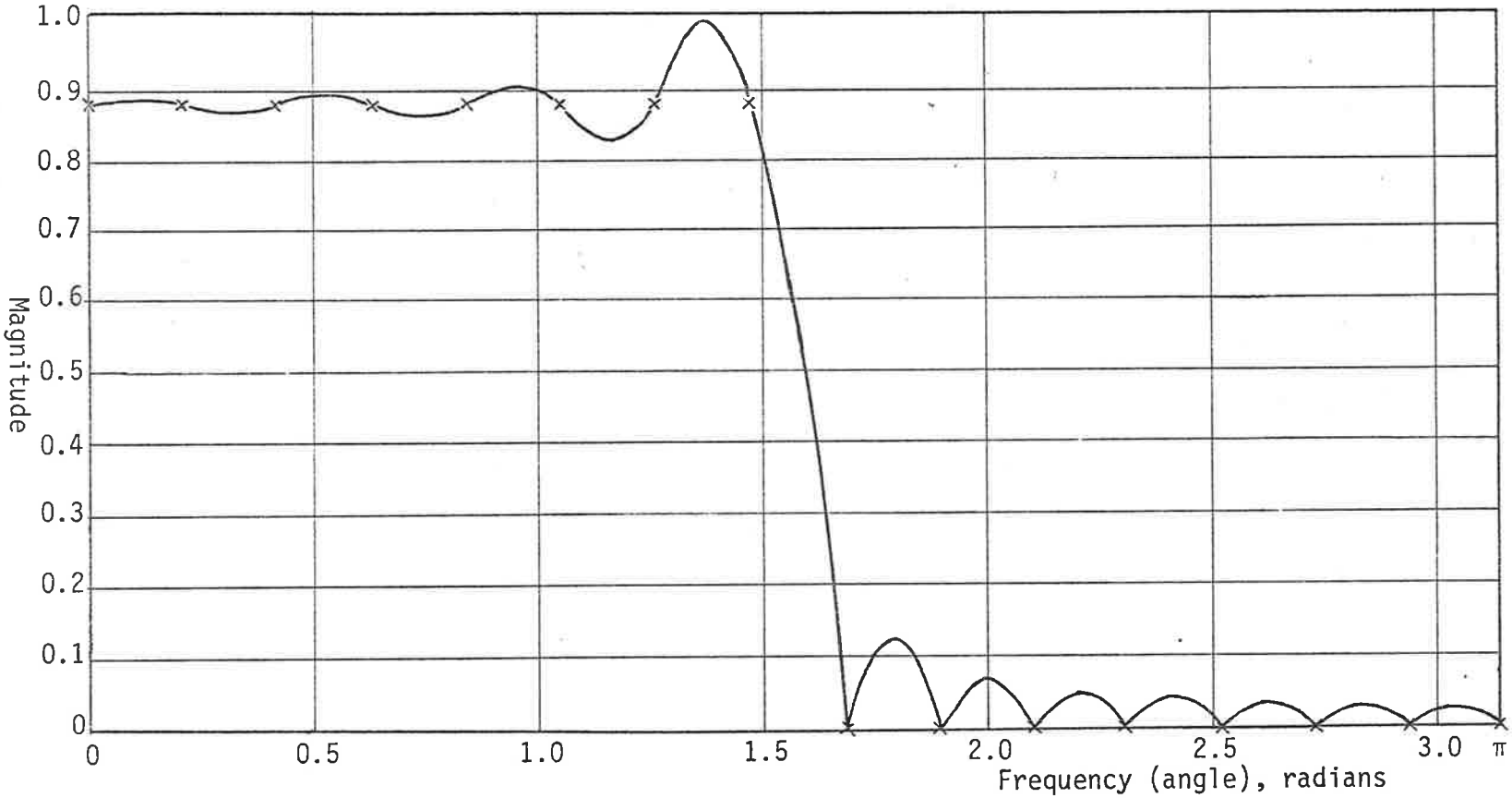


FIGURE 4.11 Magnitude Response of FIR Filter, 15-point 1-0 Specification (from Lockhart, 1975)

samples to zero. The achieved response is exact at each of the sampling frequencies but shows unacceptable deviations in between, particularly in the vicinity of the discontinuity (passband edge).

One approach to improving this situation is to multiply the impulse response by a *window function*. The effect in the frequency domain is to convolve the curve in figure 4.11 with the Fourier transform of the window, and with a good choice of window this can result in smoothing of the ripples (but with an unavoidable widening of the transition band). This approach does not expressly involve optimization techniques and will not be considered further.

The second method, due originally to Gold and Jordan (1969) and improved by Rabiner, Gold and McGonegal (1970), recognises in advance that the transition band will have to be reasonably wide in order to produce low ripple in pass- and stopbands. Accordingly, one or more frequency samples are *allocated* to the transition band and their values are taken as the variables in an optimization procedure. The frequency samples in pass- and stopbands are fixed, as before, and in the final design there will still be an exact fit at these frequencies. The beauty of the procedure arises because the magnitude response (at any given frequency) is a *linear* function of the variable frequency samples, and formulation A (section 4.4.1) can be applied. As discussed in that section, the frequency response must be evaluated at a sufficiently dense set of frequencies to ensure that the specifications on pass- and stopband ripple are met. Interpolation factors of 16:1 (Rabiner, Gold and McGonegal, 1970) and 8:1 (Rabiner,

1972) between the DFT frequency samples have been suggested. If we considered that only one transition coefficient were variable, we could plot the magnitude response at each of the (interpolated) frequencies in the stopband, and obtain the linear variation shown in figure 4.12. The lowest point on the upper envelope of all these curves defines that value of the coefficient which minimizes the maximum stopband magnitude response (ripple). This envelope (darkened in figure 4.12) bounds a convex region, and so a search procedure which follows a descent path along the envelope must find the global solution. The situation generalizes to the case of several variable transition coefficients, the lines of figure 4.12 becoming hyperplanes. If there are M variable transition coefficients the global solution will be determined by the intersection of M + 1 hyperplanes, implying equal maximum ripple values at M + 1 different frequencies. Rabiner, Gold and McGonegal (1970) (apparently not realizing the applicability of linear programming to this problem) devised a search procedure which was capable of converging to such a vertex in the (M + 1) - dimensional space. They obtained very useful results for linear phase lowpass and bandpass filters and for wideband differentiators using this technique. The disadvantages are that there is no control on the ripple in the passband (although there seems no reason why it could not be incorporated) and that the search procedure is very time-consuming (compared with linear programming). The computer time requirement grows exponentially with M, and the authors found that M = 4 was the maximum reasonably attainable.

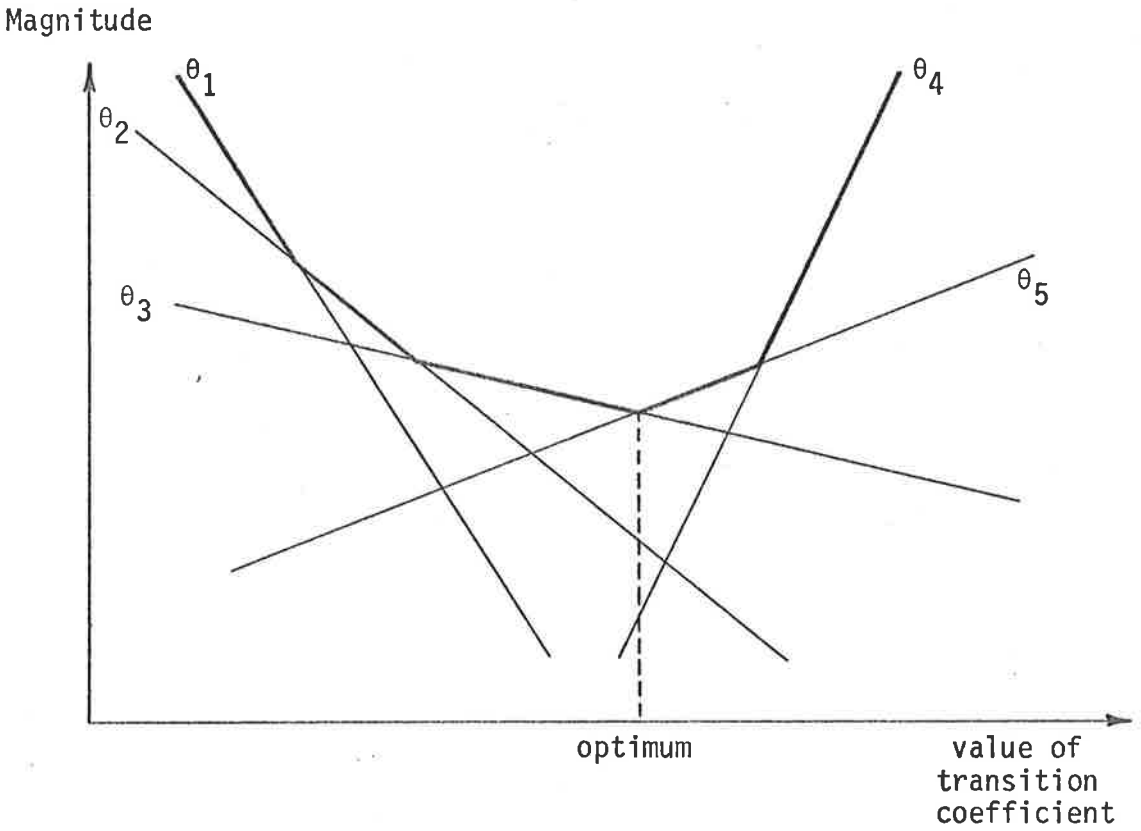


FIGURE 4.12 Linear Variation of Magnitude Response with Value of One Transition Coefficient, FIR Digital Filter

The linear programming approach to this problem of frequency-sampling design was presented by Rabiner (1972). The method is essentially that of 'formulation A' previously described in this thesis. The formula describing the linear dependence of the magnitude response on the values of variable frequency samples had been given by Rabiner and Schafer (1971) for linear-phase filters. With no constraint on the inband ripple, and the objective function defined as δ_2 (the peak stopband ripple), the linear program solution would be the same as that of the older search procedure. The previously noted equality of stopband ripple maxima at M + 1 distinct frequencies is readily interpreted as the usual linear programming principle, that at the solution p constraints become equalities, where p is the number of independent variables. In this case p is M + 1, being M transition samples and one ripple value (δ_2). Rabiner (1972) instead chose to constrain also the passband ripple to a particular value, and obtained curves depicting the tradeoff relations between δ_1 and δ_2 for linear-phase, lowpass FIR filters of varying bandwidths, assuming three transition coefficients variable.

The technique of linear programming is sufficiently powerful to handle many more variables than the few used in typical frequency-sampling designs. Thus in principle we can allow all of the frequency samples to be variable, obtaining solutions with a large number of equal ripple maxima which in fact come very close to the optimum filters in the minimax sense. The magnitude frequency response is also a linear function of the impulse response coefficients themselves (and

is related more simply to them than to the frequency samples), so that the linear program may alternatively be set up in terms of coefficients. This technique was pioneered by Cavin, Ray and Rhyne (1969). Rabiner, in his 1972 paper, designs linear-phase lowpass filters with up to 99 variable coefficients using this method. (Only 50 coefficients are independent variables because the linear-phase requirement is met by forcing symmetry on the impulse response.) He also applies the process to wideband differentiators with 32 coefficients. In this case the impulse response is restricted to be antisymmetric about its mid-point to guarantee the purely imaginary frequency response expected of a differentiator (where a constant delay necessary for realizability is ignored). The linearly-rising magnitude response can readily be approximated in an equiripple absolute or equiripple relative error sense.

Rabiner also gives an example of a lowpass filter whose time-domain (step response) ripple is constrained in addition to the frequency domain ripple. This is possible because the step response, too, is a linear function of the filter coefficients. The linear programming approach has also been extended to the design of two-dimensional FIR filters by Hu (1971). Linear programming has thus become firmly established as a design tool for FIR digital filters, and a wide variety of tradeoff opportunities is available within the standard "formulation A" framework.

Nonlinear optimization techniques have also found application in the field of FIR filter design. Herrmann (1970) sets up equations which explicitly force the passband and stopband ripple to exhibit a certain number of extrema of a pre-

assigned value. This set of nonlinear equations is then solved by a suitable iterative technique to yield the values of the impulse response coefficients. Suitable solution techniques include those which minimize the sum of squares of residuals, such as the damped Gauss-Newton and Marquardt methods (section 3.4.12). (Herrmann's paper does not identify the method originally used.) These methods ultimately achieve quadratic convergence in this case because the residuals are all zero at the solution (see 3.4.12).

To illustrate this use of nonlinear optimization for FIR filters in more detail, we consider the case of a linear phase filter of length N, an odd number. To achieve the linear phase the impulse response is symmetrical about its centre sample, and so there are only $\frac{N+1}{2}$ independent coefficients. The magnitude response $S(\theta)$ is given by

$$S(\theta) = |H(\exp(j\theta))|$$

$$= h_o + \sum_{n=1}^{(N-1)/2} 2h_n \cos n\theta \qquad (4.75)$$

and we can readily evaluate its derivative also, as

$$\frac{dS(\theta)}{d\theta} = \sum_{n=1}^{(N-1)/2} (-2n\, h_n \sin n\theta) \qquad (4.76)$$

The conditions for the ripple to exhibit an extremum of the correct magnitude at a particular frequency θ_ℓ are that

$$\left.\begin{aligned} S(\theta_\ell) &= 1 \pm \delta_1 \\ \text{or} \qquad S(\theta_\ell) &= \delta_2 \end{aligned}\right\} \text{depending on whether } \theta_\ell \text{ is in passband or stopband} \tag{4.77}$$

and

$$\left|\frac{dS(\theta)}{d\theta}\right|_{\theta=\theta_\ell} = 0$$

We suppose that there are L such "interior" extrema at frequencies as yet undetermined, but related through 2L equations of the form (4.75) and (4.76). In addition we require $S(0) = 1 \pm \delta$, and $S(\pi) = \delta_2$ giving two more equations from (4.75) (Equation (4.76) is automatically satisfied for $\theta = 0$ and $\theta = \pi$). We thus have $2L + 2$ equations in $\frac{N+1}{2} + L$ unknowns, and for a unique, exact solution these must be equal. Thus we have

$$L = \frac{N-3}{2} \tag{4.78}$$

With values of δ_1 and δ_2 fixed a priori, the only remaining freedom in the design is the allocation of a certain number of the $\frac{N-3}{2}$ extrema to the passband and the remainder to the stopband. There are thus only a finite number of these "equiripple" filters for a given δ_1 and δ_2. The passband and stopband edges are not directly controllable by this technique and they are unlikely to meet an arbitrary specification exactly. They are, however, optimal approximations in the minimax sense over the pass- and stopbands actually attained. In this respect the linear programming designs of Rabiner (1972) are slightly sub-optimal, having one ripple fewer than

the theoretical maximum. This difference is insignificant in practice, however, and both design methods are of considerable use. The nonlinear equation approach is reported to be somewhat superior in computer time requirements (Rabiner, 1972).

Further work on optimal minimax approximations for linear phase FIR filters has been reported by Hofstetter, Oppenheim and Siegel (1971) and by Parks and McClellan (1972). Both methods employ polynomial interpolation techniques and are considerably more time-efficient than those using standard linear and nonlinear optimization techniques as discussed. The procedure of Hofstetter et. al. generates the same filters as does the Hermann technique. The more general method of Parks and McClellan can obtain the optimum filter for any specified band edges, producing the Hofstetter designs as special cases.

The methods of Athanassopoulos and Kaiser (1970) using an unconstrained minimization algorithm are applicable to FIR design but are more usefully discussed in relation to IIR filters, as in the next section.

4.6 Design of Infinite Impulse Response Filters in the Frequency Domain

4.6.1 General

When an IIR filter is to be designed to meet specifications of the lowpass, bandpass, highpass or band stop type, with phase response neglected, two methods are available which do not employ optimization techniques. The first involves transformation of a classical (continuous) filter design using

the bilinear transformation, which maps the left hand half of the s (Laplace variable) plane onto the interior of the unit circle in the z plane. The second method allows a direct approximation in the z plane. These techniques are discussed in Bogner and Constantinides (1975), chapters 4 and 5 respectively, but since optimization techniques are not employed they will not be considered further.

When the specified magnitude response is not of the classical form (often referred to in the literature as "piecewise-constant", although this term would seem to include also multiband filters, which are certainly not amenable to classical techniques) some kind of optimization approach is necessary. Methods employing both "formulation A" (multi-constraint) and "formulation B" (essentially unconstrained) have been suggested, and published formulation A approaches have used both linear and nonlinear programming.

4.6.2 "Formulation A" Approaches

The linear programming approach was apparently first proposed for IIR digital filters by Thajchayapong and Rayner (1973), building on the work of Matthews, Davis, McKee, Cavin and Sibbles (1963) for continuous filters. The original Thajchayapong and Rayner paper applied the technique to a lowpass filter design and showed that the results were very similar to a Chebyshev design obtained by classical methods. In a later paper (1974) they suggested that the technique was applicable to an arbitrary magnitude response and gave a wideband differentiator example. These authors considered specifications of the "upper and lower limit" variety. Rabiner,

Graham and Helms (1974) employed similar techniques to obtain the optimum minimax approximation to a given arbitrary target function $Y(\theta)$.

To arrange the approximation problem into a form amenable to linear programming, Rabiner et. al. work with a high-order polynomial and with the square of the magnitude response. Both of these features are unfortunate because they mean that all computations must be carried out to extreme accuracy (the former because of the usual coefficient sensitivity problem with polynomials, and the latter because it doubles the precision with which response quantities must be compared - a tolerance of 10^{-5} on the magnitude response is equivalent to 10^{-10} on squared magnitude). Rabiner et. al. find that these factors limit the usefullness of the method where high-order, sharp cutoff and close-tolerance filters are concerned. The amount of computer time is also quite high (with double-precision arithmetic) and would be mugh higher if the precision were extended.

The approach of Rabiner et. al. is as follows. The magnitude-squared function of a digital filter, $M(\theta)$, may be evaluated as

$$M(\theta) = \left[H(z)H(z^{-1})\right]_{z=\exp(j\theta)} \tag{4.79}$$

Substituting for $H(z)$ from (4.3), we get

$$H(z)H(z^{-1}) = \left(\sum_{i=0}^{m} \alpha_i z^{-i}\right)\left(\sum_{j=0}^{m} \alpha_j z^{+j}\right) \Big/ \left(\sum_{i=0}^{n} \beta_i z^{-i}\right)\left(\sum_{j=0}^{n} \beta_j z^{+j}\right) \tag{4.80}$$

where $\beta_0 = 1$.

Defining suitable constants c_i and d_i (sums of products of the α_i and β_i terms) this becomes

$$H(z)H(z^{-1}) = \frac{c_o + \sum_{i=1}^{m} c_i (z^{-i} + z^{+i})}{1 + \sum_{i=1}^{m} d_i (z^{-i} + z^{+i})} \qquad (4.81)$$

and putting $z = \exp(j\theta)$, or equivalently

$$z^{-i} + z^{+i} = 2 \cos i\theta \qquad (4.82)$$

we get

$$M(\theta) = \frac{c_o + \sum_{i=i}^{m} 2\, c_i \cos i\theta}{1 + \sum_{i=1}^{n} 2\, d_i \cos i\theta} = \frac{N(\theta)}{D(\theta)} \qquad (4.83)$$

If the achieved response is to match the target response $Y(\theta)$ to within a tolerance $\delta(\theta)$, we have

$$-\delta(\theta) \leqslant \frac{N(\theta)}{D(\theta)} - Y(\theta) \leqslant \delta(\theta) \qquad (4.84)$$

Constraining $D(\theta)$ to be always positive (as it in fact is for a real filter) we can multiply through in (4.84) and so obtain the set of constraints

$$\left.\begin{aligned} N(\theta) - D(\theta)\left[Y(\theta) + \delta(\theta)\right] &\leqslant 0 \\ -N(\theta) + D(\theta)\left[Y(\theta) - \delta(\theta)\right] &\leqslant 0 \\ -N(\theta) &\leqslant 0 \\ -D(\theta) &\leqslant 0 \end{aligned}\right\} \qquad (4.85)$$

which are *linear* in the unknowns c_i and d_i, and which must

be satisfied by any filter meeting the specification. Clearly we can satisfy a *modified* set of constraints

$$\left.\begin{aligned} N_m - D_m\left[Y_m + \delta_m\right] - q &\leq 0 \\ -N_m + D_m\left[Y_m - \delta_m\right] - q &\leq 0 \\ -N_m \qquad\quad - q &\leq 0 \\ -D_m \qquad\quad - q &\leq 0 \end{aligned}\right\} \quad (4.86)$$

for some (suitably large, non-negative) value of the auxiliary variable q, where the N_m, D_m, Y_m and δ_m denote $N(\theta)$, $D(\theta)$, $Y(\theta)$ and $\delta(\theta)$ evaluated at the mth of a suitably dense set of discrete frequencies , θ_m. If there is a solution to (4.86) such that q = 0, then constraints (4.85) are approximately satisfied also. The procedure, then, is to solve the linear program defined by constraints (4.86), with the objective function F = q.

If the solution has q = 0, then the spec. can be met. If the design objective is to approximate the response $Y(\theta)$ as closely as possible in weighted minimax sense, that is with the error function achieving the maximum number of extrema, of amplitude k $\varepsilon(\theta)$, for some specified function $\varepsilon(\theta)$ and as small a value of k as possible, we can define

$$\delta_m = k\ \varepsilon(\theta_m) \qquad m = 1, 2..M \qquad (4.87)$$

for some trial value of k, and solve the linear program. If then q = 0, we can decrease k; if q > 0 we can increase k, and eventually converge to the optimum solution. Rabiner et. al. suggest a "binary search" such that the new value of k is taken

as the geometric mean of the previous k and a known upper or lower limit. Initial limits are chosen conservatively, and modified according to the result of each trial.

The results of the procedure are the optimal values of the c_i and d_i coefficients in (4.81). The numerator and denominator polynomials are then independently factorized. The roots so obtained include all the poles and zeros of the desired filter but also their inverses (reflections in the unit circle $|z| = 1$), and only half the number must be retained. The poles retained must be those inside the unit circle, for stability. The choice for zeros is arbitrary, but choosing as for the poles yields the minimum phase filter. (If zeros are paired *on* the unit circle, one of each pair is retained). The final stage of the design would be to pair the retained poles and zeros to give the coefficients of a cascade-form realization.

A formulation A (multiconstraint) approach involving nonlinear optimization has been taken by Athanassopoulos and Kaiser (1970). Their method is capable of working with any form of digital filter (e.g. cascade, or FIR) and any type of frequency response (e.g. magnitude or group delay). An error function is defined as

$$f(\mathbf{X},\theta) = B(\mathbf{X},\theta) - Y(\theta) \qquad (4.88)$$

or, more precisely, is defined at discrete frequencies as

$$f_m(\mathbf{X}) = B_m(\mathbf{X}) - Y_m \qquad m = 1, 2..M. \qquad (4.89)$$

Each $f_m(\mathbf{X})$ is required to be a differentiable function

of each x_i, which would not be true for example at any frequency for which the achieved magnitude response were zero. Thus if the filter is expected to have zeros on the unit circle it may be better to work with the magnitude squared function, which is everywhere differentiable.

The error samples are required to lie between certain limits, that is,

$$- L_m \leqslant f_m \leqslant U_m \qquad m = 1, 2...M \tag{4.90}$$

where L_m and U_m are positive quantities. As in the linear programming approach previously discussed, we can formulate a corresponding *relaxed* set of constraints

$$\begin{aligned} k\, U_m - f_m &\geqslant 0 \\ k\, L_m + f_m &\geqslant 0 \end{aligned} \qquad m = 1, 2...M. \tag{4.91}$$

which will always be satisfied for a sufficiently large positive value of k. If we can find a filter satisfying (4.91) with $0 < k \leqslant 1$, then the original specification is satisfied.

The method of Athanassopoulos and Kaiser is an interior-point penalty function approach and so uses an unconstrained minimization algorithm repeatedly. The objective function is defined as

$$F(\mathbf{x},k) = k + \sum_{m=1}^{M} \left(\frac{r}{k\, U_m - f_m(\mathbf{x})} + \frac{r}{k\, L_m - f_m(\mathbf{x})} \right) \tag{4.92}$$

where the initial value of $\mathbf{x}$ represents as good a guess as possible, and k is selected large enough to satisfy (4.91) with some margin. The penalty parameter r is chosen so that the largest term within the sum is of the same size as k. Any suitable unconstrained technique is then applied to find minimizing values of $\mathbf{x}$ and k. As with any interior point method the steps taken during the unconstrained minimization must be checked to ensure that the barriers are not jumped, i.e. the denominators in (4.92) must remain positive.

If the value of k so found is less than 1 the process may be stopped. However, if it is not, r is then reduced by some reasonable factor (say 3) and the minimization repeated. As this process is continued the minimizing value of k eventually changes little from iteration to interation and the corresponding filter represents at least a local optimum (which may or may not satisfy the specification (4.90), depending on whether $k \overset{?}{\leqslant} 1$).

This method has the advantages over the linear approach of being able to design (say) a cascade-form filter directly and of having less stringent requirements on arithmetic precision, but it shares with most nonlinear programming problems a generally non-convex objective function, and consequently there is no guarantee that the solution found is the global optimum.

4.6.3 "Formulation B" Approaches

Many papers have been published referring to uses of "formulation B", in conjunction with an unconstrained optimization algorithm to design IIR digital filters in the frequency domain.

The method is conceptually simple, and has been described in section 4.4.2. It can be applied to any form of filter and any type of response. A solution, representing at least a local optimum in the space of designable parameters, is usually obtained with one application of an unconstrained minimization algorithm(c.f. several applications for the foregoing penalty-function method). The problem is not truly unconstrained because for stability all poles must lie inside the unit circle. However, all local optima are guaranteed to be interior points of the feasible region (section 2.3.2), and all constraints are eventually ignored (when the approximation is so good that succeeding iterates remain feasible). On early iterations it may be sufficient to test the result of each predicted step for stability and reduce the step length if necessary. If the phase (or group delay) response is not of interest, infeasible iterates may even be *accepted*, because any poles having $|z| > 1$ may subsequently be reflected inside the unit circle with no change to the magnitude response (other than an irrelevant constant factor.) These and other matters are dealt with more fully in section 4.8.

Steiglitz (1970) pioneered the technique, designing low-order cascade-form filters to fit target magnitude functions only. A least-squares criterion and the Davidon-Fletcher-Powell (DFP) algorithm were used (Fletcher and Powell, 1963). In a paper (1972), and more fully in his thesis (1973), Deczky extended the application to the design of all-pass group delay equalizers, filters with specified group delay, and those with simultaneous specifications on both magnitude and group delay. The DFP algorithm was applied to a least pth objective

function. Deczky's filters were of cascade form, but he chose to use the polar coordinates of poles and zeros as the iteration parameters rather than the coefficients themselves. This has the advantage of simplifying some of the expressions (particularly for the derivatives of group delay) but the disadvantage that the exact numbers of real and complex poles and zeros must be prescribed in advance. Maria and Fahmy (1974) also used the least pth criterion, but iterated the filter coefficients to avoid the a priori division of poles and zeros into real and complex pairs. They used the classical Newton-Raphson iteration, obtaining very rapid convergence and apparently experiencing no (algorithmic) stability problems.

Bandler and Bardakjian (1973) also treated the magnitude response optimization problem, using the least pth error criterion with very large values of p (of the order of thousands). They compared the performance (execution speed) of the DFP algorithm with that of Fletcher's (1970) quasi-Newton method, reporting considerably better performance for the latter. (Chapter five of this thesis reports similar findings for least squares problems.)

Balakrishnan and Rajappan (1974) treated the magnitude response approximation problem in terms of the coefficients of a parallel-form filter, and compared the execution speed of the DFP and Fletcher-Reeves (1964) conjugate gradient algorithms. Better performance was obtained from the former.

King and Condon (1973) also used the DFP algorithm and the cascade realization, with simultaneous specifications of magnitude and phase.

Of the authors cited, Steiglitz, Balakrishnan and Rajappan, King and Condon, and Bandler and Bardakjian allowed unstable

poles to appear during the process, and then reflected them inside the unit circle. For reasons to be discussed in section 4.8 further iterations are then normally required to obtain the (locally) optimal stable solution. Deczky, and Maria and Fahmy, modified their minimization algorithms to ensure that only feasible steps were taken. Another, alternative, approach involving *transformation of variables* was suggested by Jullien and Sid-Ahmed (1974). The stable region of the z plane (i.e. the interior of the unit circle) is mapped to the left half of an auxiliary λ plane via a bilinear transformation

$$z = \frac{1 + \lambda}{1 - \lambda} \tag{4.93}$$

and expressions derived for the frequency response in terms of poles defined by the cartesian coordinates of that plane, λ_x and λ_y. The new stability constraint $\lambda_x < 0$ may be satisfied whilst using an unconstrained minimization algorithm by choosing as variables λ_x^* and λ_y, where $\lambda_x = -\lambda_x^{*2}$. The expressions for magnitude response and its derivatives in terms of λ_x^* and λ_y are shown to be acceptably simple and the authors report rapid convergence using the method.

4.6.4 Formulae for Frequency-Domain Responses and Their Derivatives

In section 4.4.3, general expressions were derived for "formulation B" objective functions and the associated gradient vectors and Gauss-Newton and Hessian matrices. There, the type of response being optimized was left unspecified. In this section we derive the expressions for the following frequency-domain responses (and derivatives), appropriate to a cascade-

form filter - (a) magnitude, (b) magnitude squared, (c) log magnitude, and (d) group delay. Expressions for derivatives of phase response become very complicated and the problem of optimizing phase is better treated via group delay.

The independent variables have been chosen as the filter coefficients themselves.

(a) Magnitude Response

Steiglitz (1970) derives expressions for the magnitude response and its derivatives with respect to the filter coefficients, involving complex arithmetic. Here we derive alternative expressions employing real arithmetic only.

We assume a cascade-form filter having transfer function

$$H(z) = \frac{\prod_{k=1}^{K_N} 1 + a_k z^{-1} + b_k z^{-2}}{\prod_{k=1}^{K_D} 1 + c_k z^{-1} + d_k z^{-2}} \qquad (4.94)$$

$$= \frac{\prod_{k=1}^{K_N} N_k(z)}{\prod_{k=1}^{K_D} D_k(z)} \qquad (4.95)$$

Formally, then, we are considering a cascade of K_N "numerator sections" and K_D "denominator sections" although in practice as many such sections as possible would be paired to form biquadratic sections as shown in figure 4.3, minimizing the storage requirements. The *magnitude squared* response at frequency θ_m is given by

$$M_m = |H(\exp(j\theta_m))|^2 \tag{4.96}$$

$$= \frac{\prod_{k=1}^{K_N} T_{mk}}{\prod_{k=1}^{K_D} U_{mk}} \tag{4.97}$$

where

$$T_{mk} = \mathrm{Re}\{N_k(z)\}^2 + \mathrm{Im}\{N_k(z)\}^2 \Big|_{z=\exp(j\theta_m)} \tag{4.98}$$

$$= (1 + a_k \cos\theta_m + b_k \cos 2\theta_m)^2 + (a_k \sin\theta_m + b_k \sin 2\theta_m)^2 \tag{4.99}$$

$$= 1 + a_k^2 + b_k^2 + 2a_k(1 + b_k)\cos\theta_m + 2b_k \cos 2\theta_m \tag{4.100}$$

and, similarly

$$U_{mk} = 1 + c_k^2 + d_k^2 + 2c_k(1 + d_k)\cos\theta_m + 2d_k \cos 2\theta_m \tag{4.101}$$

The quantity under consideration at present is the magnitude response at the mth frequency, i.e.

$$B_m = (M_m)^{\frac{1}{2}} \tag{4.102}$$

and so its derivative with respect to the ith filter parameter is given by

$$\frac{\partial B_m}{\partial x_i} = \tfrac{1}{2} (M_m)^{-\frac{1}{2}} \frac{\partial M_m}{\partial x_i} \tag{4.103}$$

$$= \frac{1}{2B_m} \frac{\partial M_m}{\partial x_i} \tag{4.104}$$

For example, if $x_i = a_k$, then

$$\frac{\partial M_m}{\partial a_k} = M_m \frac{2a_k + 2\ (1 + b_k) \cos \theta_m}{T_{mk}} \tag{4.105}$$

and so

$$\frac{\partial B_m}{\partial a_k} = \frac{B_m\ (a_k + (1 + b_k) \cos \theta_m)}{T_{mk}} \tag{4.106}$$

Making the following definitions

$$\left.\begin{aligned} S_{Amk} &= a_k + (1 + b_k) \cos \theta_m \\ S_{Bmk} &= b_k + a_k \cos \theta_m + \cos 2\ \theta_m \\ S_{Cmk} &= -\ (c_k + (1 + d_k) \cos \theta_m) \\ S_{Dmk} &= -\ (d_k + c_k \cos \theta_m + \cos 2\ \theta_m) \end{aligned}\right\} \tag{4.107}$$

we have as expressions for the first derivatives:

$$\left.\begin{aligned} \frac{\partial B_m}{\partial a_k} &= \frac{B_m\ S_{Amk}}{T_{mk}} \\ \frac{\partial B_m}{\partial b_k} &= \frac{B_m\ S_{Bmk}}{T_{mk}} \\ \frac{\partial B_m}{\partial c_k} &= \frac{B_m\ S_{Cmk}}{U_{mk}} \\ \frac{\partial B_m}{\partial d_k} &= \frac{B_m\ S_{Dmk}}{U_{mk}} \end{aligned}\right\} \tag{4.108}$$

If second derivatives are required, they are readily found by differentiating expressions (4.108), obtaining the following results:

$$\frac{\partial^2 B_m}{\partial a_k^2} = \frac{B_m}{T_{mk}^2}\left(T_{mk} - S_{Amk}^2\right)$$

$$\frac{\partial^2 B_m}{\partial b_k^2} = \frac{B_m}{T_{mk}^2}\left(T_{mk} - S_{Bmk}^2\right)$$

$$\frac{\partial^2 B_m}{\partial a_k \partial b_k} = \frac{B_m}{T_{mk}^2}\left(T_{mk}\cos\theta_m - S_{Amk}\,S_{Bmk}\right)$$

$$\frac{\partial^2 B_m}{\partial c_k^2} = \frac{B_m}{U_{mk}^2}\left(-U_{mk} + 3\,S_{Cmk}^2\right)$$

$$\frac{\partial^2 B_m}{\partial d_k^2} = \frac{B_m}{U_{mk}^2}\left(-U_{mk} + 3\,S_{Dmk}^2\right)$$

$$\frac{\partial^2 B_m}{\partial c_k \partial d_k} = \frac{B_m}{U_{mk}^2}\left(-U_{mk}\cos\theta_m + 3\,S_{Cmk}\,S_{Dmk}\right)$$

$$\frac{\partial^2 B_m}{\partial a_j \partial a_k} = \frac{B_m\,S_{Amj}\,S_{Amk}}{T_{mj}\,T_{mk}} \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial a_j \partial b_k} = \frac{B_m\,S_{Amj}\,S_{Bmk}}{T_{mj}\,T_{mk}} \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial a_j \partial c_k} = \frac{B_m\,S_{Amj}\,S_{Cmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k$$

$$\frac{\partial^2 B_m}{\partial a_j \partial d_k} = \frac{B_m\,S_{Amj}\,S_{Dmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k$$

$$\frac{\partial^2 B_m}{\partial b_j \partial b_k} = \frac{B_m\,S_{Bmj}\,S_{Bmk}}{T_{mj}\,T_{mk}} \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial b_j \partial c_k} = \frac{B_m\,S_{Bmj}\,S_{Cmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k$$

$$\frac{\partial^2 B_m}{\partial b_j \partial d_k} = \frac{B_m\,S_{Bmj}\,S_{Dmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k$$

$$\frac{\partial^2 B_m}{\partial c_j \partial c_k} = \frac{B_m\,S_{Cmj}\,S_{Cmk}}{U_{mj}\,U_{mk}} \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial c_j \partial d_k} = \frac{B_m\,S_{Cmj}\,S_{Dmk}}{U_{mj}\,U_{mk}} \qquad j \neq k$$

(4.109)

$$\frac{\partial^2 B_m}{\partial d_j \partial d_k} = \frac{B_m \, S_{Dmj} \, S_{Dmk}}{U_{mj} \, U_{mk}} \qquad j \neq k \tag{4.109}$$

(b) Magnitude Squared Response

In this case, we have $B_m = M_m$ of the previous section (only B_m changes its definition - all other symbols are used in their previous context). Formulae for first derivatives are obtained as by-products of the preceding analysis, and those for second derivatives follow with little extra effort. In summary, we have

$$B_m = M_m \tag{4.110}$$

$$\begin{aligned}
\frac{\partial B_m}{\partial a_k} &= \frac{2 \, M_m \, S_{Amk}}{T_{mk}} \\
\frac{\partial B_m}{\partial b_k} &= \frac{2 \, M_m \, S_{Bmk}}{T_{mk}} \\
\frac{\partial B_m}{\partial c_k} &= \frac{2 \, M_m \, S_{Cmk}}{U_{mk}} \\
\frac{\partial B_m}{\partial d_k} &= \frac{2 \, M_m \, S_{Dmk}}{U_{mk}}
\end{aligned} \tag{4.111}$$

$$\begin{aligned}
\frac{\partial^2 B_m}{\partial a_k^2} &= \frac{2 \, M_m}{T_{mk}} \\
\frac{\partial^2 B_m}{\partial b_k^2} &= \frac{2 \, M_m}{T_{mk}} \\
\frac{\partial^2 B_m}{\partial a_k \partial b_k} &= \frac{2 \, M_m \cos \theta_m}{T_{mk}} \\
\frac{\partial^2 B_m}{\partial c_k^2} &= \frac{2 \, M_m}{U_{mk}} \left(\frac{4 \, S_{Cmk}^2}{U_{mk}} - 1 \right)
\end{aligned} \tag{4.112}$$

$$
\begin{aligned}
&\frac{\partial^2 B_m}{\partial d_k^2} = \frac{2\,M_m}{U_{mk}}\left(\frac{4\,S_{Dmk}^2}{U_{mk}} - 1\right) \\
&\frac{\partial^2 B_m}{\partial c_k \partial d_k} = \frac{2\,M_m}{U_{mk}}\left(\frac{4\,S_{Cmk}\,S_{Dmk}}{U_{mk}} - \cos\theta_m\right) \\
&\frac{\partial^2 B_m}{\partial a_j \partial a_k} = \frac{4\,M_m\,S_{Amj}\,S_{Amk}}{T_{mj}\,T_{mk}} \qquad j \neq k \\
&\frac{\partial^2 B_m}{\partial a_j \partial b_k} = \frac{4\,M_m\,S_{Amj}\,S_{Bmk}}{T_{mj}\,T_{mk}} \qquad j \neq k \\
&\frac{\partial^2 B_m}{\partial a_j \partial c_k} = \frac{4\,M_m\,S_{Amj}\,S_{Cmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k \\
&\frac{\partial^2 B_m}{\partial a_j \partial d_k} = \frac{4\,M_m\,S_{Amj}\,S_{Dmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k \\
&\frac{\partial^2 B_m}{\partial b_j \partial b_k} = \frac{4\,M_m\,S_{Bmj}\,S_{Bmk}}{T_{mj}\,T_{mk}} \qquad j \neq k \\
&\frac{\partial^2 B_m}{\partial b_j \partial c_k} = \frac{4\,M_m\,S_{Bmj}\,S_{Cmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k \\
&\frac{\partial^2 B_m}{\partial b_j \partial d_k} = \frac{4\,M_m\,S_{Bmj}\,S_{Dmk}}{T_{mj}\,U_{mk}} \qquad \text{all } j,k \\
&\frac{\partial^2 B_m}{\partial c_j \partial c_k} = \frac{4\,M_m\,S_{Cmj}\,S_{Cmk}}{U_{mj}\,U_{mk}} \qquad j \neq k \\
&\frac{\partial^2 B_m}{\partial c_j \partial d_k} = \frac{4\,M_m\,S_{Cmj}\,S_{Dmk}}{U_{mj}\,U_{mk}} \qquad j \neq k \\
&\frac{\partial^2 B_m}{\partial d_j \partial d_k} = \frac{4\,M_m\,S_{Dmj}\,S_{Dmk}}{U_{mj}\,U_{mk}} \qquad j \neq k
\end{aligned}
\tag{4.112}
$$

The importance of the magnitude squared function is that it is differentiable (and twice differentiable) at frequencies where it is zero. This does not apply to the magnitude (and log magnitude) functions. Sharp cut-off digital filters are usually designed with zeros *on* the unit circle. In a design

by optimization this is ensured by fixing $b_k = 1$ in some numerator sections, and varying only the a_k according to the algorithm. In such cases, zeros often move so close to the frequencies of specification (θ_m) that T_{mk} becomes nearly zero, and the evaluation of derivatives of the magnitude response is unreliable or impossible. The magnitude squared response is still usable, although not all of equations (4.111) and (4.112) can be applied. Excessively small values of T_{mk} must be trapped by the computer algorithm and derivatives evaluated after factoring T_{mk} out of the expression for M_m.

(c) Log Magnitude Response

In some cases it may be more natural to fit a model to a desired or a measured spectrum represented on a logarithmic magnitude scale than on a linear scale. An example would be a model for a speech sound, because the ear appears to have a roughly logarithmic response. Mathematically, this has the desirable effect of greatly simplifying the second-derivative matrix. We choose to define

$$B_m = \log_e (M_m) \tag{4.113}$$

because the resulting expressions are simplest. If the target response were specified in some other logarithmic units (e.g. decibels), scaling by some constant factor would be required.

From (4.97) and (4.113)

$$B_m = \sum_{k=1}^{K_N} \log_e (T_{mk}) - \sum_{k=1}^{K_D} \log_e (U_{mk}) \tag{4.114}$$

Therefore

$$\left.\begin{aligned} \frac{\partial B_m}{\partial a_k} &= \frac{\frac{\partial}{\partial a_k}(T_{mk})}{T_{mk}} = \frac{2\,S_{Amk}}{T_{mk}} \\ \text{and similarly} \\ \frac{\partial B_m}{\partial b_k} &= \frac{2\,S_{Bmk}}{T_{mk}} \\ \frac{\partial B_m}{\partial c_k} &= \frac{2\,S_{Cmk}}{U_{mk}} \\ \frac{\partial B_m}{\partial d_k} &= \frac{2\,S_{Dmk}}{U_{mk}} \end{aligned}\right\} \quad (4.115)$$

Because the first derivatives are functions of the coefficients of *one section only*, most of the second derivatives are zero. The Hessian matrix **H** is thus equal to the Gauss-Newton matrix **R** except for the elements on the main diagonal and some of the elements on the immediately adjacent diagonals (assuming that the parameters are ordered such that b_k follows a_k and d_k follows c_k, for all k). The additional labour necessary to compute **H** (involving of the order of N operations) is small compared with that already expended in obtaining **R** (of order N^2 operations), so that there may be a better case for adopting a Newton method when optimizing the log magnitude response than in the other problems so far considered.

Formal expressions for second derivatives are:

$$\left.\begin{aligned} \frac{\partial^2 B_m}{\partial a_k^2} &= \frac{2\,(T_{mk} - 2\,S_{Amk}^2)}{T_{mk}^2} \\ \frac{\partial^2 B_m}{\partial b_k^2} &= \frac{2\,(T_{mk} - 2\,S_{Bmk}^2)}{T_{mk}^2} \end{aligned}\right\} \quad (4.116)$$

$$\frac{\partial^2 B_m}{\partial a_k \partial b_k} = \frac{2\,(T_{mk}\cos\theta_m - 2\,S_{Amk}\,S_{Bmk})}{T_{mk}^2}$$

$$\frac{\partial^2 B_m}{\partial c_k^2} = \frac{-2\,(U_{mk} - 2\,S_{Cmk}^2)}{U_{mk}^2}$$

$$\frac{\partial^2 B_m}{\partial d_k^2} = \frac{-2\,(U_{mk} - 2\,S_{Dmk}^2)}{U_{mk}^2}$$

$$\frac{\partial^2 B_m}{\partial c_k \partial d_k} = \frac{-2\,(U_{mk}\cos\theta_m - 2\,S_{Cmk}\,S_{Dmk})}{U_{mk}^2} \qquad (4.116)$$

$$\frac{\partial^2 B_m}{\partial a_j \partial a_k} = \frac{\partial^2 B_m}{\partial a_j \partial b_k} = \frac{\partial^2 B_m}{\partial b_j \partial b_k} = 0 \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial c_j \partial c_k} = \frac{\partial^2 B_m}{\partial c_j \partial d_k} = \frac{\partial^2 B_m}{\partial d_j \partial d_k} = 0 \qquad j \neq k$$

$$\frac{\partial^2 B_m}{\partial a_j \partial c_k} = \frac{\partial^2 B_m}{\partial a_j \partial d_k} = \frac{\partial^2 B_m}{\partial b_j \partial c_k} = \frac{\partial^2 B_m}{\partial b_j \partial d_k} = 0 \qquad \text{all } j,k$$

(d) Group Delay Response

The group delay function of a digital filter is defined as

$$\tau(\theta) = -\frac{d}{d\theta}\{\arg(H(\exp j\theta))\} \qquad (4.117)$$

Defining $H(z) = \frac{N(z)}{D(z)}$, Deczky (1969) has shown that

$$\tau(\theta) = \mathrm{Re}\,\frac{z\,\frac{dX(z)}{dz}}{X(z)} \qquad (4.118)$$

where

$$X(z) = N(z^{-1})\,D(z) \qquad (4.119)$$

and

$$z = \exp(j\theta) \quad . \qquad (4.120)$$

For the cascade-form digital filter, we have

$$N(z) = \prod_{k=1}^{K_N} (1 + a_k z^{-1} + b_k z^{-2}) \tag{4.121}$$

and

$$D(z) = \prod_{k=1}^{K_D} (1 + c_k z^{-1} + d_k z^{-2}) \tag{4.122}$$

which lead to

$$X(z) = \prod_{k=1}^{K_N} (1 + a_k z + b_k z^2) \prod_{k=1}^{K_D} (1 + c_k z^{-1} + d_k z^{-2}) \tag{4.123}$$

Considering an arbitrary function which is a product of several factors, for instance

$$F(x) = A(x)\, B(x)\, C(x) \tag{4.124}$$

then

$$\frac{dF}{dx} = AB \frac{dC}{dx} + AC \frac{dB}{dx} + BC \frac{dA}{dx} \tag{4.125}$$

so that

$$\frac{1}{F} \frac{dF}{dx} = \frac{1}{C} \frac{dC}{dx} + \frac{1}{B} \frac{dB}{dx} + \frac{1}{A} \frac{dA}{dx} \tag{4.126}$$

Applying this reasoning to the product in (4.123), we get

$$\frac{z}{X(z)} \frac{dX(z)}{dz} = z \sum_{k=1}^{K_N} \frac{a_k + 2 b_k z}{1 + a_k z + b_k z^2} - z \sum_{k=1}^{K_D} \frac{c_k z^{-2} + 2 d_k z^{-3}}{1 + c_k z^{-1} + d_k z^{-2}} \tag{4.127}$$

$$= \sum_{k=1}^{K_N} \frac{a_k z + 2b_k z^2}{1 + a_k z + b_k z^2} - \sum_{k=1}^{K_D} \frac{c_k z^{-1} + 2d_k z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (4.128)$$

Substituting $z = \cos\theta + j \sin\theta$ and solving for the real part, we get

$$\tau(\theta) = \sum_{k=1}^{K_N} \frac{a_k^2 + 2b_k^2 + a_k (1 + 3b_k) \cos\theta + 2b_k \cos 2\theta}{1 + a_k^2 + b_k^2 + 2a_k (1 + b_k) \cos\theta + 2b_k \cos 2\theta}$$

$$- \sum_{k=1}^{K_D} \frac{c_k^2 + 2d_k^2 + c_k (1 + 3d_k) \cos\theta + 2d_k \cos 2\theta}{1 + c_k^2 + d_k^2 + 2c_k (1 + d_k) \cos\theta + 2d_k \cos 2\theta} \qquad (4.129)$$

Defining

$$V_{mk} = a_k^2 + 2b_k^2 + a_k (1 + 3b_k) \cos\theta_m + 2b_k \cos 2\theta_m \qquad (4.130)$$

and

$$W_{mk} = c_k^2 + 2d_k^2 + c_k (1 + 3d_k) \cos\theta_m + 2d_k \cos 2\theta_m \qquad (4.131)$$

we have then, in terms of T_{mk} and U_{mk} already defined,

$$B_m = \tau(\theta_m) = \sum_{k=1}^{K_N} \frac{V_{mk}}{T_{mk}} - \sum_{k=1}^{K_D} \frac{W_{mk}}{U_{mk}} \qquad (4.132)$$

As in the case of the log magnitude response, the expression for group delay is the sum of several terms, each term involving the coefficients of only one (numerator or denominator) section. Hence, each expression for a first derivative contains the coefficients of one section only, and all second derivatives are zero except those with respect to two coefficients of a single section.

Making the definitions:

$$\left.\begin{aligned} R_{Amk} &= \frac{\partial V_{mk}}{\partial a_k} = 2a_k + (1 + 3b_k)\cos\theta_m \\ R_{Bmk} &= \frac{\partial V_{mk}}{\partial b_k} = 4b_k + 3a_k \cos\theta_m + 2\cos 2\theta_m \\ R_{Cmk} &= \frac{\partial W_{mk}}{\partial c_k} = 2c_k + (1 + 3d_k)\cos\theta_m \\ R_{Dmk} &= \frac{\partial W_{mk}}{\partial d_k} = 4d_k + 3c_k \cos\theta_m + 2\cos 2\theta_m \end{aligned}\right\} \quad (4.133)$$

the expressions for first derivatives become

$$\left.\begin{aligned} \frac{\partial B_m}{\partial a_k} &= \frac{T_{mk}\, R_{Amk} - 2\, V_{mk}\, S_{Amk}}{T_{mk}^2} \\ \frac{\partial B_m}{\partial b_k} &= \frac{T_{mk}\, R_{Bmk} - 2\, V_{mk}\, S_{Bmk}}{T_{mk}^2} \\ \frac{\partial B_m}{\partial c_k} &= -\frac{U_{mk}\, R_{Cmk} + 2\, W_{mk}\, S_{Cmk}}{U_{mk}^2} \\ \frac{\partial B_m}{\partial d_k} &= -\frac{U_{mk}\, R_{Dmk} + 2\, W_{mk}\, S_{Dmk}}{U_{mk}^2} \end{aligned}\right\} \quad (4.134)$$

and those for the nonzero second derivatives

$$\left.\begin{aligned} \frac{\partial^2 B_m}{\partial a_k^2} &= \frac{2}{T_{mk}} - \frac{2}{T_{mk}^2}(V_{mk} + 2R_{Amk}\, S_{Amk}) + \frac{8}{T_{mk}^3} V_{mk}\, S_{Amk}^2 \\ \frac{\partial^2 B_m}{\partial b_k^2} &= \frac{4}{T_{mk}} - \frac{2}{T_{mk}^2}(V_{mk} + 2R_{Bmk}\, S_{Bmk}) + \frac{8}{T_{mk}^3} V_{mk}\, S_{Bmk}^2 \end{aligned}\right\} \quad (4.135)$$

$$\frac{\partial^2 B_m}{\partial a_k \partial b_k} = \frac{3 \cos \theta_m}{T_{mk}} - \frac{2}{T_{mk}^2}\left(V_{mk} \cos \theta_m + R_{Amk}\, S_{Bmk} + R_{Bmk}\, S_{Amk}\right)$$

$$+ \frac{8}{T_{mk}^3} V_{mk}\, S_{Amk}\, S_{Bmk}$$

$$\frac{\partial^2 B_m}{\partial c_k^2} = \frac{-2}{U_{mk}} + \frac{2}{U_{mk}^2}\left(W_{mk} - 2R_{Cmk}\, S_{Cmk}\right) - \frac{8}{U_{mk}^3} W_{mk}\, S_{Cmk}^2$$

$$\frac{\partial^2 B_m}{\partial d_k^2} = \frac{-4}{U_{mk}} + \frac{2}{U_{mk}^2}\left(W_{mk} - 2R_{Dmk}\, S_{Dmk}\right) - \frac{8}{U_{mk}^3} W_{mk}\, S_{Dmk}^2$$

$$\frac{\partial^2 B_m}{\partial c_k \partial d_k} = \frac{-3 \cos \theta_m}{U_{mk}} + \frac{2}{U_{mk}^2}\left(W_{mk} \cos \theta_m - R_{Cmk}\, S_{Dmk} - R_{Dmk}\, S_{Cmk}\right)$$

$$- \frac{8}{U_{mk}^3} W_{mk}\, S_{Cmk}\, S_{Dmk} \qquad (4.135)$$

(e) All-Pass Group Delay Equalizers

An approach often taken (for example, by Deczky (1973)) to the simultaneous approximation of magnitude and group delay response involves the use of the so-called *all-pass* filter. A digital filter is designed by any suitable method to achieve the specified magnitude response, without at this stage paying attention to group delay. This filter is then cascaded with another, an all-pass, whose parameters are chosen to optimize the group delay response of the overall cascade filter without having any effect on the magnitude.

The resulting filter may be sub-optimal with respect to an overall specification on both magnitude and group delay, but the splitting of the design process into two stages affords a great simplification. In addition, if the importance of approximating the prescribed magnitude response versus that of

group delay can be appropriately quantified (weighted), the all-pass design affords a good starting point for an overall parameter optimization.

The most common objective as regards group delay is that it be constant within the passband (s), and an all-pass filter designed with this aim is called an *equalizer*.

All-pass filters may be realized in many ways, including the cascade of second-order sections shown in figure 4.3. In this case the special all-pass property is ensured by fixed relations between the numerator and denominator coefficients. We choose here to describe such a filter by the equation

$$H(z) = \prod_{k=1}^{K} \frac{1 + \frac{c_k}{d_k} z^{-1} + \frac{1}{d_k} z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (4.136)$$

That it has no frequency-dependent effect on the magnitude response is readily seen by considering just the kth section, and its magnitude squared response $M(\theta)$, i.e.

$$M(\theta) = \left[H(z)\, H(z^{-1}) \right]_{z=\exp(j\theta)}$$

$$= \left[\frac{1}{d_k z^2} \left(\frac{1 + c_k z^{+1} + d_k z^{+2}}{1 + c_k z^{-1} + d_k z^{-2}} \right) \cdot \frac{1}{d_k z^{-2}} \left(\frac{1 + c_k z^{-1} + d_k z^{-2}}{1 + c_k z + d_k z^2} \right) \right]_{z=\exp(j\theta)}$$

$$= \frac{1}{d_k z^2} \qquad (4.137)$$

The group delay response follows by substituting $K_N = K_D = K$ and $a_k = c_k/d_k$ and $b_k = 1/d_k$ into (4.129). After lengthy algebraic reduction, we get

$$\tau(\theta) = \sum_{k=1}^{K} \frac{2\,(1 - d_k^2 + c_k\,(1 - d_k)\cos\theta)}{1 + c_k^2 + d_k^2 + 2c_k(1 + d_k)\cos\theta + 2d_k\cos 2\theta} \tag{4.138}$$

As with any digital filter, the two poles p_1, p_2 of a second-order all-pass section (which are either both real or a complex conjugate pair) must lie inside the unit circle, for stability. The two zeros appear at $\frac{1}{p_1}$ and $\frac{1}{p_2}$ and so are *outside* the unit circle. In group delay optimization, stability checks are necessary during the process because there is no opportunity later to reflect unstable poles inside the unit circle as may be done in magnitude optimization. The above characterization of the all-pass in terms of its denominator coefficients c_k, d_k (rather than its numerator coefficients) simplifies a computer program because the stability check is the same as with "ordinary" sections.

Formulae for derivatives of group delay response with respect to the designable parameters, c_k, d_k are obtained by differentiating (4.138). Again, because $B_m = \tau(\theta_m)$ is the sum of contributors from individual sections, the intersection second derivatives will be zero. Making the definitions

$$X_{mk} = 2\,(1 - d_k^2 + c_k(1 - d_k)\cos\theta_m) \tag{4.139}$$

and

$$\left.\begin{aligned} Q_{Cmk} &= \frac{\partial X_{mk}}{\partial c_k} = 2\,(1 - d_k)\cos\theta_m \\ Q_{Dmk} &= \frac{\partial X_{mk}}{\partial d_k} = -\,4d_k - 2c_k\cos\theta_m \end{aligned}\right\} \tag{4.140}$$

and employing also symbols defined in (4.100), (4.101) and (4.107), the expressions for the mth spot group delay response

and its nonvanishing first and second derivatives become:

$$B_m = \tau(\theta_m) = \sum_{k=1}^{K} \frac{X_{mk}}{U_{mk}} \qquad (4.141)$$

$$\left.\begin{aligned} \frac{\partial B_m}{\partial c_k} &= \frac{U_{mk}\, Q_{Cmk} + 2X_{mk}\, S_{Cmk}}{U_{mk}^2} \\ \frac{\partial B_m}{\partial d_k} &= \frac{U_{mk}\, Q_{Dmk} + 2X_{mk}\, S_{Dmk}}{U_{mk}^2} \end{aligned}\right\} \qquad (4.142)$$

and

$$\left.\begin{aligned} \frac{\partial^2 B_m}{\partial c_k^2} &= \frac{2}{U_{mk}^2}\,(2Q_{Cmk}\, S_{Cmk} - X_{mk}) + \frac{8}{U_{mk}^3}\, X_{mk}\, S_{Cmk}^2 \\ \frac{\partial^2 B_m}{\partial c_k \partial d_k} &= \frac{-2\cos\theta_m}{U_{mk}} + \frac{2}{U_{mk}^2}\,(Q_{Cmk}\, S_{Dmk} + Q_{Dmk}\, S_{Cmk} - X_{mk}\cos\theta_m) \\ &\quad + \frac{8}{U_{mk}^3}\, X_{mk}\, S_{Cmk}\, S_{Dmk} \\ \frac{\partial^2 B_m}{\partial d_k^2} &= \frac{-4}{U_{mk}} + \frac{2}{U_{mk}^2}\,(2\, Q_{Dmk}\, S_{Dmk} - X_{mk}) + \frac{8}{U_{mk}^3}\, X_{mk}\, S_{Dmk}^2 \end{aligned}\right\} \qquad (4.143)$$

4.7 Design of Infinite Impulse Response Digital Filters in the Time Domain

4.7.1 Review of the Literature

The problem considered in this section has, over the last 15 years, appeared in several different guises. Steiglitz and McBride (1965), Schulz (1968), and Miller (1973), among others,

consider the problem of identifying the parameters of a model for some physical plant whose behaviour is to be controlled. It is assumed that (discrete-time samples of) input and output signals are available over a certain period of time. The problem is to find the parameters of a model which produce the "best" approximation to the given output sequence when fed with the given input sequence.

Other authors, such as Brophy and Salazar (1974) and Cadzow (1976) consider synthesizing an IIR digital filter to approximate a given impulse response. (This is clearly a less general form of the system identification problem, where the input sequence is just an impulse at t = 0). They point out that even if the original specification were on magnitude and phase in the *frequency* domain, a suitable target impulse response sequence could be produced from this by one of the standard methods for FIR filter design. Cadzow shows that as the length of the target sequence becomes large, least squares criteria in time and frequency domains become indistinguisable in terms of the models produced. Brophy and Salazar discuss several reasons for regarding the time domain as "the more natural domain" for IIR filter synthesis. Firstly, the computation of the time response and its derivatives requires nothing more than additions and multiplications (that is, digital filtering operations) whereas frequency domain responses are complicated functions of the coefficients taking much computer time to evaluate (for example, the phase function requires the arctangent). Secondly, good initial approximations are available for the coefficients when working in the time domain, using procedures such as those of Shanks (1967) and

Kalman (1958) which involve only the solution of linear equations. The fact that magnitude and phase responses are optimized inseparably, and over the entire range of frequency, is a mixed blessing. The programming is greatly simplified thereby, but stringent specifications may fail to be met at certain critical frequencies. However, as Brophy and Salazar point out, the result of time domain optimization provides an excellent starting point for a frequency-domain optimization to obtain a final fit.

Steiglitz (1977) generates a pole-zero model for a speech signal by fitting an IIR filter model to a minimum-phase vocal tract impulse response estimate obtained by homomorphic processing. (A similar problem is dealt with in chapter seven of this thesis).

In the work of all the authors cited so far in this section, the direct form of the filter is derived. The potential of optimization methods to derive a cascade (or parallel) filter and so avoid the numerical precision problems associated with the direct form has not as yet been realized. Steiglitz and McBride (1965), Schulz (1968), Miller (1973), Evans and Fischl (1973) and Steiglitz (1977) use special iterations (not general minimization methods) which are applicable only to the direct form. Brophy and Salazar (1974) try steepest-descent and DFP methods, concluding that a few iterations of steepest descent usually provides a useful improvement from a good starting point, and that the extra complexity of DFP is not worthwhile. However, Cadzow (1976) found the usual disappointing convergence of the steepest descent method, and highly recommends the damped Gauss-Newton or "linearization" algorithm. His

test of a second-derivative (Newton) method was unsuccessful, but considering that no provision was made to handle an indefinite Hessian this is not surprising.

In another paper, Bertran (1975) also obtains good results from the damped Gauss-Newton method, again for a direct-form filter.

The reason for the concentration on the direct form is presumably that good initial approximations in this form are available via the methods of Shanks and Kalman, or by Padé approximant techniques (Burrus and Parks (1970), Hastings-James and Mehra (1977)). However, if the positions of the poles and zeros are to be determined, a factorization of a high-order polynomial becomes necessary anyway, and this may as well be performed on the initial approximation (before the optimization) as afterwards.

Jackson and Wood (1978) point the way towards the direct formulation of an IIR model in cascade form. Their work treats the *linear prediction* problem, that is, the filter generated by the procedure is an FIR filter whose parameters are such as to minimize the energy in the output sequence. They depart from the usual practice of employing a direct form filter, and are thus forced to use optimization techniques to determine the parameters. In the case of linear prediction this is a great sacrifice because there are very efficient non-iterative solution methods available when using the direct form filter. However, for IIR modelling, iterative methods are needed anyhow. The chief value of the Jackson and Wood paper in this context is that auxiliary filters for generating gradient sequences were derived, and these generalize

nicely to the case where the derived filter has poles as well as zeros. This matter is taken up in the next section.

All the foregoing work has used a least-squares criterion in the time domain, that is, a function of the form

$$F(\mathbf{x}) = \sum_{m=1}^{M} w_m (B_m(\mathbf{x}) - Y_m)^2 \qquad (4.144)$$

is miminized, where B_m and Y_m are the achieved and target output samples at the mth time instant. This is an example of "formulation B1" of section 4.4.3. In principle it would be possible instead to adopt formulation A - that is, force each $|B_m' - Y_m|$ to satisfy a constraint. A penalty-function approach (similar to that outlined in section 4.6.2) was proposed for *continuous* time domain approximation by Athanassopoulos, Schoeffler and Waren (1966), but apparently this has not been used for *digital* filters in the time domain. The formulation B approach would seem simpler, and more meaningful in most practical cases.

4.7.2 Computational Schemes for Gradient Sequences

As was pointed out in section 4.4.3, an appropriate formulation for time domain optimization is "formulation B1". The appropriate expressions for the objective function and its derivatives are (4.17), (4.18), (4.57), (4.58), (4.59), (4.60) and (4.44). In the time domain, the elemental achieved responses $B_m(\mathbf{x})$ are the output samples themselves, when the filter with parameters $\mathbf{x}$ is driven with some specified sequence (which in many cases of practical interest would be just an impulse at t = 0). As (4.44) shows, we need the first

derivatives $\frac{\partial B_m(\mathbf{X})}{\partial x_i}$ for each parameter x_i and each time index m, and to use a Newton method we require also the second derivatives $\frac{\partial^2 B_m(\mathbf{X})}{\partial x_i \partial x_j}$. It turns out that all such quantities can be generated very conveniently as the output sequences of digital filterswhich are very closely related to the filter under consideration. Those sum-of-products expressions in (4.44) involving derivatives are then interpreted as zero-lag cross-correlations between the appropriate (weighted) sequences.

To show this we consider a "primary" digital filter, characterised by a particular topology and a vector of parameters $\mathbf{X}$, exhibiting an impulse reponse $h_m(\mathbf{X})$ for $m = 0, 1, \ldots$. We allow the filter to be driven with a certain sequence $u_m(m = 0, 1, \ldots)$ and obtain an output sequence v_m $(m = 0, 1, \ldots)$, which is of course a function of $\mathbf{X}$. If $U(z)$, $V(z, \mathbf{X})$ and $H(z, \mathbf{X})$ are the z transforms of the u, v and h sequences, then

$$V(z, \mathbf{X}) = H(z, \mathbf{X})\, U(z) \tag{4.145}$$

and the output samples themselves can in principle be found from the z-transform inversion formula

$$v_m(\mathbf{X}) = \frac{1}{2\pi j} \oint z^{m-1} H(z, \mathbf{X})\, U(z)\, dz \tag{4.146}$$

where the contour integral is evaluated over any closed path enclosing all the poles of $V(z, \mathbf{X})$. The unit circle suffices for any stable v sequence and this condition will be true for any useful filter. The required (first or second) derivatives of $v_m(\mathbf{X})$ may be found by differentiating (4.146),

and the differentiation and contour integration operations may be interchanged since they are with respect to totally different variables. Thus we have

$$\frac{\partial v_m(x)}{\partial x_i} = \frac{1}{2\pi j} \oint z^{m-1} \frac{\partial H(z,x)}{\partial x_i} U(z)\, dz \,. \qquad (4.147)$$

This is of the same form as (4.146), and shows that derivatives of the output time samples of the primary filter with respect to any parameter x_i are obtained simply as the output sequence of a filter with transfer function $\frac{\partial H(z,x)}{\partial x_i}$, driven with the same input sequence. Second derivatives are exactly analogous.

Consider in particular a cascade-form filter with K_N second-order numerator sections and K_D second-order denominator sections, having transfer function

$$H(z) = \frac{A_o \prod_{k=1}^{K_N} (1 + a_k z^{-1} + b_k z^{-2})}{\prod_{k=1}^{K_D} (1 + c_k z^{-1} + d_k z^{-2})} \qquad (4.148)$$

Then, by differentiation, we obtain the transfer functions of these gradient filters, for example

$$\frac{\partial H(z)}{\partial a_i} = \frac{A_o z^{-1} \prod_{\substack{k=1 \\ k \neq i}}^{K_N} (1 + a_k z^{-1} + b_k z^{-2})}{\prod_{k=1}^{K_D} (1 + c_k z^{-1} + d_k z^{-2})} \qquad (4.149)$$

so that the gradient sequence with respect to the ith a coefficient is obtained by passing the input sequence through

all but the ith numerator section, and then *delaying the sequence by one sample.* For $\frac{\partial H(z)}{\partial b_i}$ the expression is the same except that z^{-2} replaces z^{-1}, so that the $\frac{\partial v_m}{\partial b_i}$ sequence is the same as the $\frac{\partial v_m}{\partial a_i}$ sequence *but delayed by one more sample.*

Since the primary and gradient filters have so much in common, it is useful (especially for a large number of numerator sections) to avoid the duplication. By multiplying the top and bottom of the right-hand side of (4.149) by $1 + a_i z^{-1} + b_i z^{-2}$, we get

$$\frac{\partial H(z)}{\partial a_i} = H(z)\ z^{-1}\ \frac{1}{1 + a_i z^{-1} + b_i z^{-2}} \qquad (4.150)$$

so that all gradient components with respect to numerator parameters (a's and b's) can be found by passing the *output* of the primary filter through a parallel set of recursive sections in the manner shown in figure 4.13. Each recursive section effectively cancels one pair of zeros of H(z) and at the same time provides the required delay. Unfortunately the usefulness of this scheme would seem limited to transfer functions H(z) which are minimum phase, because otherwise at least one of the recursive sections is unstable. Although the unstable poles are theoretically cancelled perfectly by the zeros in H(z), there would probably be serious error buildup in the gradient sequences due to the use of finite precision arithmetic.

Differentiating (4.148) with respect to c_i and d_i yields

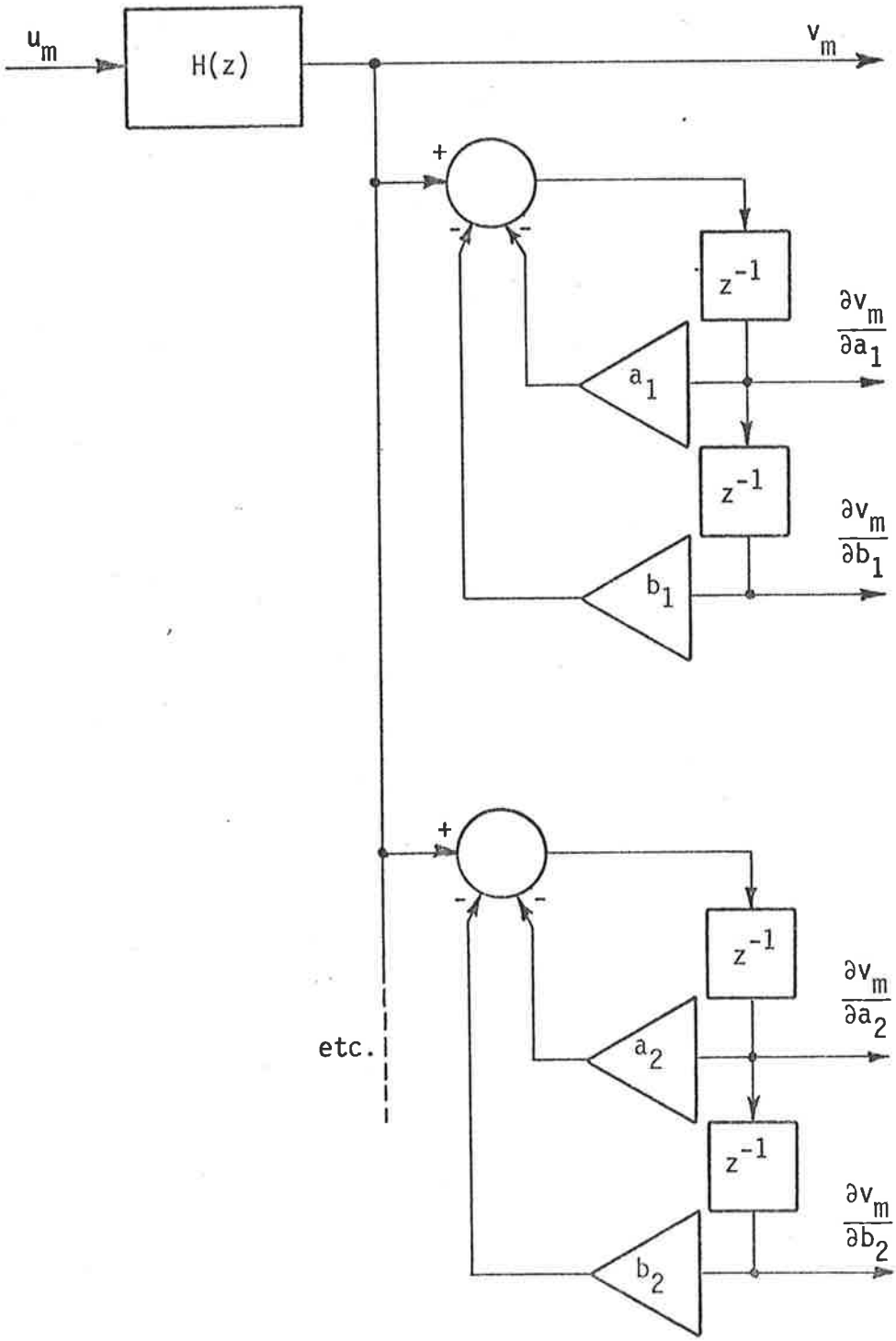


FIGURE 4.13 Recursive Gradient Computers for Numerator Parameters

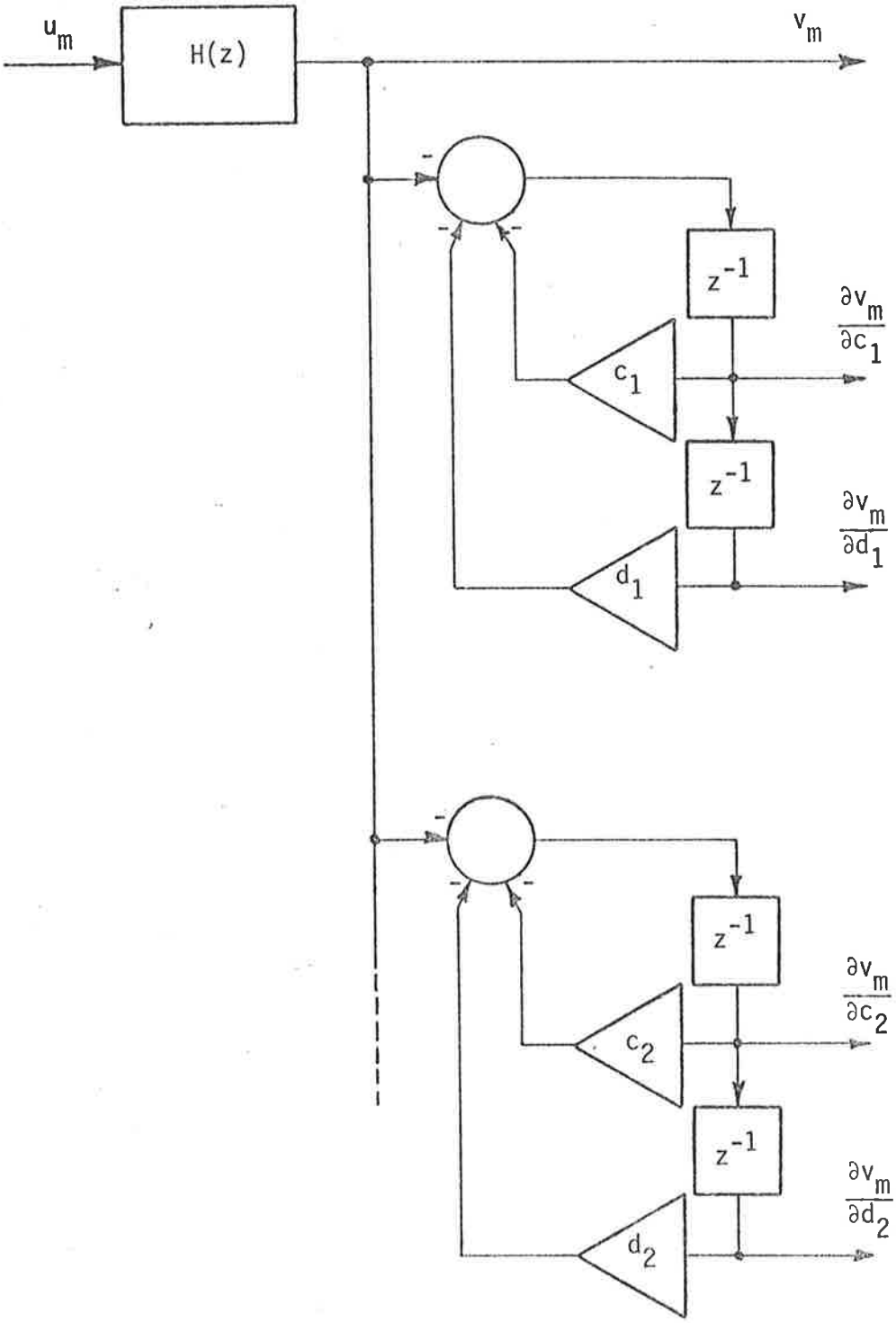


FIGURE 4.14 Recursive Gradient Computers for Denominator Parameters

$$\frac{\partial H(z)}{\partial c_i} = - H(z)\, z^{-1} \frac{1}{1 + c_i z^{-1} + d_i z^{-2}} \tag{4.151}$$

and

$$\frac{\partial H(z)}{\partial d_i} = - H(z)\, z^{-2} \frac{1}{1 + c_i z^{-1} + d_i z^{-2}} \tag{4.152}$$

so that these gradient sequences, too, are generated by simple recursive sections following the primary filter as in figure 4.14. The difference from the scheme of figure 4.13 is that the output of the primary filter is negated, and that the added sections do not cancel zeros of H(z) but, rather, repeat its poles.

Second-derivative sequences are formally derived by differentiating expressions like (4.149) once more. If all zeros are minimum-phase, they can be efficiently computed by post-filtering the first-derivative sequences with yet another second-order section. The operations involved are summarized below, where we use the shorthand expressions $H(\cdot)$, $R_{Nk}(\cdot)$ and $R_{Dk}(\cdot)$ to indicate filtering by $H(z)$, $\frac{1}{1 + a_k z^{-1} + b_k z^{-2}}$ and $\frac{1}{1 + c_k z^{-1} + d_k z^{-2}}$ respectively:

$$\left.\begin{aligned} v_m &= H(u_m) \\ \frac{\partial v_m}{\partial a_k} &= z^{-1} R_{Nk}(v_m) \\ \frac{\partial v_m}{\partial b_k} &= z^{-2} R_{Nk}(v_m) \\ \frac{\partial v_m}{\partial c_k} &= - z^{-1} R_{Dk}(v_m) \end{aligned}\right\} \tag{4.153}$$

$$\frac{\partial v_m}{\partial d_k} = - z^{-2} R_{Dk} (v_m)$$

$$\frac{\partial^2 v_m}{\partial a_k^2} = \frac{\partial^2 v_m}{\partial a_k \partial b_k} = \frac{\partial^2 v_m}{\partial b_k^2} = 0$$

$$\frac{\partial^2 v_m}{\partial a_j \partial a_k} = z^{-1} R_{Nj} \left(\frac{\partial v_m}{\partial a_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial a_j \partial b_k} = z^{-2} R_{Nj} \left(\frac{\partial v_m}{\partial a_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial b_j \partial b_k} = z^{-3} R_{Nj} \left(\frac{\partial v_m}{\partial a_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial c_k^2} = -2 z^{-1} R_{Dk} \left(\frac{\partial v_m}{\partial c_k}\right)$$

$$\frac{\partial^2 v_m}{\partial c_k \partial d_k} = - 2 z^{-2} R_{Dk} \left(\frac{\partial v_m}{\partial c_k}\right) \qquad (4.153)$$

$$\frac{\partial^2 v_m}{\partial d_k^2} = - 2 z^{-3} R_{Dk} \left(\frac{\partial v_m}{\partial c_k}\right)$$

$$\frac{\partial^2 v_m}{\partial c_j \partial c_k} = - z^{-1} R_{Dj} \left(\frac{\partial v_m}{\partial c_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial c_j \partial d_k} = - z^{-2} R_{Dj} \left(\frac{\partial v_m}{\partial c_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial d_j \partial d_k} = - z^{-3} R_{Dj} \left(\frac{\partial v_m}{\partial c_k}\right) \qquad j \neq k$$

$$\frac{\partial^2 v_m}{\partial a_j \partial c_k} = - z^{-1} R_{Dj} \left(\frac{\partial v_m}{\partial a_k}\right)$$

$$\frac{\partial^2 v_m}{\partial a_j \partial d_k} = \frac{\partial^2 v_m}{\partial b_j \partial c_k} = - z^{-2} R_{Dj} \left(\frac{\partial v_m}{\partial a_k}\right)$$

$$\frac{\partial^2 v_m}{\partial b_j \partial d_k} = - z^{-3} R_{Dj} \left(\frac{\partial v_m}{\partial a_k}\right)$$

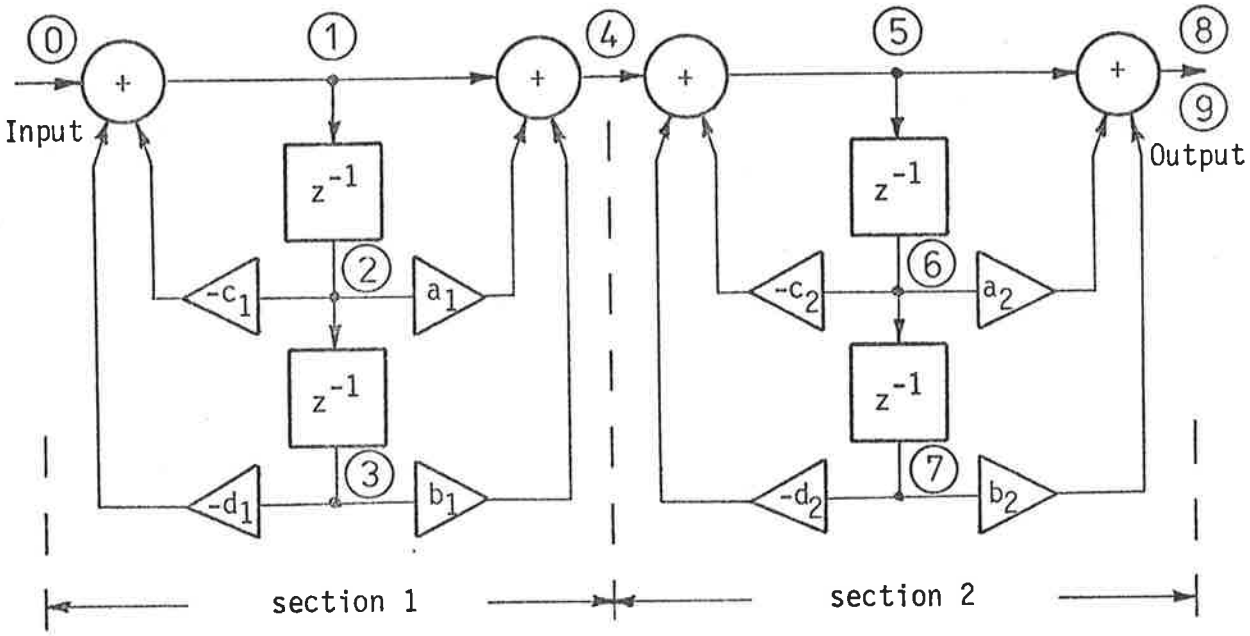
The listing of a FORTRAN subroutine which computes the objective function value, the gradient vector, the Gauss-Newton matrix and the Hessian matrix according to the principles of this section is given in Appendix B.

4.8 Tellegen's Theorem and the Adjoint Filter

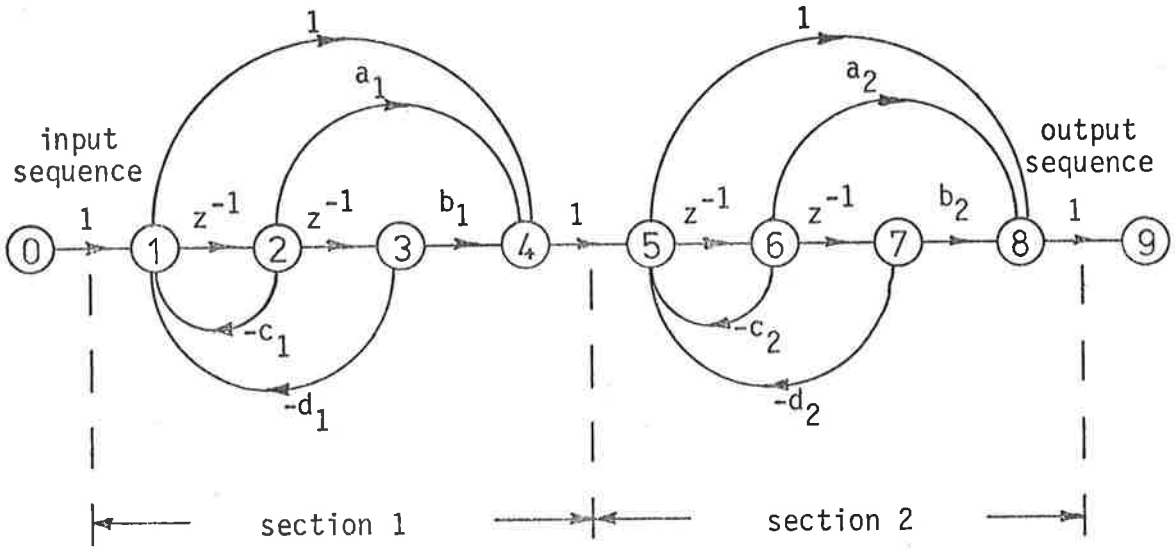
The theorem of Tellegen (1952) provides a very general, computationally efficient way to generate the first- derivative (or "sensitivity") vector of a network transfer function with respect to the network elements. This provides the basis of a method for the design by optimization of electrical networks in the frequency domain (Director and Rohrer, 1969a) and in the time domain (Director and Rohrer, 1969b).

This work for conventional electrical networks was extended to *signal-flow* networks (of which digital filters are an example) by Fettweis (1971c) and by Seviora and Sablatash (1971). Although the results are valid for a very general class of signal-flow networks, it appears that when the filter being considered is a cascade of second order sections the approach offers no advantages over the methods already described. This matter will be explored below.

The cascade-form digital filter shown in block-diagram form in figure 4.15 (a) may alternatively be represented as the signal-flow network of figure 4.15 (b). Such a network is made up of *nodes* (represented by numbered circles) and *branches* (represented by directed lines). Each node is considered to have a *signal value* which flows outwards into any branch whose input is connected to that node. The signal



(a)

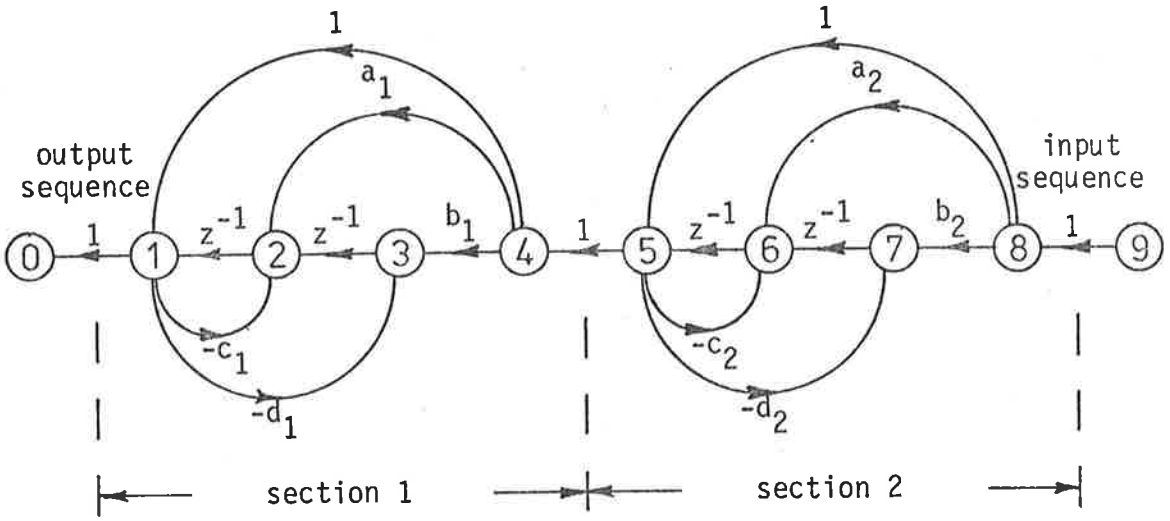


(b)

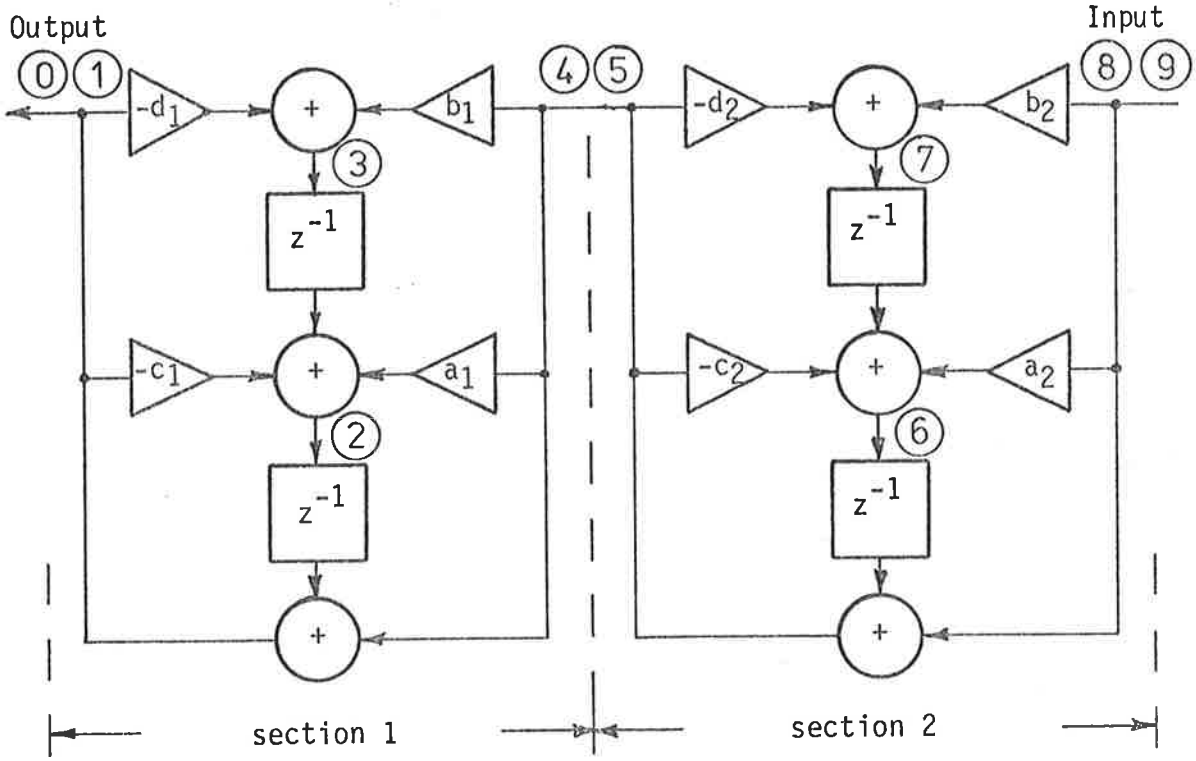
FIGURE 4.15 A Cascade-form Recursive Digital Filter

(a) Block Diagram Representation

(b) Signal-flow Network Representation



(a)



(b)

FIGURE 4.16 A Cascade-form Recursive Digital Filter

(the Adjoint of the Filter of Figure 4.15)

(a) Signal-flow Network Representation

(b) Block Diagram Representation

value is the *sum* of all the signals flowing out of those branches which are directed into the given node. Each branch has a transfer function which together with the branch input determines the branch output. In the present case each branch transfer function is either a constant multiplier or the unit delay operator z^{-1}.

The *adjoint* of a given signal-flow network is the network obtained by reversing every branch. For example, figure 4.16 (a) is the adjoint of figure 4.15 (b). This network may easily be translated back into the block-diagram form of figure 4.16 (b). Fettweis (1971c) proves that a signal-flow network and its adjoint are *inter-reciprocal*, which in the case of single-input single-output networks means that they have identical transfer functions. Thus the filters of figures 4.15 and 4.16 are alternative realizations for the same filtering action.

Fettweis (1971c) also proves an important result relating to the sensitivity of an overall network transfer function to the value of any individual branch transfer function, which may be stated as follows. Consider a signal-flow network, referred to as the "primary" network, whose nodes are numbered 1, 2, ... N, the input node having some number A and the output node number B. Assume that some node C is connected to another node D by one (and only one) branch with transfer function α, with the branch directed from C to D. A transfer function may be calculated for the signal path from the input node A to *any* node; in particular let that from input to output be denoted H_{AB} and that to node C be H_{AC}. Consider now the adjoint network. The nodes retain their numbering but now node B is the input

and node A the output. A branch of transfer function α now operates on the signal at node D to produce an input to node C instead of vice versa. Denoting transfer functions *in the adjoint network* by primed quantities, the result of Fettweis is that

$$\frac{\partial}{\partial \alpha} (H_{AB}) = H_{AC} H'_{BD} \tag{4.154}$$

(The inter-reciprocity result is of course expressed by

$$H'_{BA} = H_{AB} \quad). \tag{4.155}$$

Considering the steady-state frequency-domain case, just *one* complete analysis of the primary filter at each frequency would produce all the complex numbers H_{Ai} for $i = 1, 2 \ldots N$, and one analysis of the ajoint filter would produce the numbers H'_{Bi}, $i = 1,2 \ldots N$. All sensitivity components could then be found from (4.154) by multiplying the appropriate quantities, with no formal differentiation of analytic expressions being required.

We now return to the example shown in figures (4.15) and (4.16), and employ the notation for transfer functions introduced above. Accordingly, the transfer function of the entire filter is H_{09} (or H'_{90}). We observe that a multiplier branch of value a_1 is directed from node 2 to node 4 in the primary filter, and from (4.154) can calculate its sensitivity components as:

$$\frac{\partial H_{09}}{\partial a_1} = H_{02} H'_{94} \tag{4.156}$$

Similarly, the sensitivities for the other multipliers of section 1 are

$$\begin{aligned}
\frac{\partial H_{09}}{\partial b_1} &= H_{03}\, H'_{94} \\
\frac{\partial H_{09}}{\partial c_1} &= -H_{02}\, H'_{91} \\
\frac{\partial H_{09}}{\partial d_1} &= -H_{03}\, H'_{91}
\end{aligned} \tag{4.157}$$

The transfer function values required from the *adjoint* analysis H'_{94} and H'_{91} are those of the whole of section 2 and of the entire cascade filter respectively. Thus no special adjoint analysis is actually needed - analysis of the appropriate filter sections *in the primary form* generates the same values.

What has been shown for the example of two cascaded second-order sections obviously extends to any number. That is, any sensitivity may be found by multiplying the transfer function of a cascade of *complete* sections by a transfer function of the form of $\pm H_{02}$ or $\pm H_{03}$. From the signal-flow diagram of figure 4.15 (b) we clearly have

$$H_{02}(z) = \frac{z^{-1}}{1 + c_1 z^{-1} + d_1 z^{-2}} \tag{4.158}$$

and

$$H_{03}(z) = \frac{z^{-2}}{1 + c_1 z^{-1} + d_1 z^{-2}} \tag{4.159}$$

To quote formulae for the sensitivities with respect to the multipliers in a general cascade filter of K sections, we define for the kth section

$$H_{Nk}(z) = 1 + a_k z^{-1} + b_k z^{-2} \tag{4.160}$$

and

$$H_{Dk}(z) = 1 + c_k z^{-1} + d_k z^{-2} \tag{4.161}$$

so that the overall transfer function H(z) is

$$H(z) = \frac{\prod_{k=1}^{K} H_{Nk}(z)}{\prod_{k=1}^{K} H_{Dk}(z)} \tag{4.162}$$

and the transfer functions which constitute the sensitivities are

$$\left.\begin{aligned} \frac{\partial H(z)}{\partial a_j} &= \frac{\prod_{\substack{k=1 \\ k\neq j}}^{K} H_{Nk}(z)}{\prod_{\substack{k=1 \\ k\neq j}}^{K} H_{Dk}(z)} \cdot \frac{z^{-1}}{H_{Dj}(z)} \\ \frac{\partial H(z)}{\partial b_j} &= \frac{\prod_{\substack{k=1 \\ k\neq j}}^{K} H_{Nk}(z)}{\prod_{\substack{k=1 \\ k\neq j}}^{K} H_{Dk}(z)} \cdot \frac{z^{-2}}{H_{Dj}(z)} \end{aligned}\right\} \tag{4.163}$$

$$\frac{\partial H(z)}{\partial c_j} = \frac{\prod_{k=1}^{K} H_{Nk}(z)}{\prod_{k=1}^{K} H_{Dk}(z)} \cdot \frac{-z^{-1}}{H_{Dj}(z)}$$

$$\frac{\partial H(z)}{\partial d_j} = \frac{\prod_{k=1}^{K} H_{Nk}(z)}{\prod_{k=1}^{K} H_{Dk}(z)} \cdot \frac{-z^{-2}}{H_{Dj}(z)} \qquad (4.163)$$

The foregoing analysis via Tellegen's theorem and the adjoint digital filter has merely succeeded in deriving the same gradient filters as were found for time-domain gradient sequences in Section 4.7.2. The expressions (4.163) are no more (and no less) than what would be obtained by direct differentiation of the z-transform expression (4.162). Thus the adjoint method seems to offer no computational advantages in either the time or the frequency domain for the type of filters being considered.

4.9 Special Considerations in Digital Filter Design by Optimization

4.9.1 Stability of Filters

As mentioned in Section 4.6.3, it is not necessary to consider the stability of the digital filter at every stage of the optimization process when the magnitude frequency response alone is of interest. This is because the magnitude response shape is unchanged if a complex conjugate pole pair

at $z = r \exp(\pm j\phi)$ is replaced by a pair at $z = r^{-1} \exp(\pm j\phi)$ or a real pole at $z = p$ is replaced by one at $z = p^{-1}$. A constant gain factor associated with this *root reflection* is irrelevant because it automatically becomes incorporated into the parameter q in formulation B2 or B3 (Section 4.4.3). Stability of the filter can be guaranteed by performing reflections for poles which are outside the unit circle. However, it may *not* be sufficient to do this just once at the end of optimization - see Section 4.9.2. A similar situation exists with respect to zeros - if a minimum phase design is desired this may easily be guaranteed by reflecting outlying zeros inside the unit circle also.

For other classes of digital filter design, that is, time domain design or frequency domain design in which phase or group delay response is considered as well as magnitude, poles must always remain inside the unit circle. For a cascade-form filter this translates to a requirement for the c_k and d_k coefficients of each section k to remain within a triangular region of their plane, as shown in figure 4.17. The region above the parabola $d_k = \frac{1}{4} c_k^2$ corresponds to a complex conjugate pair of poles and the remainder of the triangle to real poles. The stability of the section is readily checked without actually evaluating the poles - that is, the filter is stable if, for every denominator section,

$$\left.\begin{aligned} & d_k < 1 \\ \text{and} \quad & 1 + c_k + d_k > 0 \\ \text{and} \quad & 1 - c_k + d_k > 0 \quad . \end{aligned}\right\} \qquad (4.164)$$

During the application of a gradient method of optimization,

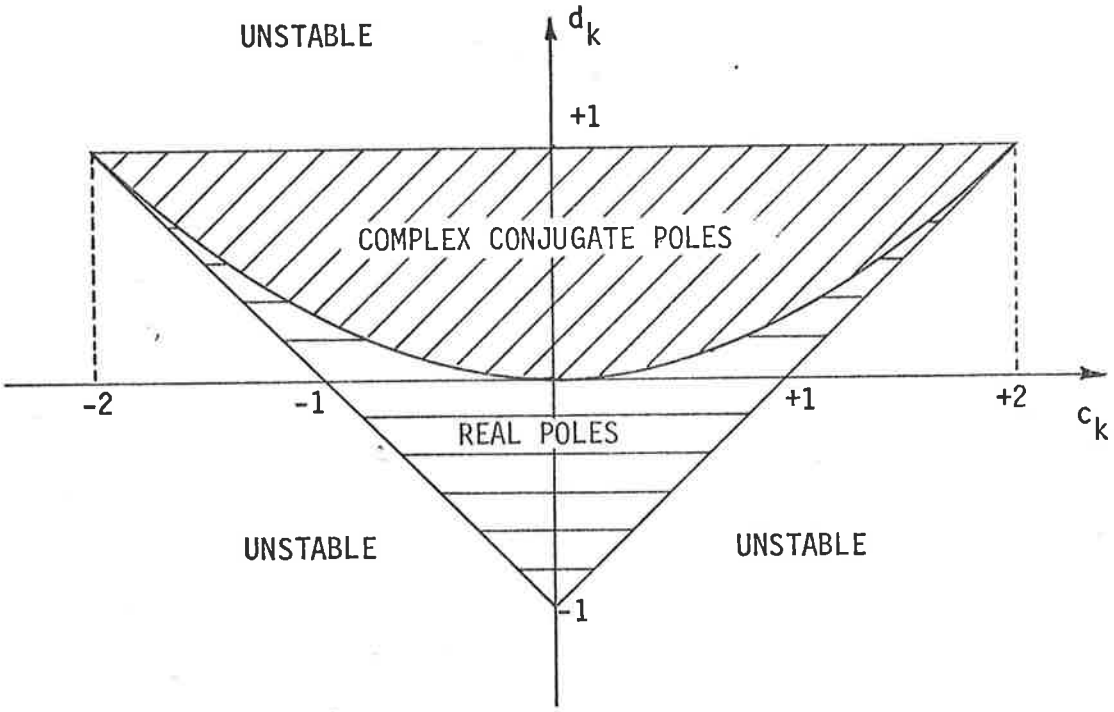


FIGURE 4.17 Stability Diagram for Coefficients (c_k , d_k) of kth Cascaded Section

the subroutine which evaluates the objective function is normally called on two distinct occasions. The first, which occurs once per iteration, is the main call to evaluate the gradient vector (and possibly also the Hessian or Gauss-Newton matrix) in order to determine a search direction $\mathbf{p}$ in parameter space. The values of c_k and d_k (for some arbitrary section k) prevailing at this call may be as shown by $c_k^{(j)}$, $d_k^{(j)}$ in figure 4.18. (The superscript (j) represents the iteration number), The search direction $\mathbf{p}^{(j)}$ projects as a semi-infinite line segment in the (c_k, d_k) plane, along which the point $(c_k^{(j+1)}, d_k^{(j+1)})$ must eventually lie. The second occasion on which the function subroutine is called, returning(usually) only the function value itself, is in making trials for the new parameter vector $\mathbf{x}^{(j+1)}$ and the (c_k, d_k) projections of all such trial vectors must also lie along the line. If the first trial steplength α violates the stability requirement in any section (readily checked from (4.164)), no attempt is made actually to evaluate the function value. Rather, the trial α is *either* repeatedly reduced by some constant factor (figure 4.18) until stability is gained in all sections, *or* the necessary reduction factor is calculated for each section (a matter of simple geometry) and a value of α used which satisfies the most stringent of these requirements by some small margin. In this way it can be guaranteed that the stability constraints are not violated by the new parameter vector $\mathbf{x}^{(j+1)}$. Of course, the value $\alpha^{(j)}$ actually employed may be smaller than the upper bound arising from filter stability considerations, because there is a necessity to secure a function decrease at each iteration

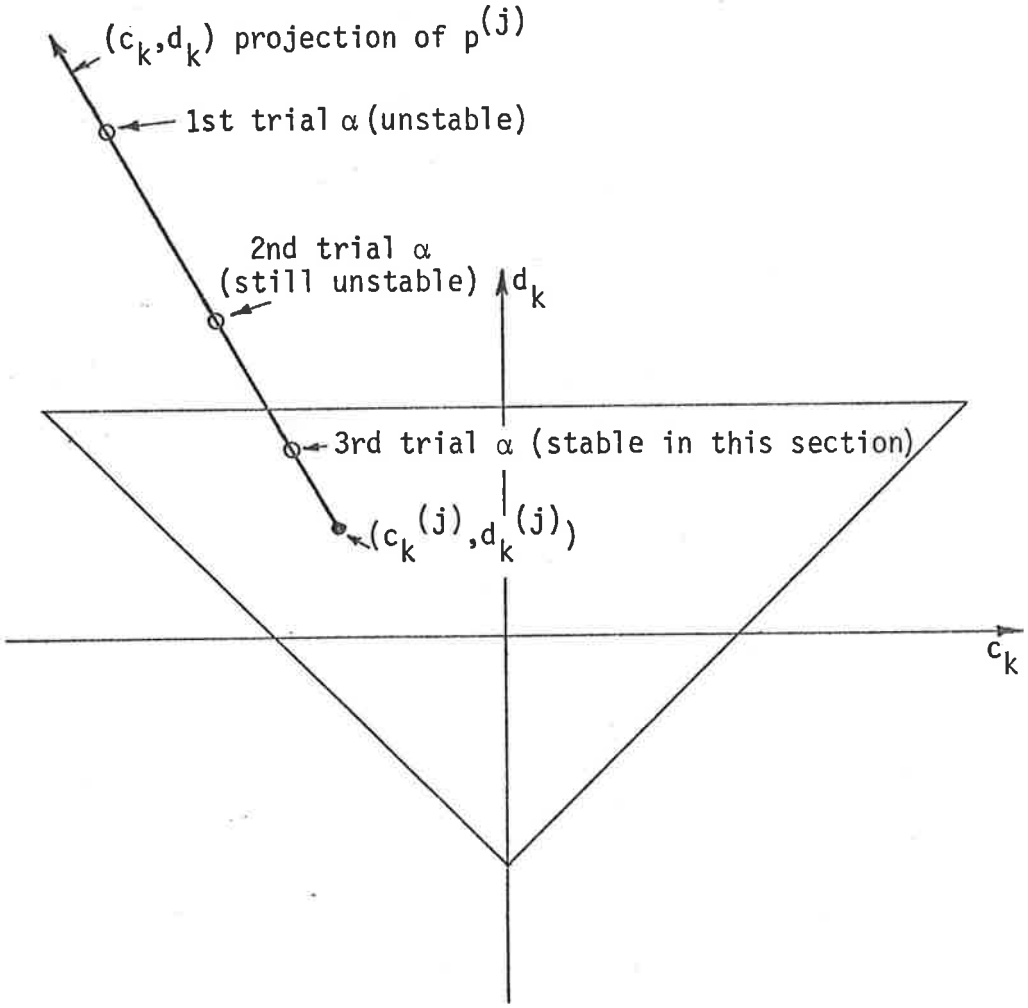


FIGURE 4.18 Successive Reduction of Trial Steplength to Ensure Filter Stability

(the *algorithmic* "stability" condition). There possibly is also an approximate line minimization requirement, depending on the method in use.

If the design is being performed in the time domain and recursive sections are in use for computing gradients (section 4.7.2) the coefficients of the numerator sections must be constrained in a similar way.

In many cases, restricting the step length as described above is sufficient to ensure convergence to a local minimum which is an interior point of the feasible region. This is particularly true when matching a decaying impulse response in the time domain, because parameter vectors just inside the stability boundary cause slowly-decaying time sequences with a consequent bad fit of the later time samples and a high objective function value. The search direction arising from a point near the stability boundary is most likely to be directed back into the feasible region.

It can happen, however (particularly with group delay designs) that successive iterates converge to a point on a stability boundary which is not even a constrained local minimum (in the sense that the gradient vector is normal to the constraint.) In such cases a "feasible direction technique" such as gradient projection (section 2.3.3) may occasionally succeed in returning the search to the interior of the feasible region. However, it may also terminate at a point on the stability boundary, and in such a case the only remedy would be to re-start the search from a different feasible starting point. In the work to be reported in Chapter Five, the added complication of gradient projection is avoided. A search is abandoned (and a new starting point used) before embarking on

the jth interation whenever the vector $\mathbf{x}^{(j)}$ is within a certain small distance (0.001) of any stability boundary. For each k, a thin strip just inside the boundary of the triangular region of figure 4.18 is thus unavailable to the search procedure. The iteration parameter vector is allowed to enter such a strip, but solely for the purpose of detecting the termination condition. This procedure has the theoretical objection that part of the feasible region becomes inaccessible, but few practical filters would have such coefficients anyway.

4.9.2 Avoidance of False Local Minima

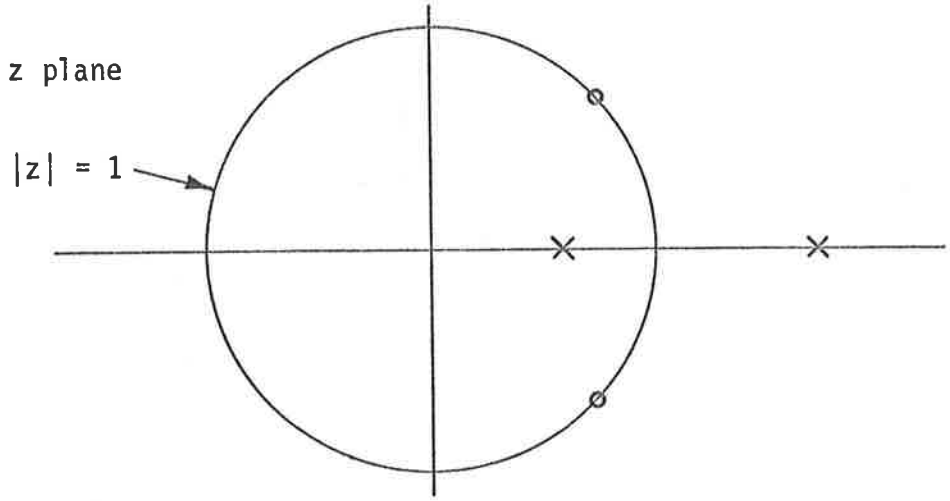
As with most practical nonlinear minimization problems, we are not dealing with a convex objective function and so there is no guarantee that the procedure will converge to the best local minimum. However, because the objective funtion has a special form it is possible to predict the existence of certain types of bad minima and modify the procedure to avoid them. Experience (reported in Chapter Five) has shown that with the algorithms suitably modified it is possible to reach the (presumably) global optimum from a very wide range of starting points in well-posed digital filter design problems.

With any multi-section (cascade) filter there are at least as many local minima as there are permutations of the order of the sections. However, this is not important, as all these solutions are equally good (section ordering to optimize dynamic range or noise performance is a separate topic to be considered only after the theoretical solution has been found).

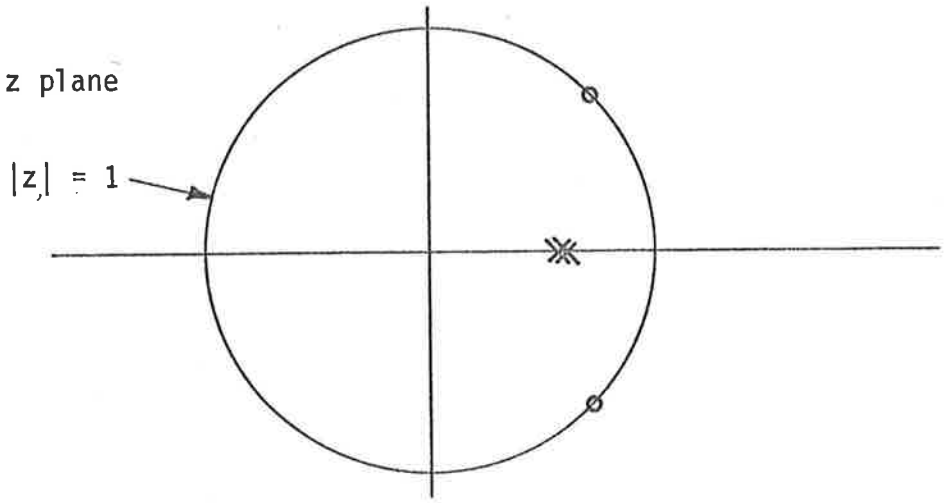
Steiglitz (1970) noted the appearnce of one type of undesirable local minimum while treating the problem of magnitude

response optimization. He allowed poles and zeros to move to any position in the z-plane during the iteration process because stability (and minimum phase) could eventually be restored by reflecting roots with $|z| > 1$ to the inside of the unit circle, with no change to the objective function. In one example (involving one pole pair and one zero pair) the iterative process converged to a "solution" with pole-zero configuration as in figure 4.19 (a). The two real poles are almost exact reciprocals of each other, and after inversion lie side by side as in figure 4.19(b). However, this solution is no longer a local minimum of the objective function - the value cán be reduced by continuing the iterations, during which time the two real poles become a complex conjugate pair (figure 4.19(c).

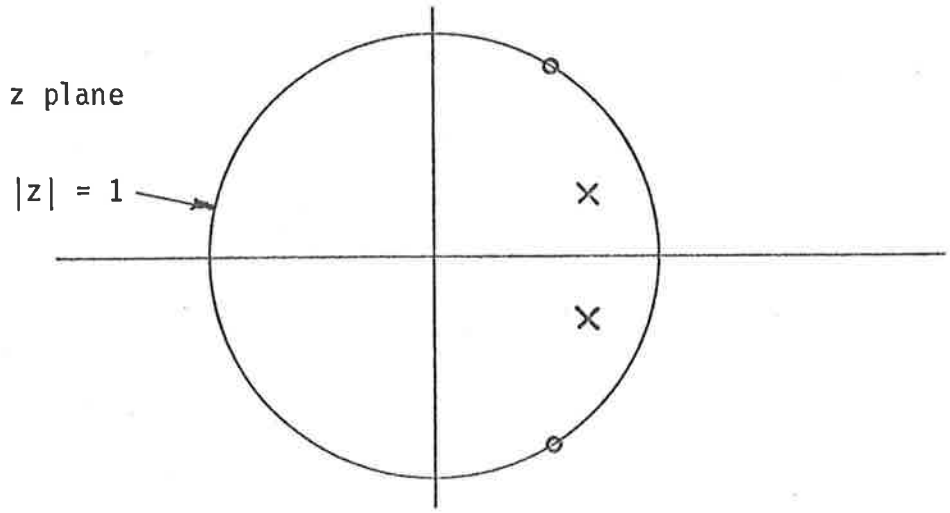
In Steiglitz's example the pole reflection to restore stability provided the mechanism that brought the poles together on the real axis allowing them to go complex, and it may well have been that many iterations would have been saved if the situation had been recognised earlier (i.e. without waiting for convergence the first time). However, this phenomenon is not really connected with stability, but is a general problem with real roots. This is because each second-order section determines *two* roots. If there are more than two real roots they may have become paired into sections in any combination, without affecting the filter transfer function. Real roots have the opportunity to go complex only if they are paired together in the *same* section. There is thus the possibility of local minima of the type shown in figure 4.20 (a). Even though all poles are inside the unit circle the configuration is sub-optimal and can be improved by changing the pairing from (A,B and



(a) at original local minimum



(b) after pole reflection

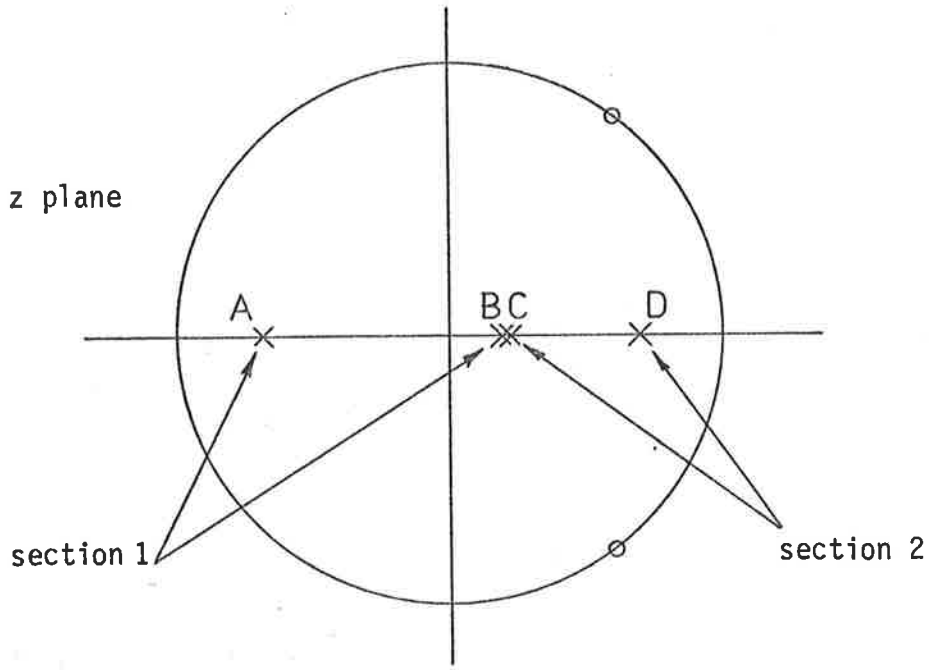


(c) after further iterations

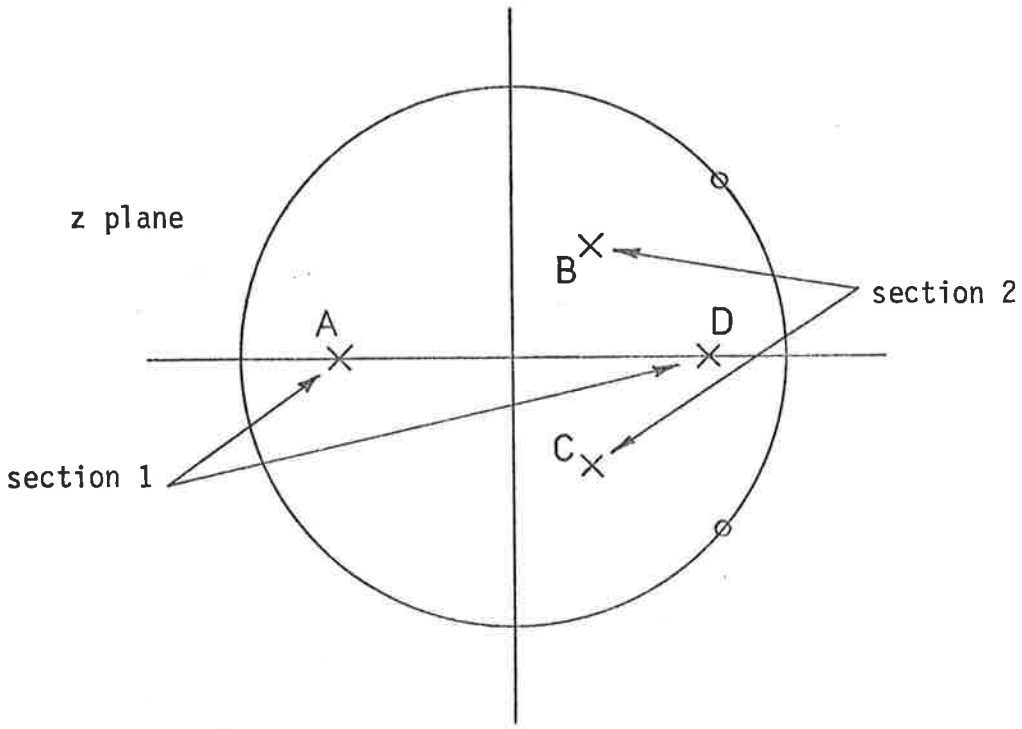
FIGURE 4.19 Pole-Zero Configurations

Example with 1 Pole Pair and 1 Zero Pair

× = pole; o = zero



(a) at original local minimum



(b) after changing pairing and continuing iterations

FIGURE 4.20 Pole-Zero Configurations

Example with 2 Pole Pairs and 1 Zero Pair

× = pole; o = zero

C,D) to (A,D and B,C). After more iterations the situation shown in figure 4.20 (b) results.

Changing the pole and zero pairing does not always result in real roots becoming complex. However, experience has shown that when two poles (or zeros) are driven close together on the real axis that this usually occurs. Furthermore, the close approach of two real roots is a *necessary* prelude to their going complex. This suggests the following scheme for avoiding this type of undesirable local minimum and simultaneously ensuring the ultimate stability of the filter, when optimizing a magnitude response only.

At the start of each iteration:

a. Compute the positions of all poles from the known values of c_k and d_k for each $k = 1, 2 \ldots K_D$.

b. For any complex conjugate poles $r_k \exp(\pm j\phi_k)$ which have $r_k > 1$, re-compute new values of c_k and d_k to correspond to $r_k^{-1} \exp(\pm j\phi_k)$ (that is, reflect poles inside unit circle).

c. Leave unchanged the c_k and d_k corresponding to complex conjugate poles with $r_k < 1$.

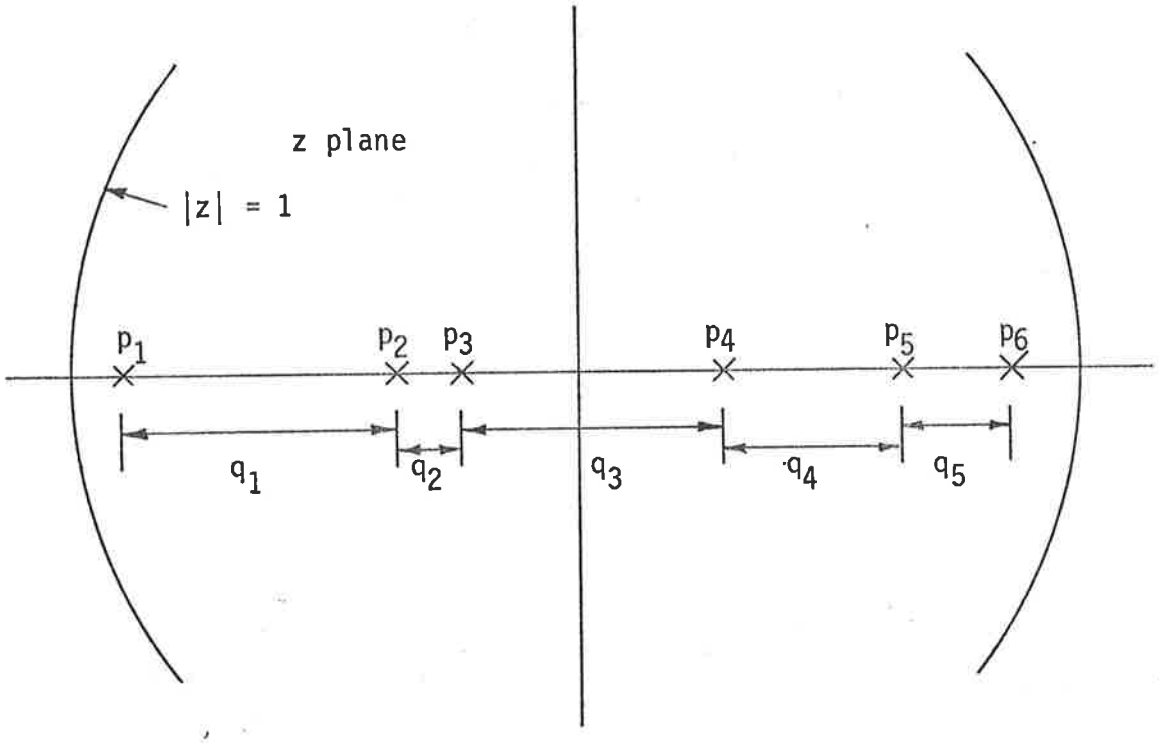
d. Replace any real poles having $|z| > 1$ with their reciprocals, and place all real poles in a "pool" where they lose their connection with a particular section (k value).

e. Sort all poles in the pool into increasing order, say $p_1, p_2 \ldots p_J$, and compute the distances between adjacent poles

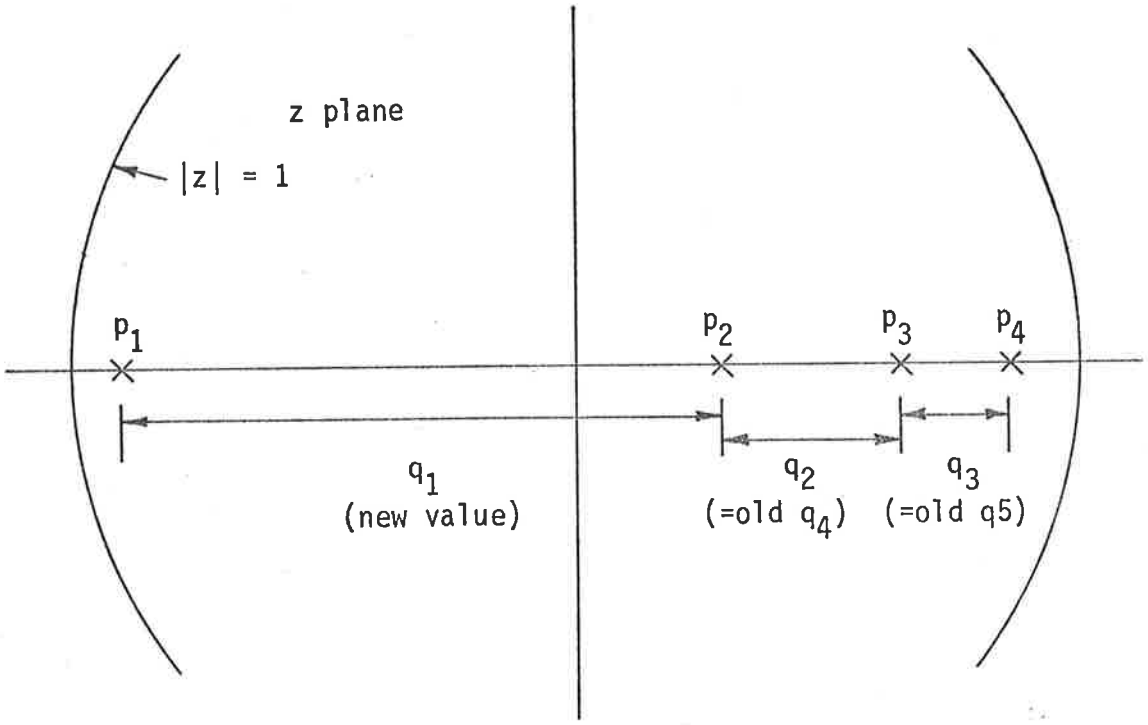
$$q_i = p_{i+1} - p_i \qquad i = 1, 2 \ldots J - 1 \quad (4.165)$$

as shown in figure 4.21 (a).

f. Pair the two closest poles (that is p_j and p_{j+1} where



(a) after reflecting unstable poles if any



(b) after removing closest pair and re-indexing

FIGURE 4.21 Real Pole Configurations during Pairing Procedure

$q_j < q_i$ for all i) and calculate corresponding new values of c_k and d_k for one of the values of k which was "vacated" when the poles were pooled.

g. Remove the paired poles from the pool and (if they were not the first or last two in the pool) compute the one new distance value $p_{j+2} - p_{j-1}$, and re-index the p and q values (figure 4.21 (b)).

Repeat steps f and g until only two poles remain, then pair those to form the last remaining section. In this way real poles become paired as soon as they are driven sufficiently close together, giving them the opportunity to go complex as soon as possible. The entire procedure is repeated for zeros.

When dealing with a time domain or group delay specification, it is not possible to reflect poles and zeros inside the unit circle with no change to the objective function. Pole positions must be controlled so that they do not wander across the unit circle, and if we are trying to match a minimum-phase time sequence using recursive sections to generate the gradients (Section 4.7.2) so must the zero positions. However, the procedure for *pairing* real roots is still relevant.

If any root reflection is performed or any change made to root pairing, the gradient vector and Hessian matrix will be affected (even though the value of the objective function is not changed). It follows that second-derivative and Gauss-Newton gradient methods (which re-calculate the iteration matrix **A** afresh at every interation) have a fundamental advantage over the quasi-Newton and conjugate-gradient algorithms when used in these problems. With a quasi-Newton method, the matrix $\mathbf{A}^{(k)}$ carried from the kth iteration should be positive

definite and so will still lead to a descent direction in iteration $(k + 1)$ even if the parameter vector is permuted. However, its usefulness as an approximation to the new Hessian is questionable and it may be better to reset $\mathbf{A}$ to the identity matrix. This leads to the idea of revising root pairing only after a certain number of iterations, to allow the quasi-Newton method a chance to operate. This matter is investigated experimentally in Chapter Five.

CHAPTER FIVE

5. COMPARATIVE TESTS OF OPTIMIZATION ALGORITHMS

5.1 Introduction

The purpose of this chapter is to report the results of numerical tests of various unconstrained minimization algorithms as applied to typical digital filter design problems. The emphasis is on the relative performance of the algorithms themselves rather than on the utility of the resulting designs. The examples were chosen to be reasonably representative of the types of problems met in practice, so that the conclusions of this chapter might be a useful guide to the selection of a minimization algorithm.

For simplicity, all of the computer routines were programmed in FORTRAN and run on a large, general purpose computer (a CDC Cyber 173) via an interactive remote terminal. Attempts have been made to evaluate the algorithms on the basis of reliability and execution speed. Some care has been taken to avoid obvious inefficiences in coding in the frequently-called subroutines (such as those for function, gradient and Hessian evaluation, and for line searches) so that some meaning may be attached to speed comparisons. However, it is realized that in the adaptation of an algorithm to a real-time system modelling application (on a minicomputer or array processor), advantage would be taken of special machine features and the economies of assembly-language programming, and some algorithms would benefit more than others. Accordingly, results of comparisons in FORTRAN need to be interpreted with such qualifications borne in mind.

5.2 A General-Purpose Test Program

5.2.1 Features of Program

A general-purpose driver program was written to facilitate the comparison of many types of gradient-based minimization algorithms applied to "formulation B" digital filter design problems. The program deals with filters of the cascade form. A simplified flow diagram is shown in figure 5.1. The principal features are:

a. Algorithms of the quasi-Newton, conjugate gradient, true Marquardt, Gauss-Newton and second-derivative types may be tested using the same driver program, by loading different functional subroutines and selecting the appropriate flow paths at run time. "True Marquardt" refers to the type of method which uses no line search; the diagonal elements of the **A** matrix are increased until the step with $\alpha = 1$ yields a function value decrease. Thus a dummy "line search" subroutine (never called) is loaded in this case; a dummy "Marquardt" subroutine is loaded otherwise.

b. Various combinations of search direction strategies (e.g. modified Newton methods) and line search techniques may be tested by loading different "DIREC" and "LINE" subroutines.

c. Function, gradient and Hessian evaluation is performed by a subroutine, so that the same driver program is directly usable for magnitude, log magnitude and magnitude squared response optimization. Furthermore, logical switches are set in the driver program so that the subroutine evaluates only the function value when called from a LINE routine, and does not evaluate the $\mathbf{R}$ $(= 2\ \mathbf{J}^T\mathbf{J})$ matrix or the $\mathbf{H}$ matrix unless the method in

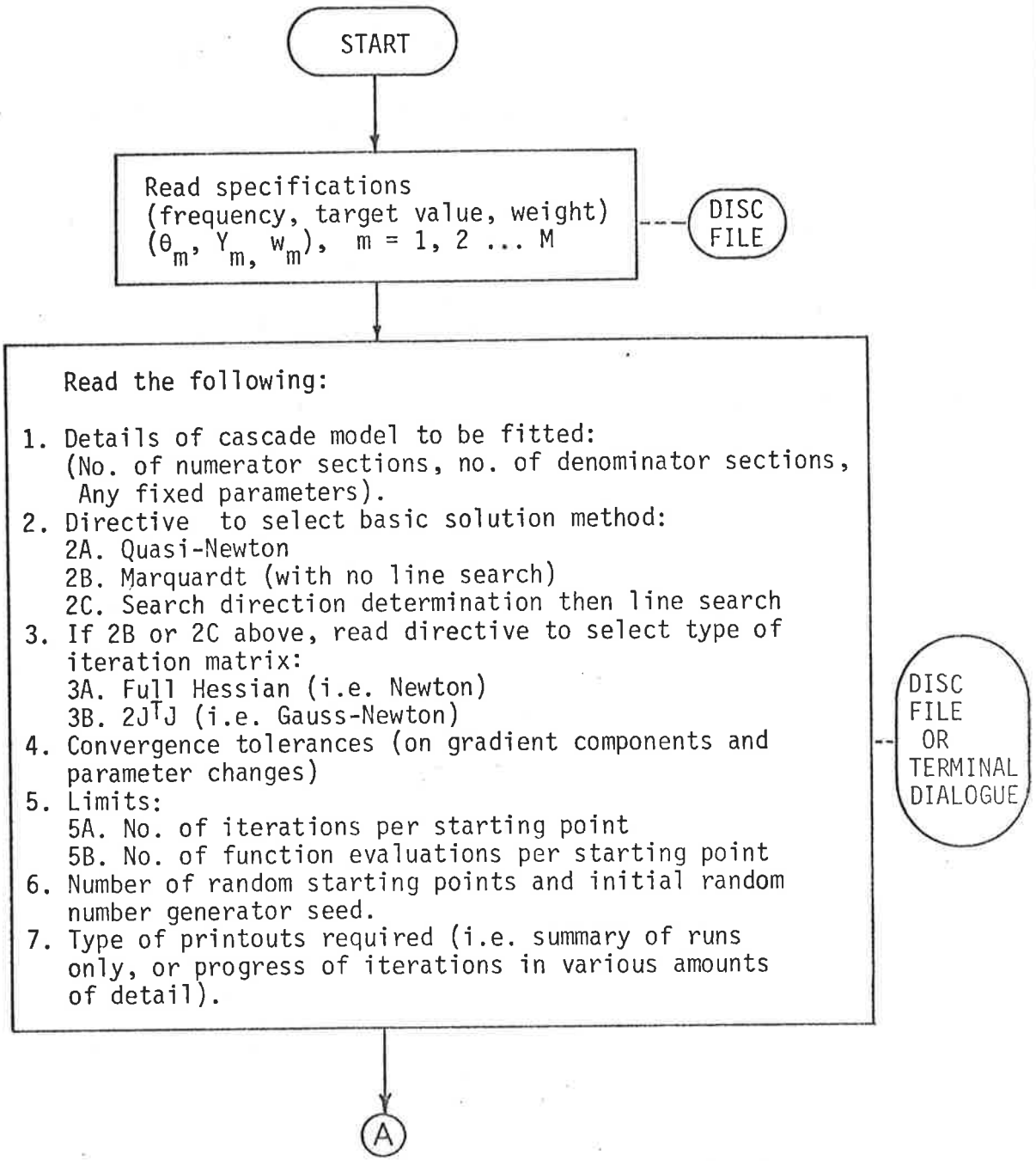


FIGURE 5.1 Sheet 1 of 3

Simplified Flow Chart of General Test Program

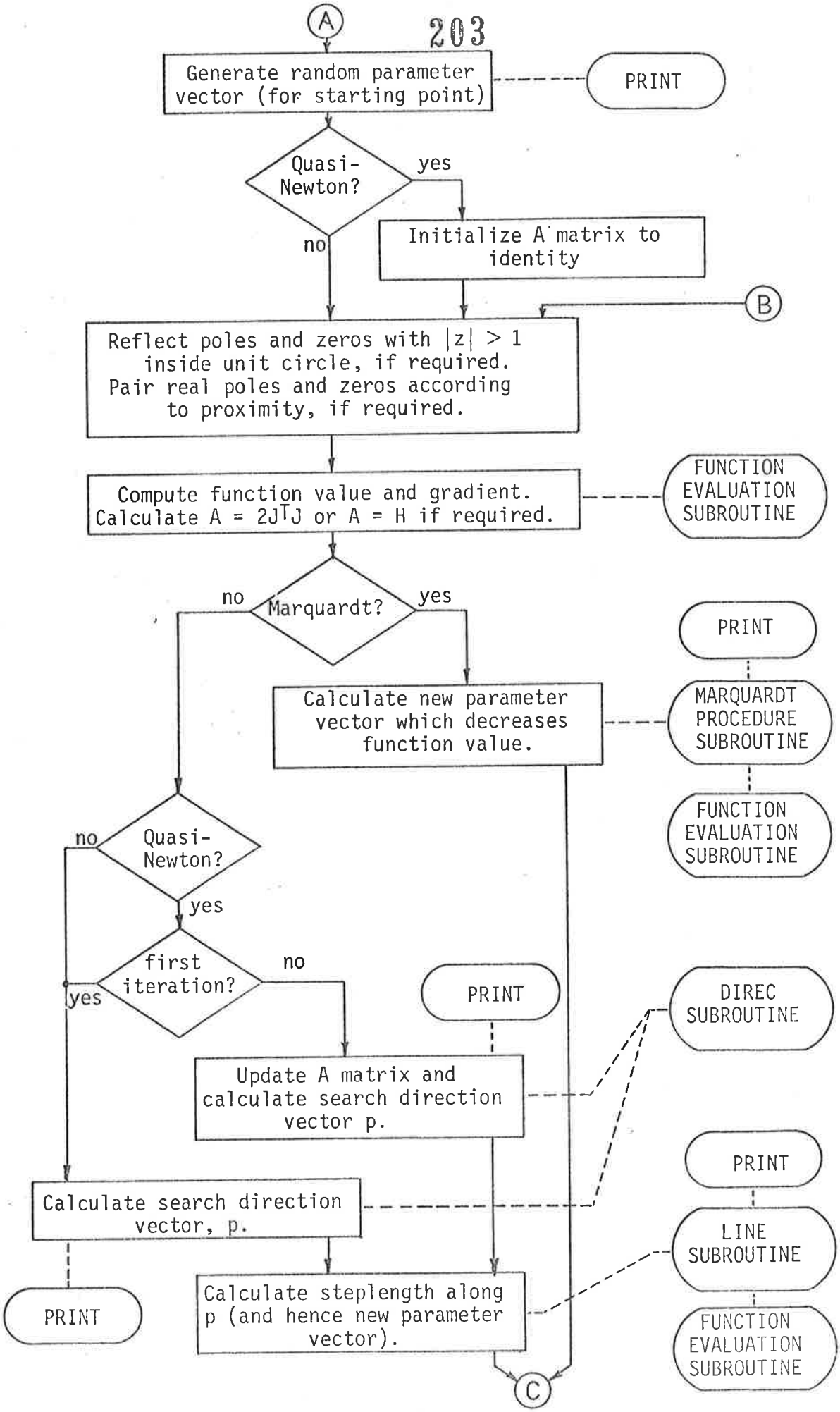


FIGURE 5.1 Sheet 2 of 3

Simplified Flow Chart of General Test Program

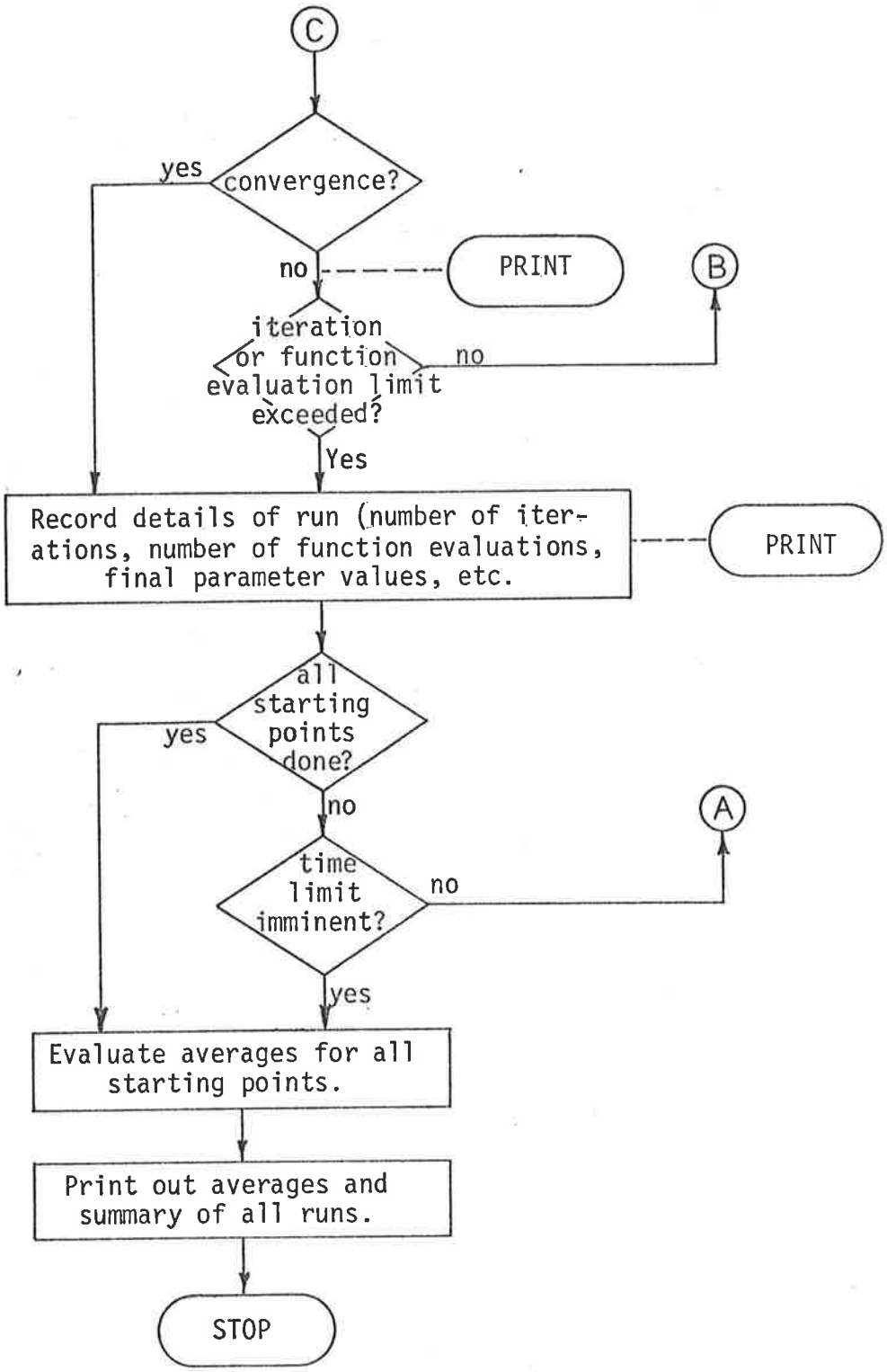


FIGURE 5.1 Sheet 3 of 3

Simplified Flow Chart of General Test Program

use calls for it.

d. Further logical switches bypass the root reflection sequences and also allow allpass filter sections to be treated. This makes the program suitable also for time domain and group delay optimization tests.

e. In order to test the ability of the various methods to converge from widely differing starting points, a number of runs are made (as specified in terminal dialogue), with randomized starting points. The generation of starting points is considered in sections 5.2.2 and 5.2.3.

f. The amount of detail printed out is selectable at run time. The extremes are: a summary printed at the end of execution giving number of iterations, etc. for each run (starting point) and averages over all runs; and detailed examination of each step during a line search. The more detailed printouts greatly aided debugging and selection of parameters when experience with the algorithms was limited. Examples of printouts are given as figures 5.2 (summary) and 5.3 (moderately detailed).

5.2.2 Generation of Random Starting Points, Method 1.

Although in many practical cases a reasonably good starting estimate would be available, it was thought advisable to test the ability of optimization algorithms to converge from arbitrary (stable) guesses for the parameter vector. This procedure is relevant from several points of view:

a. It gives a feel for the quality (closeness to the ultimate solution) required in the initial guess, if convergence to a given solution is to be virtually guaranteed. Thus it aids in gauging the amount of effort necessary in arriving at such a guess. This

```
########################################

OPTIMIZATION TEST - OVERALL DETAILS

########################################

 NO. OF STARTING POINTS                          8
 NO. REQUIRING  100 ITERATIONS                   0
 NO. NOT CONVERGING TO F.LT.1.001*FMIN           0
 NO. NOT CONVERGING TO F.LT.1.1   *FMIN          0
 NO. NOT CONVERGING TO F.LT.2.0   *FMIN          0
```

********SUMMARY********

START POINT NO.	FINAL FUNCTION VALUE	NO. OF ITERATIONS TOTAL	NO. OF ITERATIONS BEFORE NEWTON	AUX FUNC EVALS	MATRIX FACTORIZATIONS TOTAL	MATRIX FACTORIZATIONS COMPL	AVERAGE EFFICIENCY BEFORE NEWTON	AVERAGE EFFICIENCY ALL STEPS
1	7.744701E-03	62	43	284	120	62	.132	.216
2	7.744701E-03	61	44	265	116	61	.139	.224
3	7.744700E-03	63	46	285	124	63	.135	.241
4	7.744700E-03	74	48	600	137	74	.116	.279
5	7.744701E-03	77	58	332	154	77	.101	.171
6	7.744700E-03	51	33	261	97	51	.172	.280
7	7.744700E-03	56	39	253	100	56	.150	.252
8	7.744700E-03	44	25	185	79	44	.220	.313

******** OF THOSE RUNS CONVERGING TO BETTER THAN 1.001*FMIN ********

```
AVERAGE NUMBER OF ITERATIONS                          61.000
AVERAGE NUMBER OF 'PRE-NEWTON' ITERATIONS             42.000
```

FIGURE 5.2 Example of Summary Printout, General Test Program (continued next page)

```
AVERAGE NUMBER OF AUXILIARY FUNCTION EVALUATIONS      308.125
AVERAGE NUMBER OF MATRIX FACTORIZATIONS ATTEMPTED     115.875
AVERAGE NUMBER OF MATRIX FACTORIZATIONS COMPLETED      61.000
AVERAGE EFFICIENCY OF FIRST PRE-NEWTON STEPS             .684
AVERAGE EFFICIENCY OF SECOND PRE-NEWTON STEPS            .465
AVERAGE EFFICIENCY OF THIRD PRE-NEWTON STEPS             .298
AVERAGE EFFICIENCY OF ALL PRE-NEWTON STEPS               .138
AVERAGE EFFICIENCY OF ALL STEPS                          .242


NO OF ROOT REFLECTIONS =      2
NO OF PAIRING CHANGES =      26


ROOT REFLECTIONS OCCURRED AT ...

 START NO.                ITERATION NUMBERS

       2          3
       8          2

PAIRING CHANGES OCCURRED AT ...

 START NO.                ITERATION NUMBERS

       2          1     2    16    19    20
       3          1     2     3     4
       5         42
       6          0     1     2     3
       7          1     8    10
       8          0     1     2     3     4     5     6     7     8

    STOP
```

FIGURE 5.2 (continued from last page) Example of Summary Printout, General Test Program

```
*****************************

NEW STARTING POINT, NO.    1     FUNCTION VALUE=    1.419936E+02

*****************************

      IT=    1     START FN VALUE=    1.419936E+02
      PARAMETER VALUES
          1.337742E-01              -5.936663E-01              5.541614E-01
          4.887373E-01              -7.587944E-02              7.904560E-01
         -4.118205E-01               4.368318E-01              3.383327E-01
          3.172076E-01              -7.750989E-01              1.137995E-01
 GRAD NORM=   5.108E+02      OPT GAIN=   7.571E-01
 HESS NOT POS DEF  IP=   3
 NEWT STEP=   5.536E-01          ALPHA=   9.120E-01         DF/DA= -2.268E+02
   D2F/DA2=   1.665E+02
     AUX FN EVALS=    4     TOTAL MATR FACTS=   10     COMPL MATR FACTS=    1

      IT=    2     START FN VALUE=    2.388019E+01
      PARAMETER VALUES
         -1.106066E-01              -7.240468E-01              3.758504E-01
          4.700736E-01              -3.948627E-01              7.486944E-01
         -2.676421E-01               4.511722E-01              4.366670E-01
          3.244192E-01              -7.021935E-01              3.315078E-02
 GRAD NORM=   2.023E+02      OPT GAIN= -4.394E-02
 HESS NOT POS DEF  IP=   3
 NEWT STEP=   2.138E-01          ALPHA=   1.000E+00         DF/DA= -2.395E+01
   D2F/DA2=   1.496E+01
     AUX FN EVALS=    3     TOTAL MATR FACTS=    2     COMPL MATR FACTS=    1

      IT=    3     START FN VALUE=    9.688939E+00
      PARAMETER VALUES
         -1.400843E-01              -6.642902E-01              3.016813E-01
          4.583083E-01              -4.463732E-01              7.255630E-01
```

FIGURE 5.3 Example of Moderately Detailed Printout, General Test Program

may have implications for the application of optimization to real-time system identification problems, for example.

b. In off-line applications, the use of several arbitrary starting points can often replace any informed initial guess procedure. This is in fact how the solutions were discovered in the test problems to be discussed. If the objective function has several local minima, the use of a wide variety of starting points usually succeeds in uncovering them (most of them?) allowing the best to be chosen. Such a precaution would be mandatory in a real design situation.

c. Occasionally, solutions occur which appear to be very good in the sense of having a low objective function value, but which on examination have some unacceptable feature. An example is quoted in section 5.11, in which the algorithm caused a very high-Q pole to be placed in a "transition" band because the target response had been inadequately specified. The use of a wide variety of starting points is useful in discovering any such inadequacies in the definition of the objective function.

d. In the context of the testing of algorithms (as opposed to their use in a real design case), the use of arbitrary starting points allows their performance to be averaged over regions of parameter space which present varying degrees of "difficulty". This reduces the possibility that a given algorithm could seem comparatively good (or bad) simply due to a fortuitous choice of starting point.

Since the cascade realization is used, it is simple to generate random positions for each pair of poles and zeros independently. In practice it has been found that divergence of the algorithm is common if poles and zeros are initially close to the unit circle. The response curves of such filters would have

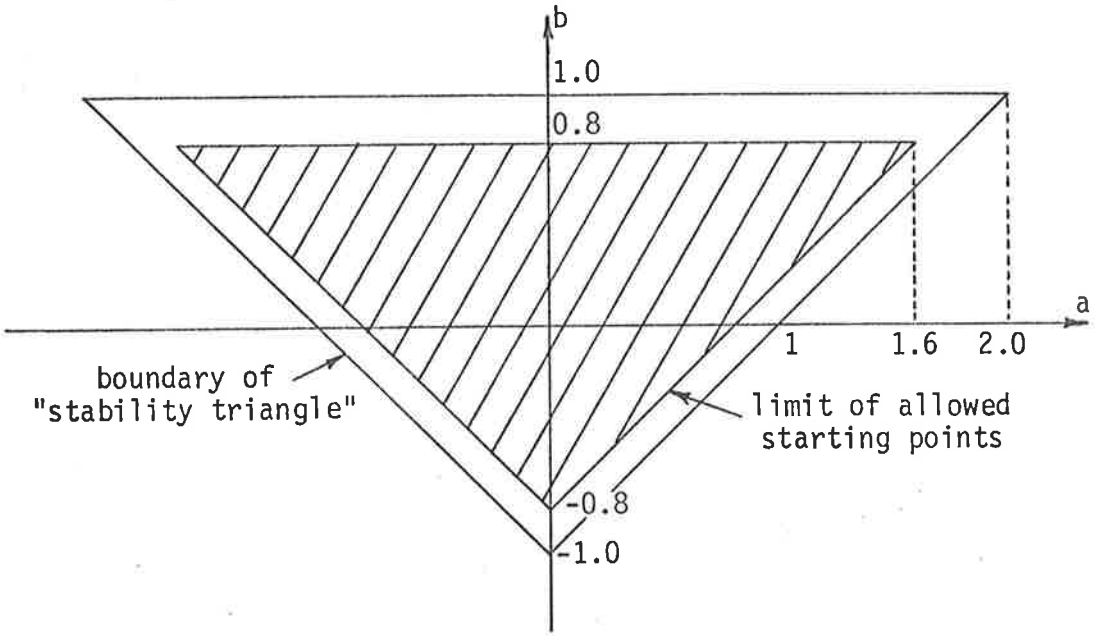
major features (peaks and troughs) which are likely to be very different from those of the response being approximated. Convergence is much more likely if the initial trial filter response is relatively "featureless". Accordingly, in the scheme finally adopted, the a and b (or c and d) coefficients of each section are randomly selected so that the points generated are restricted to a smaller triangle within the "stability triangle", and uniformly spread therein (figure 5.4 (a)).

The computer system FORTRAN library provides a random number generator outputting uniformly - distributed random numbers between 0 and 1. Two calls are made to this routine for each filter section, returning numbers r_1 and r_2. The a (or c) coefficient is first evaluated from r_1 and the b (or d) coefficient then generated using the a value and the randon number r_2 (which is assumed to be independent of r_1). We consider first the case of $r_1 > 0.5$. Since the probability that $r_1 > 0.5$ is 0.5 and we want $a > 0$ in 50% of cases, we specify that $r_1 > 0.5$ is to generate some $a > 0$. Further, we assume that a is to be related to r_1 in a monotone increasing sense. The portion of the allowed triangle for $a > 0$ is shown again in figure 5.4 (b). We denote any particular value of a as a' and the value of r_1 which is related to it as r_1'. For a uniform spread of starting points, the ratio of the *area* of the shaded trapezoid in figure 5.4 (b) to the total area of the triangle must be equal to the probability that r_1 is less than r_1' (given that it is greater than 0.5). Thus

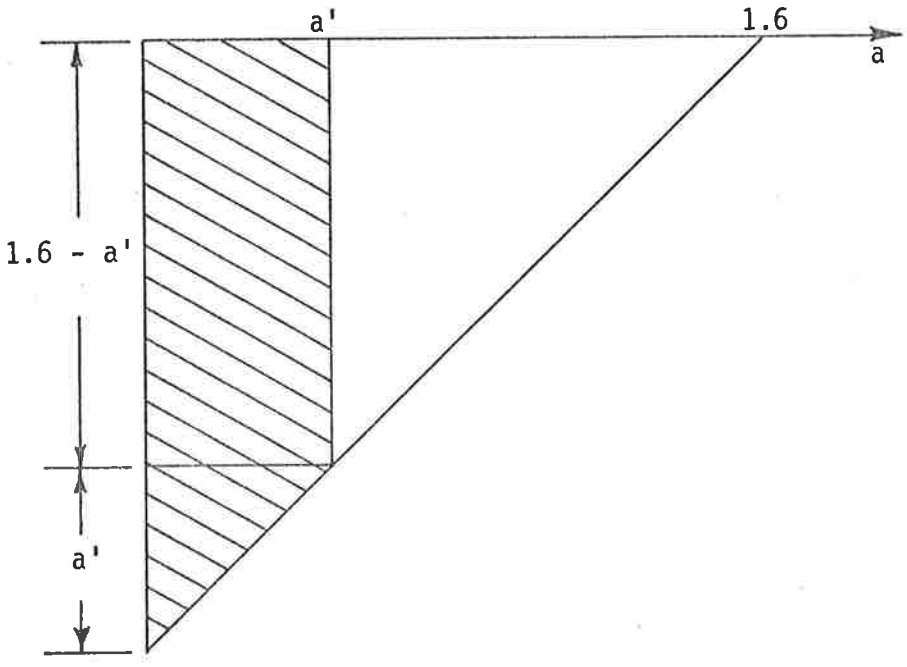
$$\frac{a'(1.6 - a') + \frac{1}{2} a'^2}{1.28} = 2\,(r_1' - 0.5) \qquad (5.1)$$

or

$$a'^2 - 3.2\,a' + 5.12\,(r_1' - 0.5) = 0 \qquad (5.2)$$



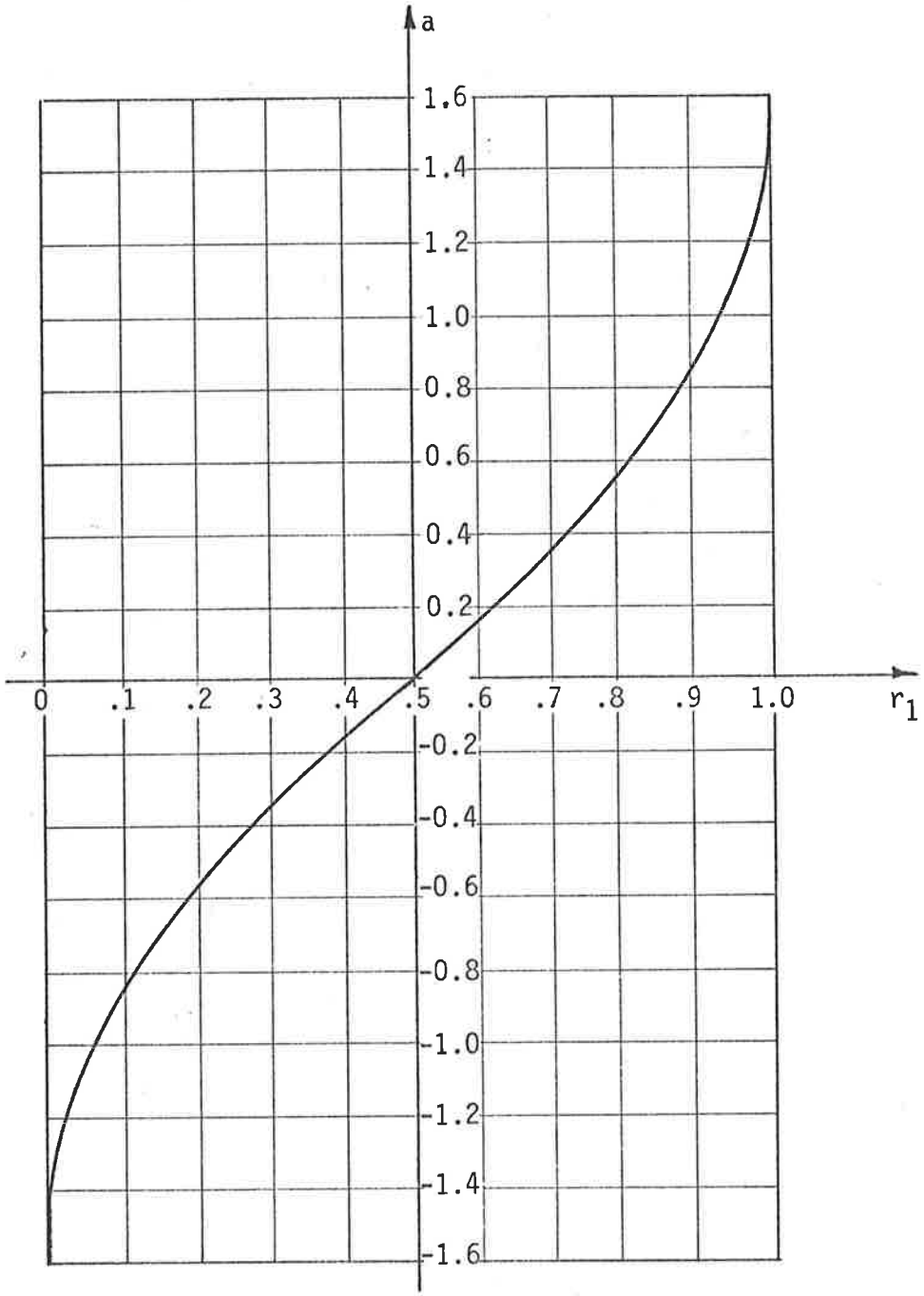
(a) Region of Allowed Starting Points (Shaded Triangle)



(b) Allowed Region for a > 0

FIGURE 5.4 Random Generation of Starting Point, Method 1

(continued next page)



(c) Relationship between 'a' Coefficient and First Random Number r_1

FIGURE 5.4 (continued) Random Generation of Starting Point, Method 1

from which

$$a' = 1.6\ (1 - \sqrt{2\ (1 - r_1')}) \tag{5.3}$$

(the alternative solution of the quadratic equation is inadmissible). By symmetry, we can deal with $r_1 < 0.5$ by replacing a with - a and $1 - r_1$ with r_1 in (5.3), and dropping the primes, giving, finally,

$$\begin{aligned} a &= 1.6\ (1 - \sqrt{2(1 - r_1)}\ ) \qquad r_1 > 0.5 \\ a &= 1.6\ (\sqrt{2r_1} - 1) \qquad\quad r_1 < 0.5 \ . \end{aligned} \tag{5.4}$$

This relationship is graphed in figure 5.4 (c).

Now that a has been found b is constrained to lie between 1.6 - |a| and 1.6 (figure 5.4 (a)) can can be generated uniformly within this range by the formula

$$b = 1.6 - r_2\ (1.6 - |a|) \ . \tag{5.5}$$

5.2.3 Generation of Random Starting Points, Method 2

The procedure of section 5.2.2 has been used when the widest variety of random starting points is desired. However, with some examples (to be discussed) it was found necessary to restrict the starting points to a smaller region of parameter space. This is to ensure convergence in a sufficiently large number of cases for the results to be meaningfully averaged.

In these cases, the N-dimensional analogue of a "rectangular region" in the space of designable parameters is used as the allowable starting region. For each parameter (say x_i) a "centre value" p_i and a "maximum deviation" q_i are specified

in terminal dialogue. A random number r_i (between 0 and 1) is generated and the value of x_i calculated from

$$x_i = p_i + 2q_i\ (r_i - 0.5) \quad . \tag{5.6}$$

5.3 Search Direction Routines (DIREC subroutine)

Various second-derivative methods and various quasi-Newton and conjugate gradient methods were tested by loading different versions of subroutine DIREC. These are distinguished by mnemonic names. A summary of these names appears in table 5.1. The essential features of these methods have been described in Chapter three; the following notes provide, for completeness, some further finer detail.

5.3.1 Second-Derivative Methods

All second-derivative methods generate the search vector $\mathbf{p}$ from the formula

$$\mathbf{A}\,\mathbf{p} = -\mathbf{g} \tag{5.7}$$

where $\mathbf{g}$ is the gradient vector of the objective function and the matrix $\mathbf{A}$ is set equal to the Hessian matrix $\mathbf{H}$ whenever that is positive definite. The differences between methods occur when $\mathbf{H}$ is not positive definite. Any of these methods may also be used with the Gauss-Newton matrix $\mathbf{R}$ in place of $\mathbf{H}$; in which case the method takes care of the few instances when $\mathbf{R}$ is indefinite due to roundoff error.

Mnemonic Name	Description
GR-S	Greenstadt with scaled matrix
GR-U	Greenstadt with unscaled matrix
MQ-S	Marquardt (line search) with scaled matrix
MQ-U	Marquardt (line search) with unscaled matrix
GM	Gill-Murray
FF	Fletcher - Freeman
DFP	Davidon-Fletcher-Powell
BFGS	Broyden-Fletcher-Goldfarb-Shanno
SR1	Symmetric Rank-1, no positive definiteness test
SR1-M	Symmetric Rank-1, Murtagh and Sargent
CG-A	Conjugate gradient, formula A
CG-B	Conjugate gradient, formula B

TABLE 5.1 Abbreviations for Methods of Search Direction Calculation

a. GR-S - Scaled version of Greenstadt algorithm - see section 3.4.7. The Hessian matrix is first scaled to have diagonal elements of unit magnitude (section 3.4.8). Cholesky factorization is attempted and if successful the factors are used to solve (5.7) for $\mathbf{p}$. Otherwise, the eigenvalues of $\mathbf{H}$ are found and each λ_i replaced with $\max\{|\lambda_i|, 10^{-10}\}$ and $\mathbf{p}$ found from (3.29). The row cyclic Jacobi method has been used to find eigenvalues and eigenvectors.

b. GR-U - Unscaled version of Greenstadt algorithm - see section 3.4.7. As for GR-S, with the pre-scaling omitted.

c. MQ-S - Scaled version of Marquardt algorithm with line search - see section 3.4.8. The Hessian matrix is scaled to have diagonal elements of unit magnitude. Cholesky factorization is attempted and if successful the factors are used to solve (5.7) for $\mathbf{p}$. Otherwise an amount β is added to all diagonal elements and factorization again attempted. This process if repeated for values of β increased by a factor of 4 each time until Cholesky factorization succeeds. The value of β used is passed to the next iteration, except that if the *first* value of β was successful, 0.25β is passed. The first value of β tried (at the first occurrence of an indefinite $\mathbf{H}$) is 0.003.

d. MQ-U - Unscaled version of Marquardt algorithm with line search - see section 3.4.8. As for MQ-S, with the pre-scaling omitted.

e. GM - The method of Gill and Murray

The method is as outlined in section 3.4.9. A direction of negative curvature is used if the Hessian is indefinite and the Euclidean norm of the gradient vector is less than 10^{-2}.

f. FF - The method of Fletcher and Freeman - see section 3.4.10. The Hessian matrix is factorized using the pivoting strategy of

Fletcher (1976). The subroutine MA29B, obtained from the Harwell subroutine library (AERE Harwell, Didcot, Berks., U.K.) has been used, but a modification was necessary to force all 2 by 2 pivots to have one eigenvalue of each sign. The "growth factor" used is 4.0.

For positive definite **H** the ordinary "Newton" direction (eq 5.7) is used. For singular **H**, the following types of search vectors are employed in alternate cases: (a) a descent direction of zero curvature and (b) a "Newton" direction restricted to the subspace of directions of positive curvature. The matrix is taken as "singular" when subroutine MA29B returns the parameter IRANK less than N. For non-singular but indefinite **H**, the following types of search vectors are employed in alternate cases:

a. a direction of negative curvature (section 3.4.10)
b. a "Newton" direction restricted to the subspace of directions of positive curvature.

5.3.2 Quasi-Newton Methods

For all quasi-Newton methods tested (see section 3.4.14) the inverse Hessian approximation **T** is carried from iteration to iteration, and it is initialized to the identity matrix. On all but the first iteration **T** is first updated (based on the results of the last iteration) and the search direction then calculated from

$$\mathbf{p} = -\mathbf{T}\,\mathbf{g} \quad , \tag{5.8}$$

both of these actions occurring within the DIREC subroutine. Versions are

(a) DFP - the Davidon-Fletcher-Powell update

(b) BFGS - the Broyden-Fletcher-Goldfarb-Shanno update

(c) SR1 - the symmetric rank one update

(d) SR1-M - the symmetric rank one update, Murtagh and Sargent (1970) version.

The difference between SR1 and SR1-M is as follows. With SR1 the update formulae (3.69) and (3.70) are used at every iteration. The inner product of the resulting search direction and the gradient $\mathbf{p}^T\mathbf{g}$ is then evaluated. If this value is greater than or equal to zero (indicating that $\mathbf{p}$ is not a descent direction) $\mathbf{p}$ is replaced by $-\mathbf{g}$ and the matrix $\mathbf{T}$ is reset to the identity matrix.

With the SR1-M algorithm the tests of Murtagh and Sargent are employed as described in section 3.4.14. That is, the scalars $\mathbf{v}^T\mathbf{y}$, $\mathbf{v}^T\mathbf{v}$ and $\mathbf{v}^T\mathbf{g}$ are evaluated at each iteration. If it happens that either

$$| \mathbf{v}^T\mathbf{y} | < 10^{-8} \quad \mathbf{v}^T\mathbf{v} \tag{5.9}$$

or

$$\frac{\mathbf{v}^T\mathbf{g}}{\mathbf{v}^T\mathbf{y}} > -10^{-8} \quad , \tag{5.10}$$

then $\mathbf{T}$ is updated according to equation (3.72) in place of (3.69). In other respects the procedure is as for SR1. If $\mathbf{p}$ fails to be a descent direction, $\mathbf{T}$ is reset to $\mathbf{I}$ and $\mathbf{p}$ is taken as $-\mathbf{g}$.

5.3.3 Conjugate Gradient Methods

Two versions of a "Fletcher-Reeves" conjugate gradient algorithm (section 3.4.13) have been tested. In each of these, the search direction $\mathbf{p}$ is calculated from (3.60), except that

$\mathbf{p}$ is reset to $-\mathbf{g}$ after every cycle of N iterations. The difference between the two versions is in the formula for the scalar β to be used in (3.60):

CG-A: β calculated from equation (3.62)

CG-B: β calculated from equation (3.61).

5.4 Line Search Routines (LINE subroutine)

The matter of the determination of a suitable steplength α along the selected direction $\mathbf{p}$ was discussed briefly in section 3.5. Various steplength algorithms have been tested (incorporated into versions of subroutine LINE) and are outlined below. Each algorithm employs some form of *search* in the one - dimensional space which results when α is considered as a variable. In the outlines which follow the following notation will be employed:

α general symbol for independent variable

$F(\alpha)$ objective function value for an arbitrary value of α, kth iteration assumed. (strictly, $F(\mathbf{x}^{(k)} + \alpha\, \mathbf{p}^{(k)})$)

α^* value of α ultimately used

α_f value of α first tried

α_t, α_u } subsequent values of α to be tried

α_s first *acceptable* value of α (i.e. first value for which a function value decrease is obtained; $F(\alpha_s) < F(0)$).

$\alpha_1, \alpha_2, \alpha_3$ three particular values of α, in increasing order.

s slope at origin ($= \frac{dF(\alpha)}{d\alpha}$ at $\alpha = 0$)

u curvature at origin ($= \frac{d^2F(\alpha)}{d\alpha^2}$ at $\alpha = 0$)

Some of the algorithms considered make use of the *slope* s (the projection of the gradient along $\mathbf{p}$ at $\alpha = 0$) to predict an initial trial steplength α_f, and are universally applicable because the gradient is known in all the methods considered. Others use also the *curvature* at $\alpha = 0$ and so are only applicable to second-derivative methods (because the curvature can be calculated only when the Hessian matrix is known). Mnemonic names given to versions of subroutine LINE are defined in table 5.2.

a. FAV-CF First acceptable value - constant factor reduction

The value of $\alpha_f = 1$ (the theoretical best value for Newton and quasi-Newton methods) is tried first. If a function value decrease is achieved, exit, otherwise successively reduce α by a factor of 0.6 until there is a decrease in value.

b. FAV-QI First acceptable value - reduction by quadratic interpolation

Try $\alpha_f = 1$. If the function value does not decrease, *predict* a new α by fitting a quadratic to the known function values at $\alpha = 0$ and $\alpha = 1$, and the slope $s = \frac{dF}{d\alpha}$ at $\alpha = 0$. The new α ($= \alpha_t$) is chosen as the point which minimizes this quadratic, or 0.2 α_f, whichever is the larger. The value of s is found from

$$s = \mathbf{g}^T \mathbf{p} \tag{5.11}$$

and the formula for α_t is then

$$\alpha_t = \max \{0.2\ \alpha_f, \frac{\frac{1}{2}\alpha_f^2 s}{F(0) - F(\alpha_f) + \alpha_f s}\} \tag{5.12}$$

If $F(\alpha_t) < F(0)$, exit. Otherwise set $\alpha_f = \alpha_t$ and predict another α_t by (5.12). Continue until the function value is less than F(0).

Mnemonic Name	Description
FAV-CF	First acceptable value, constant reduction factor
FAV-QI	First acceptable value, quadratic interpolation
AT-QI	One additional try, quadratic interpolation
PQI	Progressive quadratic interpolation
LM	Locate minimum within tolerance ε
(a) Methods ignoring curvature along the line, and taking $\alpha_f = 1$ as initial try.	

FAV-CF-C	First acceptable value, constant factor, curvature
FAV-CI-C	First acceptable value, cubic interpolation when possible, curvature
AT-CI-C	One additional try, cubic interpolation, curvature
PQI-C	Progressive quadratic interpolation, curvature
(b) Methods using curvature along the line to predict an initial try α_f.	

TABLE 5.2 Abbreviations for Line Search Methods

c. AT-QI One additional try - quadratic interpolation

Proceed as for FAV-QI, but when a function value decrease is first recorded (at say $\alpha = \alpha_s$) find the value of α (α_t) which minimizes the quadratic fitted to $F(0)$, $F(\alpha_s)$ and s, i.e.

$$\alpha_t = \frac{\frac{1}{2}\alpha_s^2 s}{F(0) - F(\alpha_s) - \alpha_s s} \tag{5.13}$$

If $\alpha_t < 0$ or $\alpha_t > 1.5\,\alpha_s$, set $\alpha_t = 1.5\,\alpha_s$. If $|\alpha_t - \alpha_s| < 0.05\,\alpha_s$, exit with $\alpha^* = \alpha_s$, because a function evaluation at α_t would be virtually duplicating work already done at $\alpha = \alpha_s$. Otherwise, evaluate $F(\alpha_t)$ and take $\alpha^* = \alpha_t$ or $\alpha^* = \alpha_s$, whichever yields the lower function value.

d. PQI Progressive interpolation with quadratics

Proceed as for AT-QI, until a value of α which gives a function reduction (α_s) is found and a prediction α_t is made. If $|\alpha_t - \alpha_s| < 0.04\,\alpha_s$ and $\alpha_s = 1.0$, exit, because the quadratic prediction has indicated that the ideal "Newton" step is close to optimum. This avoids needless function evaluations close to convergence when the method is behaving as a classical Newton-Raphson algorithm. If $|\alpha_t - \alpha_s| < 0.04\,\alpha_s$ but $\alpha_s \neq 1$, set $\alpha_t = 1.04\,\alpha_s$ or $0.96\,\alpha_s$ according to whether $\alpha_t > \alpha_s$ or $\alpha_t \leqslant \alpha_s$ respectively. Evaluate $F(\alpha_t)$. We now have the situation of figure 5.5 (a), (b), (c) or (d) depending on the relative values of α_s, α_t, $F(\alpha_s)$ and $F(\alpha_t)$. The distance $|\alpha_t - \alpha_s|$ is always at least $0.04\,\alpha_s$. A function evaluation is then made at a third point α_u, an equal distance in the direction which would appear to decrease the function. All possible outcomes are also shown in figure 5.5. We now re-index the α values as α_1, α_2 and α_3

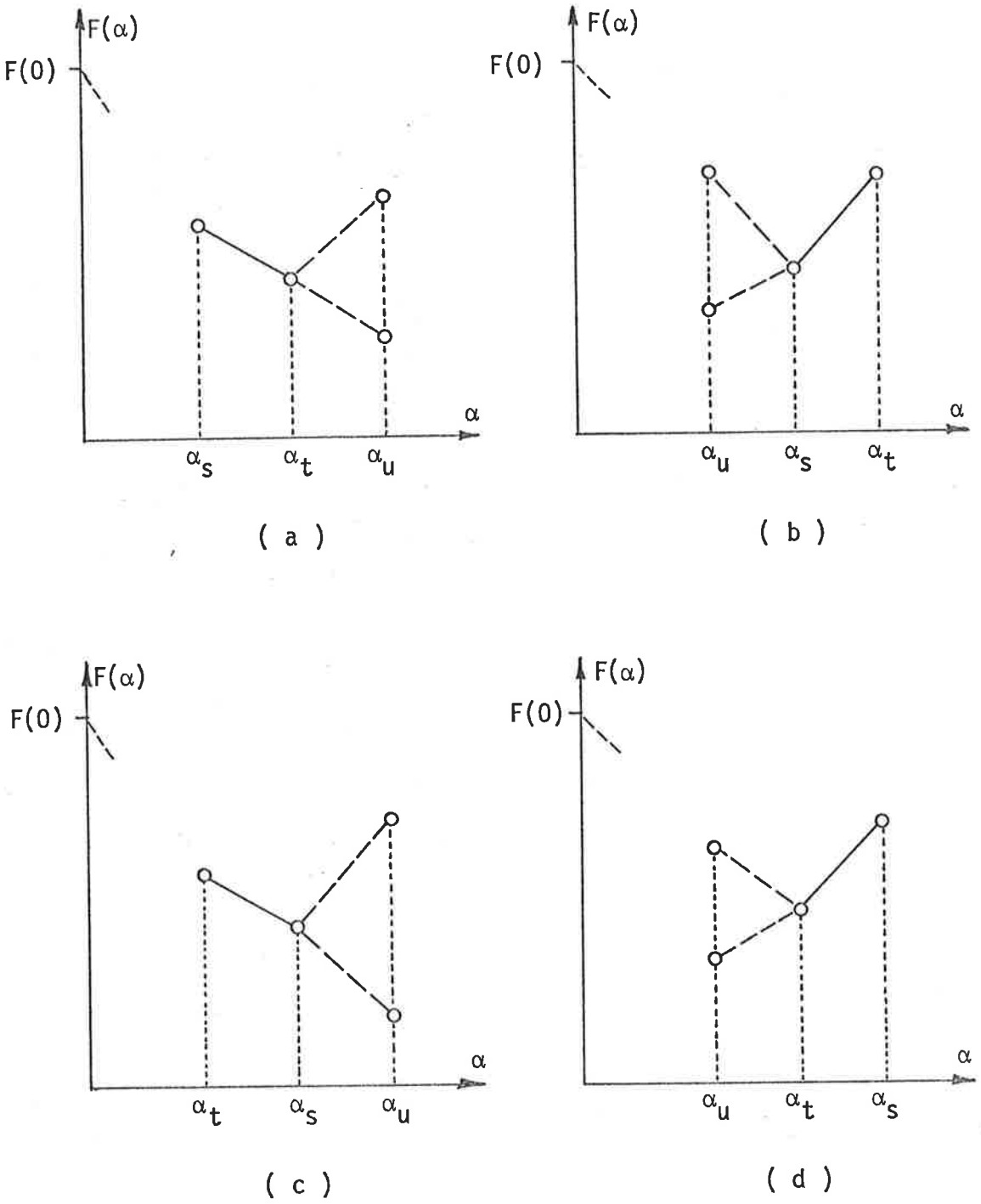


FIGURE 5.5 Possible Dispositions of First Three Trial Values of α and Corresponding Function Values, PQI Routine

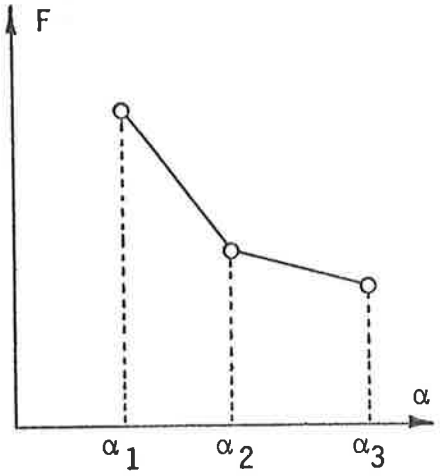
(for increasing α) and, making a further distinction on the basis of relative slopes, defined by

$$s_1 = \frac{F(\alpha_2) - F(\alpha_1)}{\alpha_2 - \alpha_1} \tag{5.14}$$

$$s_2 = \frac{F(\alpha_3) - F(\alpha_2)}{\alpha_3 - \alpha_2} \tag{5.15}$$

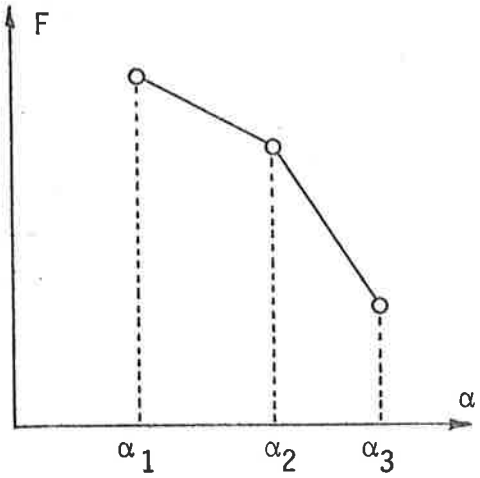
have five possibilities as shown in figure 5.6. Case (c) is the ideal, where we know that a minimum has been bracketed, and we proceed to predict that minimizing value of α by fitting a quadratic to the three points. In cases (a) and (d) a quadratic is still fitted, but the minimizing value of α may be outside the interval (α_1, α_3). If it does, the interval of interest is moved in the appropriate direction by making a further function evaluation at the predicted minimum (or at a default point $2.7\,\alpha_3 - 1.7\,\alpha_2$ (upward) or $2.7\,\alpha_1 - 1.7\,\alpha_2$ (downward) in cases when the predicted minimum is further away). In cases (b) and (e) a fitted quadratic would have a maximum instead of a minimum, and is of no use. In these cases a further evaluation is made at $2\alpha_3 - \alpha_2$ (case (b)) or $2\alpha_1 - \alpha_2$ (case (e)) to re-define the three point interval.

The process of interval re-definition (deleting one point each time) is continued until the predicted minimum of the quadratic is within the three point interval (or "just outside" - that is, coinciding with α_1 or α_3 to within 2% of α_s). A final function evaluation is then made at the predicted point except when it coincides with α_1, α_2 or α_3 to within 2% of α_s. The best value of α tried during the entire process is taken as α^*.



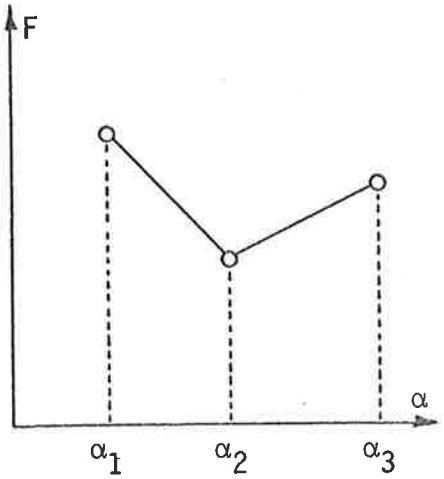
(a) $F(\alpha_1) > F(\alpha_2) > F(\alpha_3)$

$|s_1| > |s_2|$



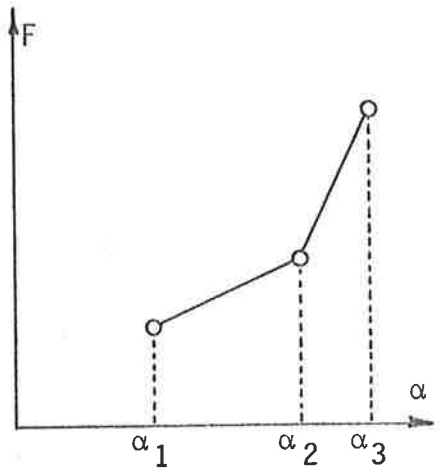
(b) $F(\alpha_1) > F(\alpha_2) > F(\alpha_3)$

$|s_1| \leq |s_2|$



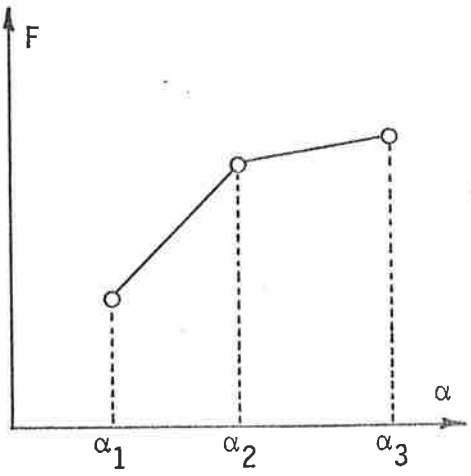
(c) $F(\alpha_1) > F(\alpha_2)$ and $F(\alpha_2) \leq F(\alpha_3)$

OR $F(\alpha_1) \geq F(\alpha_2)$ and $F(\alpha_2) < F(\alpha_3)$



(d) $F(\alpha_1) < F(\alpha_2) < F(\alpha_3)$

$|s_1| < |s_2|$



(e) $F(\alpha_1) < F(\alpha_2) < F(\alpha_3)$

$|s_1| \geq |s_2|$

FIGURE 5.6 Possible Dispositions of Three Adjacent Function Evaluations, PQI Routine

To recapitulate, the basic philosophy of the PQI line search routine is that a three-point interval is first established and then slid along the α axis (by deleting the upper or lower point at each step) until the true function minimum in the one-dimensional (α) space is either "bracketed" or predicted to be "very close" to one of the interval bounds. A quadratic is fitted to the three known points (when possible) to aid in defining useful bounds for the new interval. However, safeguards are employed to avoid using severe extrapolations, which are likely to be unreliable. Further safeguards are incorporated to avoid evaluation of the function at very closely neighboring points, which would represent wasteful duplication.

e. LM - Locate minimum within tolerance ε.

A routine has been written which allows the position of the one-dimensional minimum to be found to any desired tolerance. It will not be described here in complete detail because the experimental work has indicated that such accurate line searches are never advantageous.

The LM routine proceeds as for PQI up to the stage that a function value reduction is achieved and a three-point interval established (except that exit is *not* made if "Newton-type" convergence seems likely). The three-point interval is (if necessary) then slid in the direction of decreasing function value until a bracket is established (that is, the function has been evaluated at three points and the middle one has the lowest value). This procedure is different from PQI in that quadratic extrapolations are not used to define the new highest or lowest point of the interval. Rather, the new evaluation is made at a

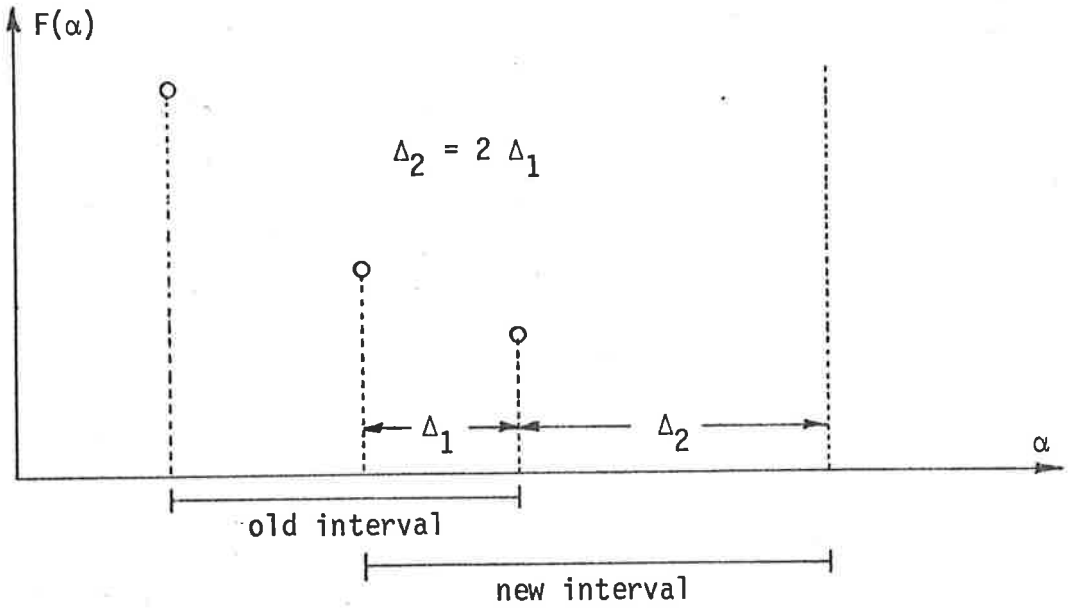
"reasonable" distance from the previous extreme point. An "acceleration factor" is used to ensure that the size of the interval grows with successive steps, thus avoiding the situation where progress could consist of many tiny steps, with a great waste of computer time. The procedure is illustrated in figure 5.7. A larger acceleration factor (2.0) is used in the upward direction than in the downward direction (1.5).

When a bracket has been established, quadratic interpolation is used to predict the position of the minimum, at which point another function evaluation is made. One of the extreme points of the interval is then discarded, resulting in a new three-point interval which is narrower than the original and still brackets the minimum. The process of interpolation and interval refinement is repeated until the interval width is less than a specified tolerance ε.

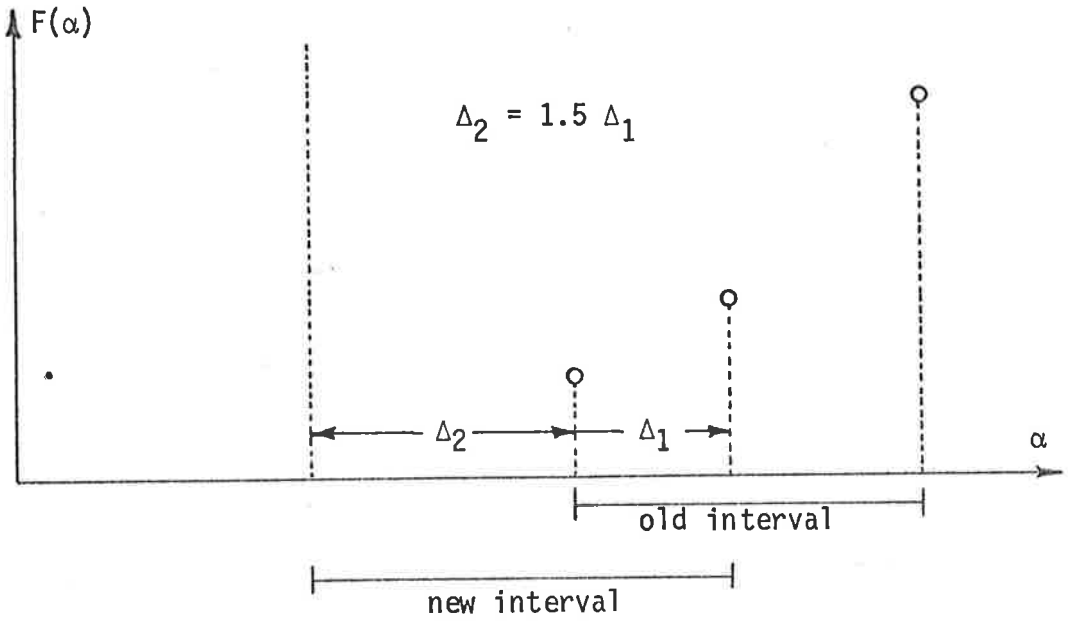
During the expansion phase tests are made to ensure that the interval does not grow downwards past $\alpha = 0$ or upwards past any previously-tried value. In such a case, the appropriate limiting value would be used instead (a bracket would necessarily be established), and the program would proceed to the refinement phase.

f. FAV-CF-C First acceptable value - constant factor reduction - curvature considered

The method is as for FAV-CF in that successive reductions of α are by a constant factor (0.6) and the first function value decrease is accepted. However, the initial value α_f tried is not necessarily unity but is based on the slope s and curvature u at $\alpha = 0$. The chief motivation for this and other methods employing curvature is that it may be beneficial to the Fletcher-Freeman algorithm in which negative-curvature search directions are especially frequent. Methods (such as FAV-QI,



(a) Upward Sliding



(b) Downward Sliding

FIGURE 5.7 Interval Sliding to Obtain Bracket on Minimum, LM Routine

AT-QI and PQI) employing a quadratic fit at $\alpha = 0$ are at a theoretical disadvantage because such a quadratic has positive curvature and is unlikely to be a good model when $u < 0$. The strategy follows Fletcher and Freeman's (1977) outline, but the mathematics was derived independently. The value of s ($= \frac{dF}{d\alpha}$ at $\alpha = 0$) is calculated from (5.11) and the curvature u ($= \frac{d^2F}{d\alpha^2}$ at $\alpha = 0$) from

$$u = \mathbf{p}^T \mathbf{H}\ \mathbf{p} \tag{5.16}$$

The action taken by FAV-CF-C to determine the trial value α_f depends upon the manner in which $\mathbf{p}$ was calculated in the DIREC subroutine, as follows:

(1) If $\mathbf{H}$ is positive definite, and $\mathbf{p}$ accordingly found using $\mathbf{H}\ \mathbf{p} = -\mathbf{g}$, then $\alpha_f = 1.0$. (This value is identical to that which would be predicted by fitting a quadratic to F(0), s and u, for such a quadratic is

$$f(\alpha) = \tfrac{1}{2} u\alpha^2 + s\alpha + F(0) \tag{5.17}$$

which has its stationary value at $\alpha = -s/u$. Substituting $\mathbf{H}\ \mathbf{p} = -\mathbf{g}$ into (5.16) gives $u = -\mathbf{p}^T \mathbf{g} = -s$).

(2) If $\mathbf{H}$ is indefinite and $\mathbf{p}$ is found as a "Newton" direction in the subspace of directions of postive curvature (FF only), then again $\alpha_f = -s/u = 1.0$.

(3) If $\mathbf{p}$ is found using FF, as a negative curvature direction, the value α_f is found (as described by Fletcher and Freeman)

by fitting a *quartic* polynomial according to the following criteria:

(i) the function value, slope and curvature of the quartic are equal to the known values for the function itself at $\alpha = 0$.

(ii) the quartic attains a maximum at $\alpha = -s/u$ (which is < 0), and

(iii) the quartic attains a minimum for some $\alpha = \alpha_f$ (> 0) and the value of this minimum is less than $F(0)$ by a pre-assigned "target" amount T.

This is shown in figure 5.8. The target is initially set to half of the function value, and revised after every Newton (positive curvature) step to equal the latest reduction achieved or half the current function value, whichever is the smaller.

The quartic polynomial only has the form shown in figure 5.8 if

$$T > -\,{}^{11}/_{12}\, s^2/u \tag{5.18}$$

and in such a case the value of α_f is given by

$$\alpha_f = \sqrt{{}^{16}/_{9}\,(s/u)^2 - 4\,(T/u)} \; - {}^{4}/_{3}\,(s/u) \tag{5.19}$$

If (5.18) is not satisfied, then $\alpha_f = s/u$ is used instead.

(4) If $\mathbf{p}$ is generated by FF as a descent direction of *zero* curvature, a *cubic* is fitted such that the function value, slope and curvature are matched at $\alpha = 0$ and the specified target function reduction is achieved by the cubic, giving the formula:

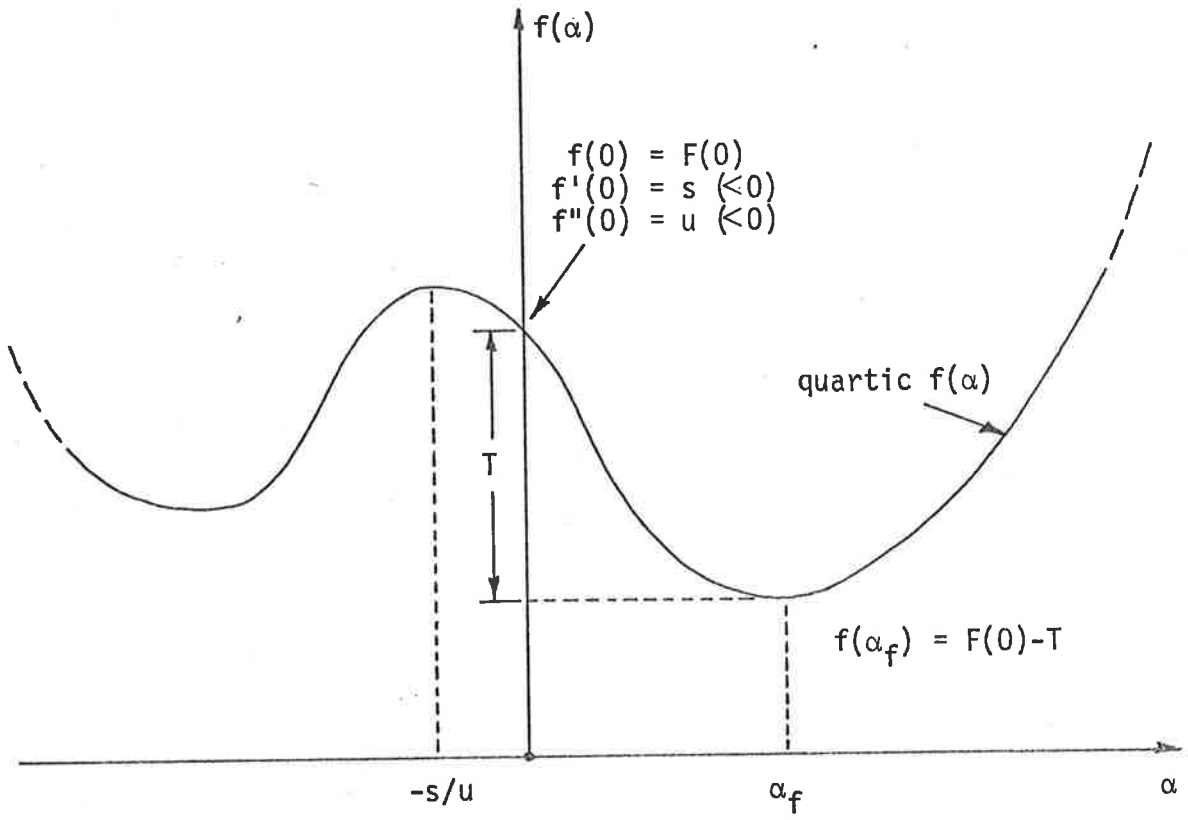


FIGURE 5.8 Quartic Fit to Predict α_f, Negative Curvature Case

$$\alpha_f = - 1.5\ T/s \qquad (5.20)$$

(5) If any DIREC subroutine other than FF (the Fletcher-Freeman method)is used and **H** is not positive definite, the sign of the curvature u is not known in advance, but it may still be found using (5.16). Methods similar to those in (3) and (4) above are used, or if $u > 0$, α_f is taken as $- s/u$. However, in contrast to the FF method, the methods of Greenstadt, Marquardt and Gill-Murray types calculate the search direction **p** from the equation

$$\mathbf{A\,p} = - \mathbf{g} \qquad (5.7)$$

where the matrix **A** is still related to the Hessian **H** , even when **H** is indefinite. In particular, the components of **p** are scaled such that it is expected that the optimum value of α will not be vastly different from unity (with FF there is no such indication). Since there is no theoretical reason to believe that the value α_f predicted by "target function reduction" is especially reliable, it is only used when it is less than 2.0. Otherwise, the default value $\alpha_f = 2.0$ is used instead.

g. FAV-CI-C - First acceptable value - Reduction by cubic interpolation, curvature considered.

Initial α_f selection is as for FAV-CF-C . The function is evaluated at $\alpha = \alpha_f$, and if the function value $F(\alpha_f)$ is not less than $F(0)$, an attempt is made to predict a good point for further evaluation by fitting a *cubic* polynomial to the function value, slope *and curvature* at $\alpha = 0$ and the known function value at $\alpha = \alpha_f$. The equation of this cubic is

$$f(\alpha) = F(0) + s\alpha + \tfrac{1}{2} u\alpha^2 + d\alpha^3 \qquad (5.21)$$

where the formula for d is

$$d = \frac{F(\alpha_f) - F(0) - s\alpha_f - \tfrac{1}{2} u\alpha_f^2}{\alpha_f^3} \qquad (5.22)$$

and the value of α which corresponds to the minimizing stationary point of the cubic (α_t) is given by

$$\alpha_t = \frac{\sqrt{u^2 - 12\, sd} - u}{6d} \qquad (5.23)$$

The value of α_t cannot be calculated from (5.23) when $d \approx 0$, and this is the case when the parameter vector is close to convergence and the function behaves very much like a quadratic. Hence, if the numerator in (5.22) is less in absolute value than $|0.01\, s\alpha_f|$, equation (5.23) is not used - rather, a quadratic interpolation (as for FAV-QI) is made, ignoring the curvature, As with FAV-QI, if $\alpha_t < 0.2\, \alpha_f$, $0.2\, \alpha_f$ is used instead. The interpolation process is continued until a function value decrease is achieved.

h. AT-CI-C - One additional try - cubic interpolation - curvature considered

An initial function value decrease is obtained (at $\alpha = \alpha_s$) as for FAV-CI-C. Then, using α_s rather than α_f, (5.22) is evaluated and a prediction α_t made from (5.23) if d is large enough (see above) or from (5.13) otherwise. If $|\alpha_t - \alpha_s| < 0.04\, \alpha_s$

no extra evaluation is made but otherwise $F(\alpha_t)$ is found and α^* set equal to α_s or α_t, whichever yields the lower value.

i. PQI-C - Progressive quadratic interpolation - initial interval by cubic interpolation - curvature considered

α_s and α_t are found as for AT-CI-C. The minimum function value along $\mathbf{p}$ is then localized using quadratic interpolation as for PQI.

5.5 General Abbreviations

In reporting on the performance of the various minimization methods tested it is convenient to refer to them by shorthand names. Tables 5.1 and 5.2 have summarized such mnemonics for search direction and steplength algorithms. In what follows, some additional abbreviations will also be used. These are listed in Table 5.3.

5.6 Further Considerations in the Application of Minimization Algorithms

5.6.1 Symmetry of Matrices

All matrices involved in the minimization algorithms are symmetric. Accordingly, it was decided at the outset that only the upper triangle would be calculated and stored. Subroutines for matrix-vector multiplication, Cholesky factorization, and so on, were specially written to handle matrices in this form. In this way some economy of storage and savings in computer time were achieved, at the cost of slightly more complex subscript calculations. In particular, the subroutines which calculate the Hessian and Gauss-Newton matrices are believed to be particularly efficient (considering that they are written in a high-level language) partly for this reason.

Mnemonic	Description
GN	Gauss-Newton methods (in general)
QN	Quasi-Newton (matrix update) methods (in general)
CG	Conjugate gradient methods (in general)
SD	Second derivative methods (in general)
TM	True Marquardt (no line search) methods (in general)
TM-U-N	True Marquardt (no line search), unscaled matrix, full Hessian (Newton) version
TM-S-N	True Marquardt (no line search), scaled matrix, full Hessian (Newton) version
TM-U-G	True Marquardt (no line search), unscaled matrix, Gauss-Newton version
TM-S-G	True Marquardt (no line search), scaled matrix, Gauss-Newton version

TABLE 5.3 Miscellaneous Abbreviations used for Optimization Methods

5.6.2 Implications of Root Reflection and Pairing

As discussed in Section 4.9, many sub-optimal local minima may be eliminated from consideration by allowing root reflection and pairing during the application of a minimization process. In the case of an SD or a GN algorithm this process is unlikely to have any deleterious effect on the convergence of the algorithm itself because the **H** or **R** matrix is calculated afresh at each interation. However, with other methods, information is carried from one iteration to the next (in a matrix with QN, and in a vector with CG methods). If root reflection or a change in the pairing of real roots occurs, the correspondence of the elements of the parameter vector with the actual filter coefficients is changed, and so this information becomes useless, and the process must be re-started with a steepest descent step. If such re-starts are too frequent the algorithm will not have a chance to achieve its theoretical convergence performance but will tend to behave like a steepest descent method. There is thus the possibility of improving performance by allowing reflection and pairing only after every N_p iterations, where N_p is some small integer. The matter is considered later in conjunction with specific examples.

5.6.3 Special Line Search Methods for Conjugate - Gradient Algorithms

There is a special problem in the implementation of the line minimization, or steplength determination, with CG methods. This is that the formulae used provide no indication of the order of magnitude of the steplength α to be used at each iteration. This is to be contrasted with the situation with SD, GN and QN methods where it is known that the ideal steplength is unity

close to convergence. Moreover $\alpha = 1$ will usually be of the right order with SD and GN methods even in regions remote from convergence (although with QN methods this need not be so following the resetting of the matrix to $\mathbf{I}$).

If one of the "ordinary" line search routines of Section 5.4 is used with a CG method (or a QN method following a reset) and $\alpha = 1$ happens to be far too large, then many function evaluations will be wasted as α is gradually reduced in the search for an "acceptable" value. On the other hand, if $\alpha = 1$ is too small, and a FAV or AT search is used, a step far shorter than the optimum would be used resulting in a large increase in the number of iterations. The PQI and LM searches would find the correct steplength but would employ an excessive number of evaluations in doing so.

Accordingly, a further set of LINE subroutines were written, identical with FAV-CF, FAV-QI, AT-QI and PQI except that the "first acceptable value" α_s is passed to the next iteration to be used as the first trial. If, however, the *first* trial is "acceptable" on any iteration, then 1.6 times this value is passed on, providing a mechanism whereby α can "grow outwards" as well as shrink. These subroutines are referred to as the "α pass" versions (in contrast to "$\alpha = 1$" versions).

Tests with QN methods have shown that "α pass" seems always to be inferior to "$\alpha = 1$", any benefit following matrix resets apparently being outweighted by the appropriateness of $\alpha = 1$ when the $\mathbf{T}$ matrix has become a good approximation to the inverse Hessian. Accordingly, in the results to be reported, "$\alpha = 1$" is always used with QN methods.

With CG methods, "α pass" is sometimes better and sometimes not. The matter is further discussed in conjunction with specific

examples.

5.7 The Examples, and Methods of Assessment

Six examples related to digital filter design have been treated in a considerable amount of detail. Many different gradient-based optimization algorithms have been applied to each problem. Although some general aspects of filter design by optimization are mentioned in the preliminary discussion of each example, the relative performance of the algorithms is the primary concern here.

Algorithms are criticised firstly in terms of reliability. This is largely a qualitative matter and comments appear both in the sections relating to individual examples, and in Section 5.14 (dealing with overall impressions).

The second criterion of performance is the amount of computing effort. In most cases this is measured as the time taken for the evaluation of the objective function and its derivatives only, with all overheads (such as matrix algebra) ignored. This course was adopted because:

a. it seems, even in problems with only six variables, that function evaluation time *does* dominate the total computer time. In the more realistic problems (of 12, 14 and 20 variables) this is an even better assumption.

b. the general purpose test program performs many tasks besides the optimization itself, such as generation of random starting points and collection of statistics. This, and the fact that there were often runs which failed to converge, makes the "total execution time" an unrealiable measure.

In some of the mathematical literature relating to comparisons of optimization methods (see section 3.6) it has been assumed that the evaluation of the gradient of a function of N variables requires N times the effort of a function evaluation above. The same reasoning, presumably, would assign a factor of $\frac{1}{2}N\ (N + 1)$ to a Hessian evaluation, this being the number of independent elements of $\mathbf{H}$. With general algebraically-defined functions in which the variable appear in "random" combinations, this may be approximately true, but even then the likely occurrence of common sub-expressions in the formulae for the derivatives is ignored. With the very "regular" objective functions of the digital filter design problems, gradient and Hessian evaluation can be performed far more efficiently than this.

In the work which follows, the speed comparisons rest on experimentally-determined execution times for F alone, for F plus $\mathbf{g}$, F plus $\mathbf{g}$ plus $\mathbf{R}$, and for F plus $\mathbf{g}$ plus $\mathbf{H}$. Table 5.4 lists the main characteristics of the examples and also gives these experimental times. The time for evaluation of function alone is quoted in milliseconds; that for the various classes of derivative evaluation is expressed as a factor to be applied to this base figure. For each method and each example the "total labour" is computed by applying the appropriate factor from this table, and then reduced to a figure for "relative labour", which is the total labour expressed relative to that required by the MQ-U method with AT-QI line search, for the same example. This particular method is taken as the datum because in all examples it is the best, or close to the best, method tested.

Example	Type of Response Optimized	Number of Variables	"Function only" Evaluation Time (ms)	Evaluation time for derivatives (relative to function only)		
				F, **g**	F, **g**, **R**	F, **g**, **H**
A	Log magnitude	6	3.28	2.61	4.45	5.61
B	Log magnitude	12	9.35	3.13	6.86	8.42
C	Group delay (all-pass equalizer)	6	7.54	3.61	4.66	6.70
D	Simultaneous magnitude and group delay	20	42.3	4.09	9.54	20.8
E	Impulse response	6	4.15	2.28	5.19	7.58
F	Impulse response	14	24.0	2.42	6.38	10.6

TABLE 5.4 Details of Test Examples and Function Evaluation Times

5.8 Example A - Log Magnitude Response

5.8.1 General

The target log magnitude response is specified at 15 frequencies as detailed in Table 5.5. All weight factors are unity. The digital filter has one second-order feed-forward (numerator) section and two second-order feedback (denominator) sections. There are thus two zeros and four poles. All six coefficients are variable, i.e. N = 6. Details of the optimum least squares solution are given in Table 5.6, and the achieved response is plotted (together with the target points) in figure 5.9.

Extensive tests were made of a variety of gradient-based optimization methods for obtaining this solution. The problem is too simple to be really representative of those met in practice, but it served usefully to validate most features of the general test program. For each method tested, thirty random starting points were used, generated in the manner described in Section 5.2.2. Convergence was deemed to have occurred when (a) all components of the gradient were less than 10^{-3}, and (b) all components of the last $\mathbf{p}$ (change) vector were less than 10^{-3}.

5.8.2 Second-Derivative Methods

The results obtained for the cases where a line search is used and the initial trial value of α is unity are summarized in Tables 5.7 to 5.10. Convergence to the optimal solution is obtained from every starting point in every case, and it is believed that there is only this one stable, minimum phase solution yielding a local minimum of the objective function in

digital frequency (angle) θ (rad)	log magnitude (dB)
.2	0
.4	3
.6	6
.8	9
1.0	15
1.2	18.5
1.4	20
1.6	17.5
1.8	15.5
2.0	17.5
2.2	15
2.4	12
2.6	9
2.8	6
3.0	3

TABLE 5.5 Specification for Example A

Section Number	Num. or Den.	Coefficients	Comments
1	N	a = 0.057164 b = -0.368090	Two real zeros, -0.635959, 0.578795
2	D	c = 0.770169 d = 0.444873	Complex conjugate pole pair z = 0.666988 exp (±j 2.186274)
3	D	c = -0.507832 d = 0.627021	Complex conjugate pole pair z = 0.791847 exp (±j 1.244367)

Minimum objective function value = 2.790295×10^{-1}

TABLE 5.6 Solution for Example A

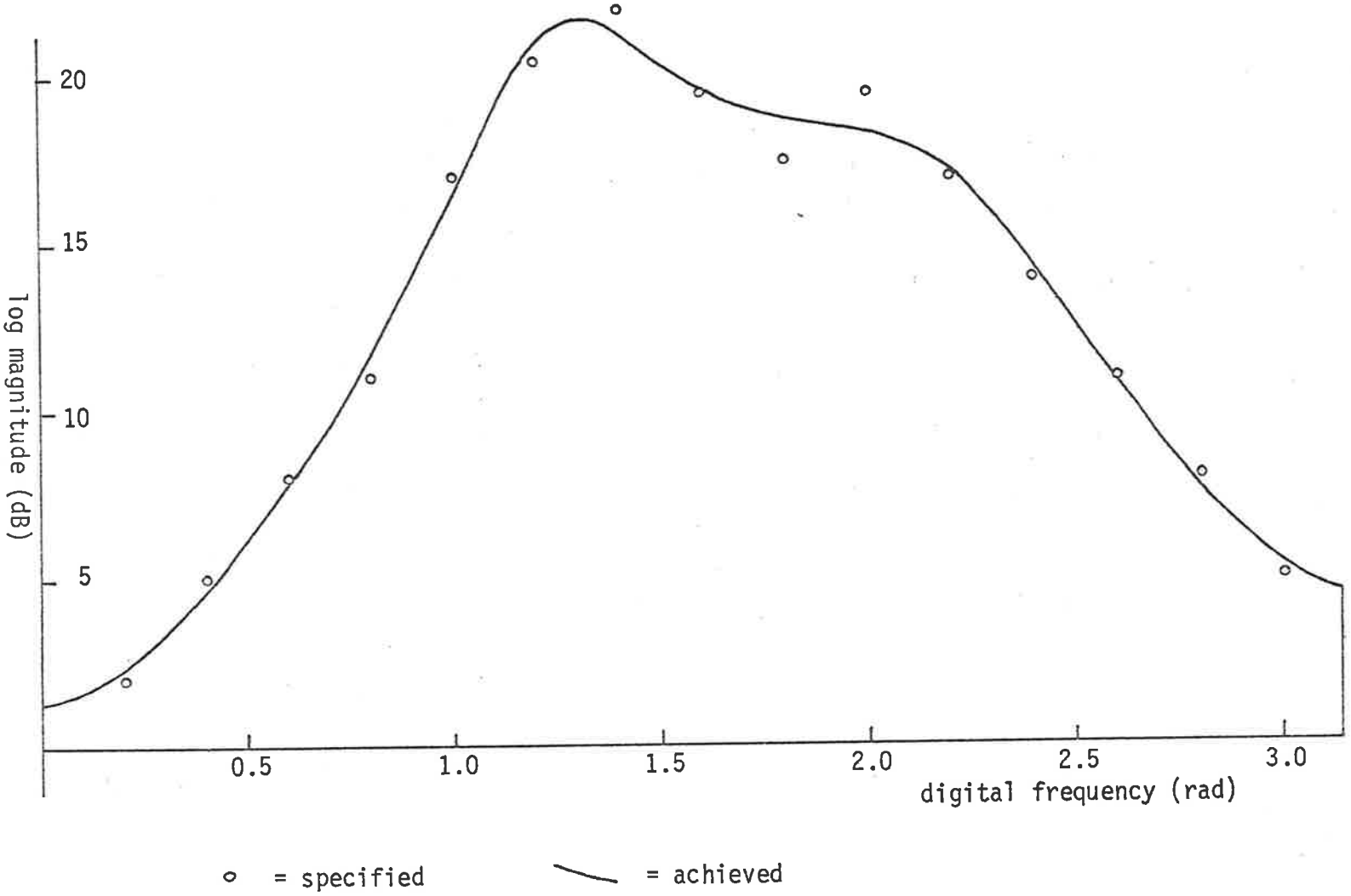


FIGURE 5.9 Example A. Log Magnitude Response

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
GR-S	11.8	11.4	9.5	8.6	8.5	8.4	8.4
GR-U	11.5	11.5	9.1	8.6	8.5	8.3	8.3
MQ-S	10.5	10.9	9.5	8.9	8.7	9.0	8.9
MQ-U	9.9	10.2	8.9	8.0	8.0	8.1	8.1
GM	14.4	14.3	11.3	10.8	10.5	10.1	9.6
FF	12.6	13.2	11.1	10.6	10.2	10.0	10.4

TABLE 5.7 Example A. Second-derivative methods

Average Number of Iterations

Note: For an explanation of abbreviations used, see Tables 5.1 (page 215)and 5.2 (page 221)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD						
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
GR-S	17.9	15.5	19.1	27.5	38.0	49.5	54.2
GR-U	16.3	16.4	17.9	27.7	38.9	47.9	53.4
MQ-S	13.9	12.8	17.2	32.4	45.4	61.1	63.4
MQ-U	12.3	11.6	15.6	27.7	38.0	49.0	53.4
GM	41.4	29.1	28.8	43.9	54.4	65.3	72.1
FF	24.0	19.1	23.1	43.2	51.3	62.4	75.0

TABLE 5.8 Example A. Second-derivative methods

Average Number of Auxiliary Function Evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
GR-S	1.27	1.20	1.09	1.15	1.30	1.47	1.53
GR-U	1.23	1.23	1.05	1.15	1.32	1.42	1.52
MQ-S	1.11	1.12	1.06	1.24	1.42	1.70	1.71
MQ-U	1.03	1.05	1.00	1.11	1.26	1.42	1.50
GM	1.85	1.65	1.39	1.58	1.71	1.85	1.91
FF	1.44	1.41	1.29	1.56	1.65	1.80	2.02

TABLE 5.9 Example A. Second-derivative methods

Relative Labour (CP time) Spent in Function Evaluation

(Best method = 1.00)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
GR-S	6.4	5.9	4.5	3.9	3.8	3.7	3.7
GR-U	6.5	6.0	4.4	3.7	3.5	3.6	3.5
MQ-S	5.8	5.8	5.0	4.3	4.3	4.4	4.3
MQ-U	4.9	4.9	4.1	3.4	3.5	3.4	3.4
GM	9.2	8.7	6.0	5.6	5.3	5.3	4.9
FF	7.3	7.7	6.1	5.5	5.2	5.0	5.3

TABLE 5.10 Example A. Second-derivative methods

Average Number of "Pre-Newton" Iterations

this problem.

In the tables the line search methods are ordered from left to right in terms of increasing amount of effort spent in locating the minimum along the line. As expected, the average number of iterations required decreases as the accuracy of the line search improves (Table 5.7). However, there is virtually no improvement in going from PQI to LM or from any reduction in the parameter ϵ in the LM algorithm. Moreover, PQI offers only a slight improvement over AT-QI. However, a significant benefit is gained from one additional quadratic interpolation after the first function reduction is achieved.(AT-QI versus FAV-QI). The FAV-CF algorithm seems virtually as good as FAV-QI, although the reduction factor used (0.6) is merely a guess as to an appropriate value.

The number of iterations is not of course an adequate measure of the relative efficiency of the methods because accurate line searches require many more "auxiliary" function evaluations (figures for this example appear in Table 5.8, being averages per starting point over the 30 starting points). When this is taken into account, the improvement offered by AT-QI over FAV-QI is not as spectacular, although it is still just worthwhile. PQI becomes worse than AT-QI, and LM is quite uncompetitive. Table 5.9 rates the algorithms in terms of the total CP time expended in evaluating the function and its derivatives (relative to the best algorithm tested, given a value of 1.00).

On the basis of this example it is not possible to justify the added complexity of PQI (or LM). The slighly simpler FAV algorithms are very nearly as good as the best line search algorithm, AT-QI. These conclusions are independent of the method used to determine a search direction.

Coming now to the matter of search direction algorithms, Table 5.9 indicates that the substantially more complex GM and FF methods perform significantly worse than Marquardt and Greenstadt methods and are not therefore worthwhile. This would seem to be due to some inferiority of these methods of search direction determination themselves (rather than some subtle interaction with the line search method) because they are also inferior in terms of number of iterations, whatever line search is used (table 5.7). Since all second-derivative methods use the same direction when the Hessian matrix is positive definite, it is more illuminating to consider just the "pre-Newton" iterations, i.e. those iterations *before* **H** becomes positive definite and remains so. These numbers are tabulated in Table 5.10. In particular, the conjecture of Fletcher and Freeman (1977) that their method should force the search into the domain of positive definiteness sooner (because it uses descent directions of negative curvature) is not borne out by this digital filter example.

In terms of any of the measures used thus far there is little to choose between the Greenstadt and Marquardt methods. However, Greenstadt requires a complete eigensystem analysis at every iteration for which **H** is indefinite, and this substantially higher "overhead" leads to a decided preference for Marquardt. Using the AT-QI line search, function, gradient and Hessian evaluation accounts for 63% and 66% of the total computer time for MQ-S and MQ-U methods respectively, but only 47% for GR-S and GR-U.

MQ-U performs slightly but consistently better than MQ-S. Thus it seems that the characteristic of the search direction to approach the steepest descent direction as the Marquardt para-

meter becomes large (in the unscaled case) has a beneficial effect outweighing any improved numerical conditioning due to scaling. This is not surprising considering that the dimensionality of the problem is only 6, and the computations are being performed in 60-bit floating point.

The matter of the use of curvature information to aid the line search is next considered. Table 5.11 lists the relative amounts of labour (CP time) for each line search method employing curvature. These figures may be directly compared with those of Table 5.9. Table 5.11 shows that again the AT line searches are better than the less complex FAV and more complex PQI versions. FAV-CI-C shows some superiority over FAV-CF-C, which was not evident in the figures for FAV-QI vs. FAV-CF. Thus it seems that curvature information is of some use in predicting an acceptable value of α once an initial trial has been made.

Table 5.12 is a rearrangement of certain of the data of Tables 5.9 and 5.11 designed to display any significant differences in performance between curvature and non-curvature algorithms. Section (a) of this table seems to indicate that curvature information is not a particularly useful aid in predicting an initial trial value for α. In particular, its use does not aid the FF algorithm, even though the scheme used is that suggested by Fletcher and Freeman (1977) in describing their method. However, section (b) of Table 5.12 shows that the *cubic* interpolation made possible when the curvature at $\alpha = 0$ is known generally provides a better prediction than *quadratic* interpolation after one trial evaluation has already been made (as was also mentioned above). This probably

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	1.26	1.21	1.00	1.17
GR-U	1.30	1.15	0.98	1.11
MQ-S	1.12	1.09	1.02	1.20
MQ-U	1.00	0.95	0.91	1.06
GM	1.70	1.53	1.29	1.42
FF	1.53	1.47	1.32	1.42

TABLE 5.11 Example A. Second derivative methods utilizing curvature

Relative Labour (CP Time) Spent in Function Evaluation

	GR-S	GR-U	MQ-S	MQ-U	GM	FF
FAV-CF	1.27	1.23	1.11	1.03	1.85	1.44
FAV-CF-C	1.26	1.30	1.12	1.00	1.70	1.53
(a) FAV searches, constant factor						
FAV-QI	1.20	1.23	1.12	1.05	1.65	1.41
FAV-CI-C	1.21	1.15	1.09	0.95	1.53	1.47
(b) FAV searches, interpolation						
AT-QI	1.09	1.05	1.06	1.00	1.39	1.29
AT-CI- C	1.00	0.98	1.02	0.91	1.29	1.32
(c) AT searches						
PQI	1.15	1.15	1.24	1.11	1.58	1.56
PQI-C	1.17	1.11	1.20	1.06	1.42	1.42
(d) PQI searches						

TABLE 5.12 Example A. Relative Labour (CP Time) Spent in Function Evaluation

Curvature vs. Non-curvature Methods

holds also for the interpolation for an additional try (Table 5.12 section (c)). Curvature information is not used by the PQI-C algorithm beyond the stage of the "first additional try" so that there are no additional conclusions to be drawn from Table 5.12 section (d).

An alternative procedure, that of ignoring curvature and using the "α pass" searches of Section 5.6.3, was also investigated. In no case did this produce better results than simply taking $\alpha_s = 1.0$, either in terms of number of iterations or total labour. For example, using the AT-QI line search, performance degradations in terms of total labour were:

for MQ-U	1.00 to 1.17
for GM	1.39 to 2.00
for FF	1.29 to 1.70

Two other second-derivative algorithms were tested, namely, the true Marquardt with scaled and unscaled matrices, TM-S-N and TM-U-N. Results are summarized in Table 5.13. The unscaled algorithm is slightly better. Comparison with the tables for line search algorithms shows that each TM method is virtually identical in performance to the corresponding FAV line search Marquardt method, and thus slightly worse than the AT search. Use of a true Marquardt method avoids the need for a line search, but because (on the average) more sets of linear equations must be solved, the edge in efficiency lies with the line search algorithm.

	TM-S-N	TM-U-N	Compare with table
Average number of iterations	11.0	10.1	5.7
Average number of pre-Newton iterations	6.2	4.8	5.10
Average number of auxiliary function evaluations	12.6	11.8	5.8
Average labour (CP time)	1.12	1.03	5.9

TABLE 5.13 Example A. Results for True Marquardt Second-derivative algorithms

5.8.3 Gauss-Newton Methods

The term Gauss-Newton (GN) is applied in this chapter to any method using the $\mathbf{R}$ ($2\ \mathbf{J}^T\mathbf{J}$) matrix in place of the Hessian $\mathbf{H}$. Methods tested included both true Marquardt and line search methods. Although the matrix $\mathbf{R}$ is theoretically always positive definite (Section 3.4.12), some means is required to guarantee a descent direction in those cases when it becomes indefinite due to roundoff error. True Marquardt does this automatically by biasing the step towards the steepest descent direction, and this has proved to be the most robust algorithm. To test line search algorithms two approaches were taken - GR-S and MQ-S. In both cases the algorithm occasionally failed to converge, the reason being that the descent direction produced was nearly orthogonal to the gradient vector and very small steps were taken until the algorithm ran into the preset iteration limit. Results obtained are summarized in Table 5.14, being averages over the same 30 starting points used with the SD methods. Values for the "labour" are again directly comparable with those of Table 5.9.

As with SD methods the AT line search is the best. However, with GN methods all line search algorithms are outperformed by true Marquardt, both in terms of reliability and efficiency. The TM algorithms are on a par with the best SD methods tested and so it would seem on the basis of this one example that the evaluation of second derivatives is not worthwhile.

	TM-U-G	TM-S-G	GR-S						MQ-S			
			FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	FAV-CF	FAV-QI	AT-QI	PQI
Number of starts converging	30	30	26	27	27	29	29	26	29	28	29	29
Average number of iterations	10.9	10.5	14.2	14.2	12.0	11.9	11.7	12.2	13.0	13.4	11.0	11.4
Average number of auxiliary function evaluations	15.6	14.8	42.1	29.6	33.0	48.4	58.2	81.6	29.0	22.5	25.9	44.6
Relative labour (CP time)	0.97	0.94	1.59	1.41	1.30	1.53	1.67	2.06	1.32	1.24	1.14	1.44

TABLE 5.14 Example A. Gauss-Newton Methods

5.8.4 Quasi-Newton and Conjugate Gradient Methods

The results of tests of QN and CG algorithms on example A are summarized in Tables 5.15 to 5.17. All performed realiably, converging to the optimal solution in all 30 cases. Reflection of poles and zeros in the unit circle, and changes to the pairing of real roots (section 4.9) were allowed at *every* iteration (i.e. $N_p = 1$) because tests with the DFP algorithm (with AT search) showed that values of N_p of 2, 3 and 6 were inferior to $N_p = 1$. Whenever such a change occurred, the process was re-started with steepest descent (for the QN methods, $\mathbf{T}$ was reset to $\mathbf{I}$). With regard to labour, the comparison of Tables 5.17 and 5.9, say, may be slightly unfair on the QN methods because they require only a matrix-vector multiplication to determine $\mathbf{p}$ whereas SD methods require solution of linear equations. Moreover, CG methods require only vector addition. Such overheads do not show in the tables which are for function evaluation only. However, it is felt that the main utility of (simple) example A is in predicting performance in more complicated cases, when function evaluation will certainly dominate the computer time.

The most obvious result is that even the best QN method (BFGS with AT-QI search) is considerably inferior to the true Marquardt GN algorithms and the best SD methods (1.58 as opposed to values of 1.00 and better). CG methods are much less efficient again, noneyielding a labour factor of less than 3.00.

The AT-QI line search is again the best for every QN and CG method tried. This is somewhat surprising, especially for the CG methods, because, theoretically, "exact" line searches are needed to obtain conjugacy. In theory the DFP algorithm also needs exact line searches although other workers (e.g.

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
DFP	32.8	27.6	23.6	18.1	17.9	16.9	16.8
BFGS	24.9	22.0	18.4	18.2	18.0	16.9	16.8
SR1	27.7	24.5	20.7	20.1	19.1	18.6	18.2
SR1-M	25.0	22.7	19.3	18.5	18.3	17.2	17.1
CG-A		54.8	41.4	31.0	30.0	29.0	28.6
CG-B		59.3	40.9	29.5	29.5	29.0	29.4

TABLE 5.15 Example A. Quasi-Newton and Conjugate Gradient Methods

Average number of iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD						
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
DFP	93.3	51.9	63.4	81.9	97.9	112.5	124.0
BFGS	99.3	49.5	56.1	80.3	97.3	110.3	121.0
SR1	118.9	61.9	67.6	96.3	108.0	125.8	135.2
SR1-M	91.3	49.2	57.9	85.1	100.8	115.0	125.9
CG-A		75.3	91.4	131.2	155.7	176.8	188.0
CG-B		81.5	90.0	125.0	154.6	177.8	192.9

TABLE 5.16 Example A. Quasi-Newton and Conjugate Gradient Methods

Average number of auxiliary function evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD						
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	LM (ε=0.1)	LM (ε=0.01)
DFP	2.71	1.88	1.89	1.95	2.20	2.38	2.55
BFGS	2.48	1.62	1.58	1.94	2.18	2.33	2.50
SR1	2.89	1.91	1.85	2.26	2.39	2.64	2.77
SR1-M	2.38	1.64	1.64	2.02	2.26	2.42	2.59
CG-A		3.30	3.02	3.21	3.55	3.82	3.98
CG-B		3.58	2.98	3.06	3.52	3.83	4.09

TABLE 5.17 Example A. Quasi-Newton and Conjugate-Gradient Methods

Relative labour spent in function evaluation

(best method = 1.00)

Sargent and Sebastian, 1972) have also found good performance with cruder search methods.

The best QN methods are BFGS and SR1-M, and there seems little to choose between the two. SR1-M is always better than SR1, indicating that the safeguards of Murtagh and Sargent are indeed worthwhile. The DFP algorithm achieves the same performance as BFGS and SR1-M when "exact" line searches are used (LM with $\varepsilon = 0.01$), but although all three get better as the line search accuracy is relaxed, DFP does not benefit as much as the other two.

The much improved performance of FAV-QI over FAV-CF is presumably due to the greater number of function evaluations needed to obtain an acceptable α in the early iterations, or following a matrix reset, when $\alpha = 1$ is a poor approximation (in this example, much too large). QI, which allows α to shrink by a factor of 0.2 (Section 5.4) will reach an appropriate small value long before CF (which uses 0.6).

There seems little to choose between the two CG update formulae A and B. All results quoted for CG methods use the "α pass" search technique which, for this example, proved more efficient than taking $\alpha = 1$. This was in spite of the need for substantially fewer iterations with $\alpha = 1$ in FAV and AT algorithms (42.6 vs 54.8 for FAV-QI and 29.5 vs 41.4 for AT-QI, both with CG-A). This represents possibly a serious deficiency of the α-pass algorithm used. Presumably α was required to grow outwards at a greater rate than that provided for by the adaptation factor of 1.6, resulting in many shorter-than-optimum steps. It would seem that a larger adaptation factor may be better, but this is likely to be problem-dependent. Since CG methods have proved to be the least efficient methods tested (with later, more complex

examples, extremely inefficient) this matter has not been explored further. As expected, the number of iterations required with the PQI search is virtually the same for "α-pass" and "α=1" variants, because the optimum steplength is located accurately with both. The α-pass method is considerably more efficient (3.21 vs 3.85 in labour) because of the better first trial.

5.8.5 Overall Conclusions - Example A

The best algorithm tested was MQ-U with AT-CI-C line search, that is, a second-derivative method employing curvature information to predict an initial value of α for the line search and to allow cubic interpolation rather than quadratic. It is probable that a slight improvement could be gained by using the curvature only for cubic interpolation, and taking an initial α of unity.

However, it seems scarcely worthwhile to evaluate second-derivatives at all, because the TM-GN algorithms (which do not require them) perform virtually as well. TM-GN algorithms are significantly better than GN with line search, both in reliability and efficiency.

If any line search algorithm is used, the AT search is recommended.

Quasi-Newton and particularly conjugate gradient methods are inferior in speed performance to the other classes, although they appear to be equally reliable.

The foregoing extensive analysis of this rather simple example was undertaken for a number of reasons. Firstly, many of the computer runs were performed during the development of the test program itself, when a realistic yet simple and well-

behaved test problem which did not require excessive computer time was needed. Secondly, the averaging of the figures over a fairly large number of starting points (30) creates some confidence that the small differences in performance observed are more than just coincidental. They should thus generalize (at least qualitatively) to more complex problems. Study of example A then provides a "feel" for the kinds of tests that could prove illuminating when applied to other problems. It must be emphasized, though, that some of the conclusions drawn so far are modified in the light of experience with later examples.

5.9 Example B - Log Magnitude Response

5.9.1 General

Example B presents a much more difficult problem than example A. The log magnitude response is specified at 30 frequencies and has the Chebyshev" form shown in Figure 5.10. The actual values used are in Table 5.18. Once again a least squares approximation is sought, with all weight factors set to unity. The approximating filter is a cascade of second-order sections, having three pole pairs and three zero pairs. All filter coefficients are variable, so that the dimensionality of the optimization problem (N) is 12.

Since only the (log) magnitude response is of interest, stability constraints are easily dealt with by the process of root reflection described in Section 4.9.1. This, together with the process of real root pairing (Section 4.9.2) eliminates many local minima of the objective function. However, during the course of testing, two distinct local minima were uncovered. The "good" solution has an objective function value of 7.7447×10^{-3} and coefficients as detailed in Table 5.19. The "bad"

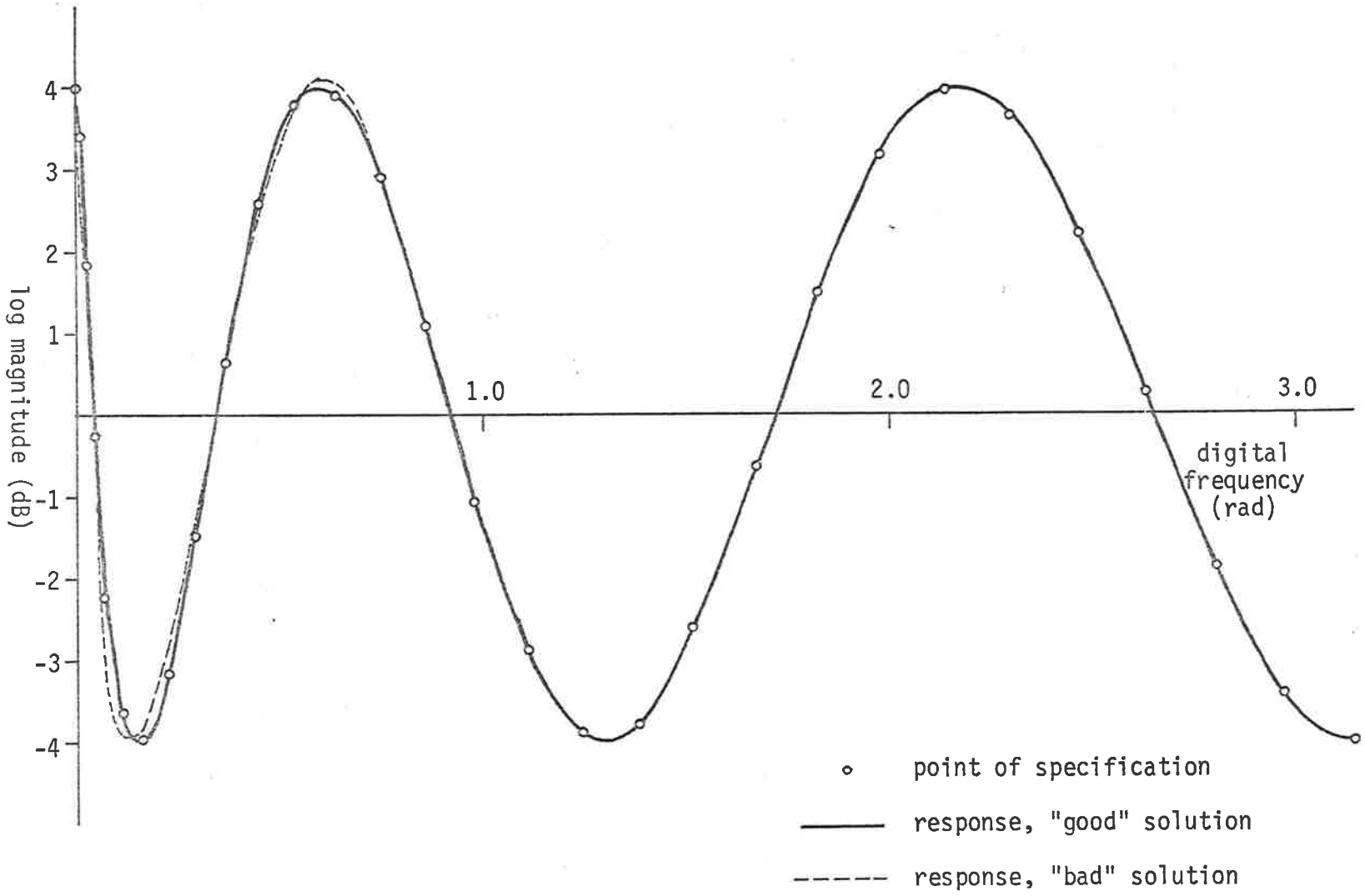


FIGURE 5.10 Example B. Specified and Achieved Log Magnitude Response

Digital frequency (radians)	log magnitude (dB)
0.000200000000	4.000000000000
.004607416271	3.427428704702
.018416150737	1.873633762909
.041385700002	-.216555634156
.073448690420	-2.244748261244
.114511075715	-3.630301678554
.164452412837	-3.976551828658
.223126215238	-3.184372263085
.290360382548	-1.480552621820
.365957705369	.647127985661
.449696443728	2.589545138655
.541330977478	3.790612684514
.640592526742	3.906482223000
.747189940294	2.903981968246
.860810549553	1.070113354952
.981121085697	-1.070113353222
1.107768657193	-2.903981967011
1.240381784894	-3.906482222614
1.378571491646	-3.790612685087
1.521932443230	-2.589545140024
1.670044137274	-.647127987433
1.822472136661	1.480552620151
1.978769343805	3.184372261998
2.138477312067	3.976551828463
2.301127590461	3.630301679308
2.466243097695	2.244748262731
2.633339521536	.216555635950
2.801926739385	-1.873633761321
2.971510255884	-3.427428703776
3.141000000000	-4.000000000000

TABLE 5.18 Target Response for Example B

Pair number	Num. or Den.	Coefficients	Comments
1	N	a = -0.401600 b = 0.469985	Complex conjugate zero pair r = 0.685555, θ = ±1.273536
2	N	a = -1.735203 b = 0.761976	Complex conjugate zero pair r = 0.872912, θ = ±0.110365
3	N	a = -0.350289 b = -0.611791	Real zeros z_1 = -0.626396, z_2 = 0.976685
4	D	c = -1.281080 d = 0.583310	Complex conjugate pole pair r = 0.763747, θ = ±0.575940
5	D	c = 0.717921 d = 0.416657	Complex conjugate pole pair r = 0.645490, θ = ±2.160489
6	D	c = -1.914682 d = 0.915758	Real poles p_1 = 0.930068, p_2 = 0.984614

TABLE 5.19 Example B - "Good" solution

Pair number	Num. or Den.	Coefficients	Comments
1	N	a = -0.413849 b = 0.455574	Complex conjugate zero pair r = 0.674962, θ = ±1.259207
2	N	a = -1.727317 b = 0.746987	Complex conjugate zero pair r = 0.864284, θ = ±0.038050
3	N	a = 1.617394 b = 0.617587	Real zeros z_1 = -0.617899, z_2 = -0.999495
4	D	c = -1.283316 d = 0.599666	Complex conjugate pole pair r = 0.774381, θ = ±0.594180
5	D	c = 0.736115 d = 0.429021	Complex conjugate pole pair r = 0.654997, θ = ±2.167504
6	D	c = 0.031565 d = -0.967452	Real poles p_1 = 0.967935, p_2 = -0.999500

TABLE 5.20 Example B - "Bad" solution

solution, having a function value of 5.4184×10^{-2}, is detailed in Table 5.20. The obvious deficiency is that one real pole almost exactly cancels a real zero, and so two degrees of freedom are effectively wasted. Figure 5.10 shows the inferior spectral fit resulting.

It is somewhat surprising (and fortunate) that there seem to be only two local minima in this complicated twelve-dimensional problem. It is still possible to generate starting points at random (in the manner of Section 5.2.2) with a good chance of convergence to the optimal solution.

The tests performed were not as comprehensive as those for example A, owing mainly to the much greater computer time required. For each algorithm tested, only ten (random) starting points were used, and the statistical reliability of the results is further reduced because averaging was done only over those runs which converged to the "good" solution. However, some interesting points emerge.

5.9.2 Second-Derivative Methods

The results obtained for the SD methods are summarized in Tables 5.21 to 5.25. Blank positions indicate that the test was not run. The figures for "total labour" (CP time) are again relative to the MQ-U algorithm with AT-QI search (this is not the best method for example B, but it was adopted as the datum for uniformity with other examples).

The interpretation of the results for this example is greatly complicated by the fact that convergence was not obtained from all starting points. For example, FAV-CF seems vastly superior to FAV-QI for the GM method (56 pre-Newton iterations versus 78)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD								
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	4	6	6			5	4	6	5
GR-U			7				5	8	8
MQ-S	7	10	8	8	9	9	10	9	9
MQ-U	10	10	10	9	9	9	10	9	9
GM	4	3	8	5		7	8	6	5
FF	7	7	9	9		6	6	10	10

TM-U-N: 10

TM-S-N: 9

TABLE 5.21 Example B. Second Derivative Methods

Number of runs converging to "good" solution (out of 10)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD								
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	64	73	60			102	66	62	62
GR-U			58				74	63	60
MQ-S	70	83	66	63	65	74	72	66	68
MQ-U	72	81	69	61	62	69	72	66	65
GM	67	92	77	74		82	94	93	65
FF	73	81	81	75		104	95	77	81

TM-U-N: 87

TM-S-N: 92

TABLE 5.22 Example B. Second-derivative methods

Average number of iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD								
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	52	56	42			89	51	43	43
GR-U			40				56	46	41
MQ-S	56	67	47	44	44	61	57	46	48
MQ-U	58	64	51	43	42	57	59	51	45
GM	56	78	59	56		71	76	77	46
FF	59	66	62	55		93	79	63	62

TM-U-N: 75

TM-S-N: 76

TABLE 5.23 Example B. Second-derivative methods

Average number of "Pre-Newton" iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD								
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	153	128	158			274	126	176	282
GR-U			148				137	175	278
MQ-S	138	120	156	293	343	148	105	158	291
MQ-U	137	121	165	278	326	129	110	163	276
GM	362	257	275	410		268	198	281	314
FF	304	166	236	378		274	163	205	353

TM-U-N: 129

TM-S-N: 134

TABLE 5.24 Example B. Second-derivative methods

Average Number of Auxiliary Function Evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C	PQI-C
GR-S	0.93	1.00	0.89			1.52	0.91	0.94	1.08
GR-U			0.85				1.02	0.95	1.05
MQ-S	0.97	1.10	0.95	1.10	1.19	1.03	0.95	0.96	1.16
MQ-U	1.00	1.08	1.00	1.06	1.14	0.95	0.96	0.96	1.10
GM	1.24	1.38	1.24	1.38		1.28	1.33	1.43	1.15
FF	1.23	1.14	1.23	1.35		1.54	1.29	1.14	1.39

TM-U-N: 1.16

TM-S-N: 1.22

TABLE 5.25 Example B. Second-derivative methods

Relative Labour (CP time) Spent in Function Evaluation

but the difference is due mainly to the fact that only starts 3, 4, 5 and 10 converged for FAV-CF whereas only starts 2, 6 and 9 did so for FAV-QI. With the GR-S method, starts 1, 2, 3 and 7 converged for both FAV-CF-C and FAC-QI-C (in similar numbers of iterations) but FAV-CF-C looks much worse in the tables merely because a further start also converged, taking 226 iterations to do so. In all cases, those runs which missed the "good" solution converged to the "bad" solution, and so the algorithms themselves can still be regarded as reliable.

With these provisos regarding interpretation, the following conclusions are drawn:

(a) The AT-QI line search is still marginally the best of those not using curvature. However, FAV-CF is virtually as good. The added complexity of PQI and LM is not justified. The performance of FAV-QI seems poor relative to that of FAV-CF, but this is probably not statistically significant. (Considering the MQ-U algorithm (for which all starts converged), FAV-QI requires 121/81 = 1.49 auxiliary function evaluations *per iteration* whereas FAV-CF needs 137/72 = 1.90, so that the incorporation of slope information *is* still useful in predicting the minimum in the line searches. The poorer performance of FAV-QI is then entirely due to the greater number of iterations.) The introduction of curvature information does not markedly affect the figures. The differences in performance figures for all line searches are small anyway, so that again AT-QI can be recommended (or FAV-CF, because of even greater simplicity).

(b) Regarding methods for determining the search direction, the two Greenstadt methods seem slightly better than Marquardt,

with GM and FF significantly worse again. Marquardt would still be recommended, however, because of Greenstadt's requirement for eigensystem analysis, which becomes increasingly time consuming as dimensionality increases.

For example, the total CP times (for 10 starting points) using the AT-QI search were 108s for MQ-S and 173s for GR-S. Admittedly, the row cyclic Jacobi method used for eigensystem analysis is much inferior to others such as Householder's method (Wilkinson, 1960). Nevertheless, the Marquardt method would still have the edge in efficiency, as well as being much simpler to implement. There are no significant differences in performance between scaled and unscaled algorithms.

(c) The true Marquardt methods are again simple and reliable, but significantly inferior in performance to the best line search algorithms.

5.9.3 Gauss-Newton Methods

Results for the GN methods are summarized in Table 5.26. The true Marquardt GN methods are comparable in efficiency with the best SD line search algorithms whereas the line search GN methods generally are not. This finding is in accordance with that of example A. In fact, in the present example, TM-U-G seems significantly better than any SD algorithm and it would be tempting to recommend it for general use. However, the study has revealed one further very important feature. This is discussed below.

In the running of GN tests, limits were placed on both the number of iterations (250) and auxiliary evaluations (1200).

	METHOD				
	AT-QI MQ-S	AT-QI GR-S	AT-QI MQ-U	TM-S-G	TM-U-G
Number of Starts Converging*	5/9	5/9	7/10	6/10	6/10
Average number of Iterations	106	94	69	75	66
Average Number of Auxiliary Function Evaluations	590	527	325	170	151
Relative Labour (CP time)	1.77	1.57	1.07	0.92	0.81

* to the "good" solution, $F = 7.7447 \times 10^{-3}$

TABLE 5.26 Example B. Gauss-Newton Methods

Of those runs which did not converge, the true Marquardt runs always ran into the iteration limit whereas the line search tests ran into the evaluation limit. One reason for the latter failure is that found in conjunction with example A, viz., a search vector is calculated which is almost orthogonal to the gradient, resulting in very small steps. This problem surfaces regardless of whether the search is within the "capture area" of the "good solution" or the "bad solution". However, there is another fundamental problem associated with the convergence of any GN method to the "bad" solution; that problem elucidated by McKeown (1975) and discussed in Section 3.4.12 of this thesis. In the context of the present example, it seems that the matrix $\mathbf{R}$ ($= 2\ \mathbf{J}^T\mathbf{J}$) used in GN methods is *at the bad solution* an insufficiently good approximation to the Hessian $\mathbf{H}$ to allow Newton-like convergence. Under such conditions even true Marquardt GN methods can perform very poorly. In the test of TM-S-G, one of the failing runs had approached the "bad" solution very closely after 250 iterations and three still exhibited vastly greater function values. With TM-U-G the numbers were respectively two and two.

This behaviour contrasts with the good convergence of the GN methods to the "good" solution (where the residuals are much smaller) and of the true Newton (SD) methods to both solutions (for SD methods the sizes of residuals do not matter). The "McKeown parameter" τ/λ (Section 3.4.12) was calculated as 9.2×10^3 at the "good" solution and 7.9×10^9 at the "bad" solution, and convergence problems could certainly be expected in the latter case.

Thus it seems that unless there is good reason to believe

that the residuals *will* be small (that is, that the optimal filter will achieve the desired response to a high degree of accuracy), it is most risky to rely on a GN method. A second-derivative method would be much more reliable for general purpose use.

5.9.4 Quasi-Newton and Conjugate Gradient Methods

Several tests of quasi-Newton methods were run on example B. When root reflection and pairing changes were allowed at every iteration (i.e. $N_p = 1$), convergence was never obtained, the frequent resets not allowing any build-up of Hessian information in the iteration matrix. In fact, real root pairing was changed at almost every iteration, so that the convergence would be only steepest-descent like.

Setting $N_p = 20$ produced the results summarized in Table 5.27. The algorithms for the most part converge reliably, although all except SR1-M with FAV-QI and DFP failed with one particular starting point. This was merely because "convergence" occurred to a "solution" with badly-paired real roots after only 14 iterations; that is, before pairing had been checked. The very simple remedy would be to check the pairing at any "solution" and continue if necessary. In the three cases where DFP failed to converge, the test was stopped at 600 iterations. The function value was decreasing steadily but very slowly.

BFGS shows a slight superiority in performance over SR1-M, with DFP being easily the worst. Since BFGS comes well within a factor of 2 of the efficiency of SD methods, it could be recommended as a worthwhile alternative if time is not critical and the calculation of second derivatives is considered too laborious to program. In particular, it would be preferable to TM-U-G

	METHOD				
	FAV-QI BFGS	AT-QI BFGS	FAV-QI SR1-M	AT-QI SR1-M	AT-QI DFP
Number of Starts Converging*	9	8	10	9	7
Average Number of Iterations	234	201	265	232	444
Average Number of Auxiliary Function Evaluations	421	540	467	605	1077
Relative Labour (CP time)	1.55	1.57	1.74	1.78	3.31

* to "good" solution, out of 10.

TABLE 5.27 Example B. Quasi-Newton Methods

Pairing and Reflection Allowed Every 20 Iterations

because of the convergence problems of the latter.

BFGS with AT-QI was also tested with two other variants. Firstly, N_p was reduced to 10 (from 20) with no degradation of performance. Secondly, with N_p = 10, the existing iteration matrix was *retained* even if reflection or a pairing change had occurred, (but was reset to **I** whenever the process failed to produce a descent direction). In this case, the labour figure increased from 1.57 to 1.58, confirming the intuitive prediction that the information in the matrix would be useless after such a parameter "shuffle", and the matrix accordingly should be re-initialized.

Tests of CG algorithms were not encouraging. The CG-A method with the AT-QI "α-pass" line search (and resets to steepest descent after every 12 iterations) was run to 600 iterations from each of nine starting points. In no case was the solution even remotely approached, the objective function value always remaining above 10^{-1}. Since it was suspected (from example A) that the "α-pass" technique could lead to under-estimation of the steplength with AT-QI, with an excessive number of iterations being required, five runs using AT-QI and four using PQI were performed using the "α = 1" trial. All ran to the limit of 600 iterations without the convergence criterion being met, although in four cases (AT-QI) and three cases (PQI) a "reasonable" solution having a smaller F value than the so-called "bad" minimum had been obtained. However, the number of function evaluations required was of the order of 4000 for AT and 5000 for PQI, which is grossly excessive when compared with other methods.

5.10 Example C - An All-Pass Group Delay Equalizer

5.10.1 General

This example concerns the design of an all-pass equalizing filter to be cascaded with an elliptic low-pass filter of order 4. The passband is $\theta = 0$ to 0.5π and the stopband is $\theta = 0.6\pi$ to π. If the low-pass is realized as a cascade of two second-order recursive sections (as in equation 4.5 and figure 4.3) its coefficients are as in Table 5.28 and its magnitude and group delay responses are shown in Figure 5.11 and 5.12 respectively. The group delay distortion (deviation from flatness) within the passband is about 8T seconds. This example is due to Deczky (1973).

The equalizer to be designed consists of three cascaded all-pass sections. A least-squares objective function is used, together with twenty-one nonuniformly distributed frequency sample points, generated in accordance with the formula

$$\theta_i = \frac{\pi}{2} \sin\left(\frac{\pi i}{40}\right) \qquad i = 0, 1, \ldots 20 \tag{5.24}$$

At each of these points the "desired" group delay, is specified as zero, although any constant value would have served just as well because the problem is an example of "formulation B3" (Section 4.4.2) in which the *shape* of the response curve is optimized rather than the actual level. The group delay resulting after equalization is also shown in Figure 5.12. The reasons for the use of a non-uniform frequency set are:

(a) that the number of points per cycle of the error function is approximately constant and

(b) that there are proportionately more points in the frequency

Section 1

$a_1 = 1.62178$

$b_1 = 1.0$

$c_1 = -0.403133$

$d_1 = 0.233280$

Section 2

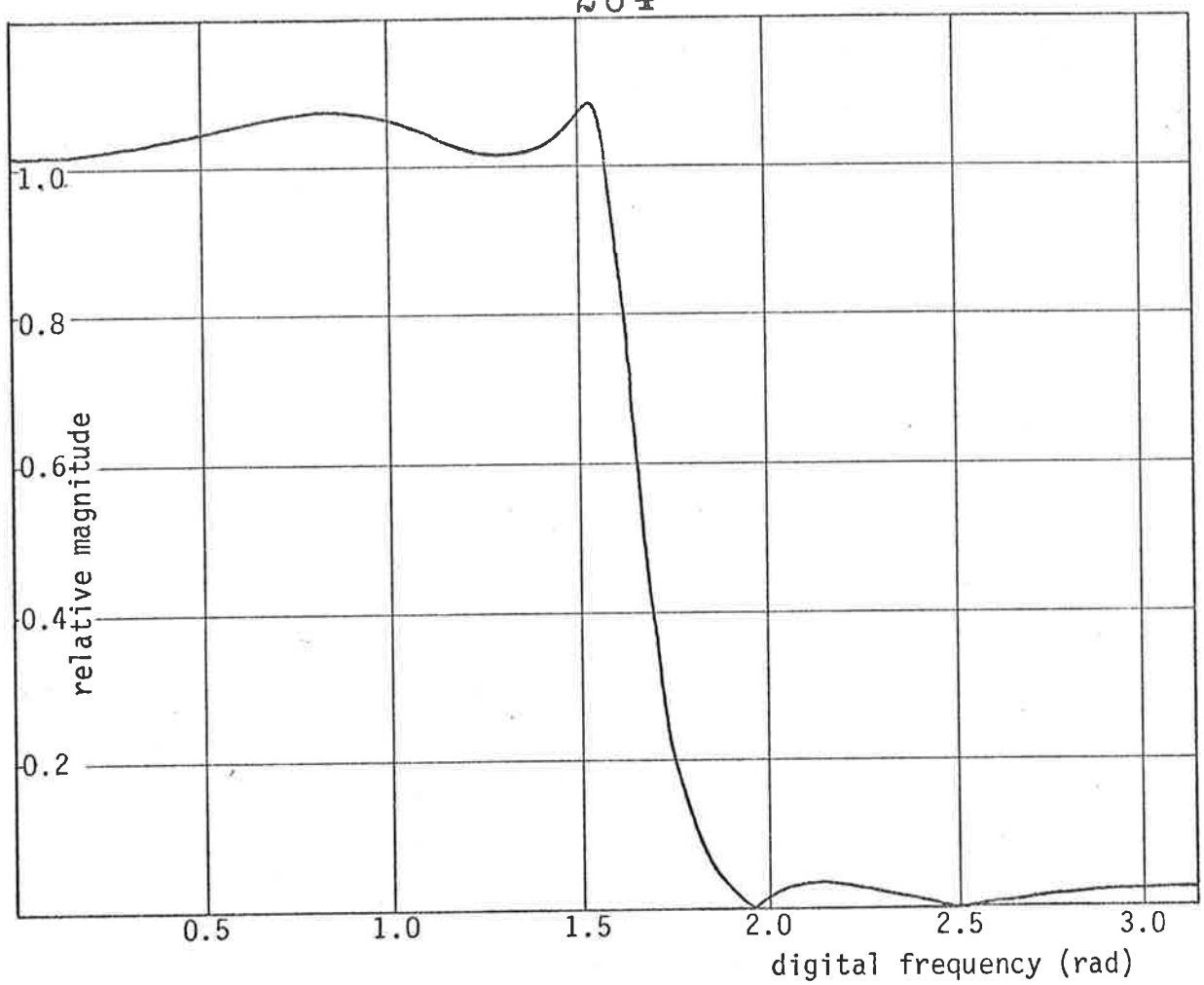
$a_2 = 0.718956$

$b_2 = 1.0$

$c_2 = 0.051401$

$d_2 = 0.797295$

TABLE 5.28 Coefficients for Elliptic Low-pass Filter of Example C



$$H(z) = \prod_{k=1}^{2} \frac{1 + a_k z^{-1} + z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}}$$

$a_1 = 1.62178$ $\quad c_1 = -0.403133$ $\quad d_1 = 0.233280$

$a_2 = 0.718956$ $\quad c_2 = 0.051401$ $\quad d_2 = 0.797295$

FIGURE 5.11 Example C. Magnitude Response of 4th-order Elliptic Lowpass Filter

(Parameters from Deczky (1973) p99)

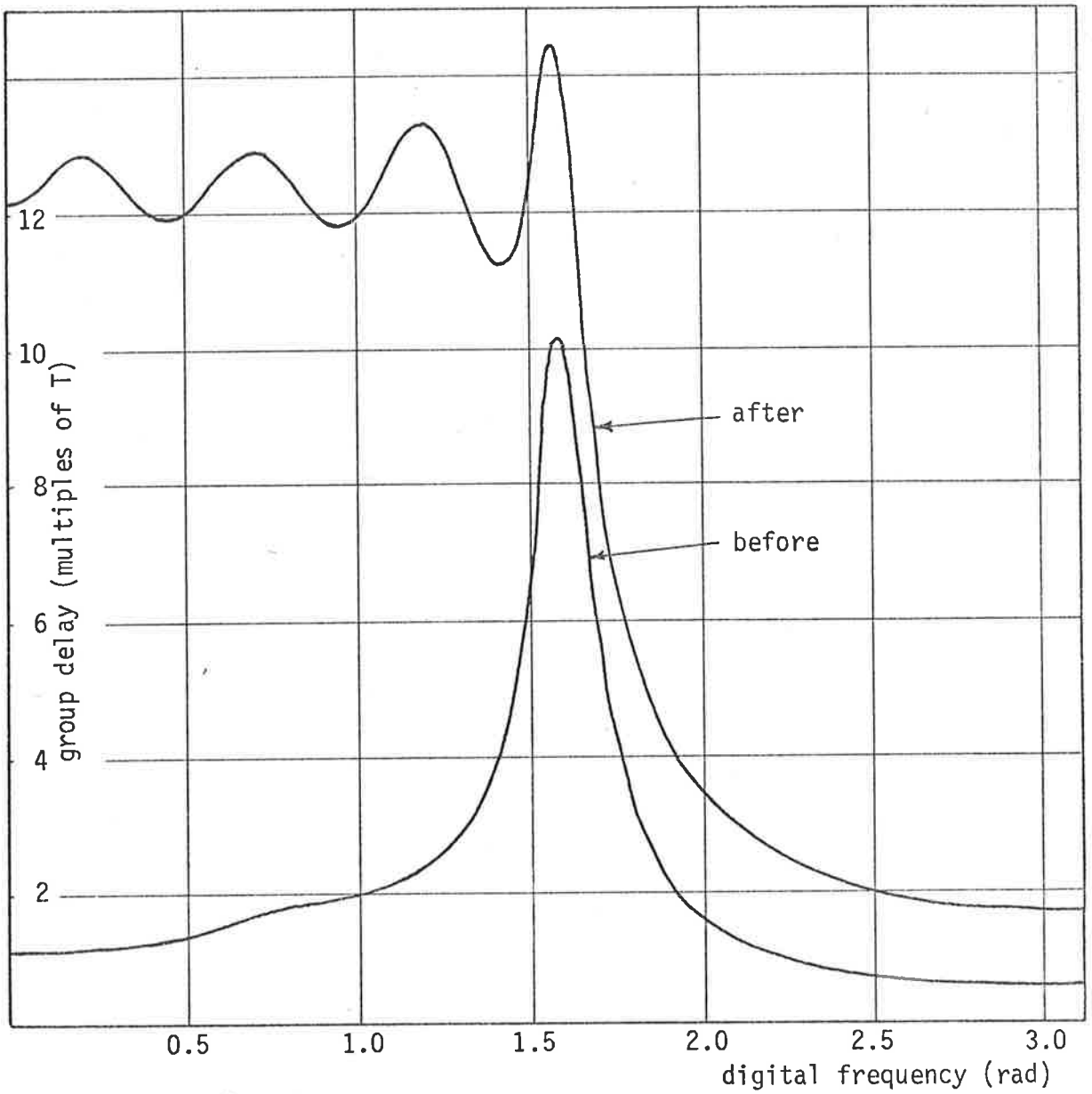


FIGURE 5.12 Example C. Group Delay Response of Elliptic Lowpass Filter

Before and After Equalization by a 3-section Allpass Filter
Fitted to a 21-point Non-uniform Discrete Frequency Set

Section 3

$a_3 = -0.869690$

$b_3 = 1.679755$

$c_3 = -0.517748$

$d_3 = 0.595325$

Section 4

$a_4 = -2.655233$

$b_4 = 1.862732$

$c_4 = -1.425451$

$d_4 = 0.536846$

Section 5

$a_5 = -2.018043$

$b_5 = 1.821610$

$c_5 = -1.107835$

$d_5 = 0.548965$

TABLE 5.29 Coefficients of Group Delay Equalizer to be Cascaded with Low-pass Filter of Table 5.28

region where the delay distortion tends to be large, i.e. near the passband edge.

Deczky (1973) recommends at least 10 points per error cycle (i.e. 35 points in this example) but it was found experimentally that 21- and 40- point sets produced almost indistinguishable results. Deczky has designed one-and four- section equalizers using a least pth objective function with p = 5, and a DFP optimization algorithm. In the present work, however, only a least squares solution was sought because of the ready applicability of SD and GN methods in this case. A second-order all-pass section has two independent designable parameters, so that the dimensionality of this problem is 6. To simplify stability checking, the denominator coefficients c_k and d_k (for the kth section) are taken as the independent variables, the numerator coefficients being obtained through the relations:

$$a_k = c_k/d_k \tag{5.25}$$

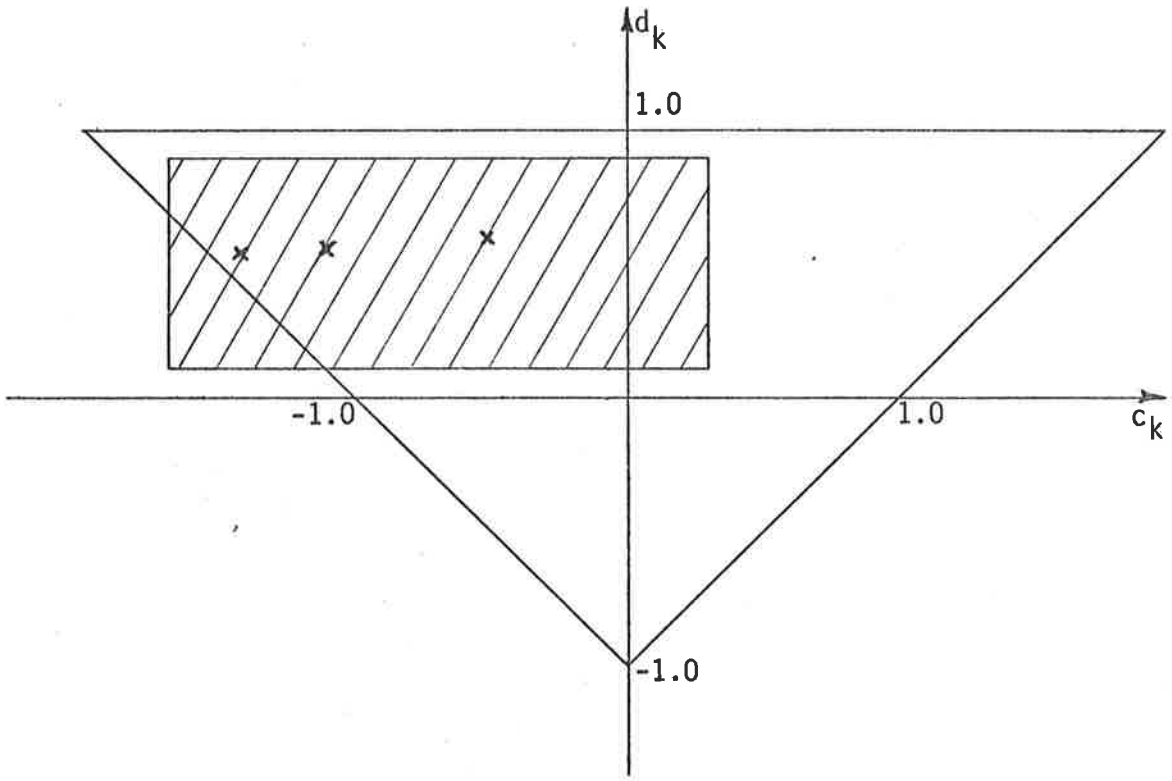
$$b_k = 1/d_k \, . \tag{5.26}$$

The optimal coefficients of the equalizer are given in Table 5.29. The minimum value of the objective function is 12.81817. Three other local minima, having function values between 30 and 40, were obtained very occasionally. Convergence of the optimization process was deemed to have occurred when all gradient components and all components of the $\mathbf{p}$ vector were absolutely less than 0.001. Reflection of poles and zeros in the unit circle is *not* a valid procedure as it was for the examples involving log magnitude response, and so stability constraints must be specifically considered during the course of

each iteration. When the random starting points were generated in accordance with Section 5.2.2, many runs had to be abandoned because a constraint was encountered, as no provision had been made to handle such a situation. This is not surprising, but nor is it a serious objection to the use of such methods for the design of equalizers because convergence *was* obtained in quite a large proportion of cases. What it does mean is that for a meaningful comparison of the convergence properties of different algorithms, a more localized set of starting points is required. Accordingly, the randomization method of Section 5.2.3 was used, with all c coefficients having a "centre value" of -0.7 and a "maximum deviation" of 1.0, and d coefficients using 0.5 (centre) and 0.4 (deviation). Figure 5.13 shows the allowed starting region as well as the locations of the optimal coefficients. Forty such starting points were used in the tests. Of these, eight violated a stability constraint and so were abandoned immediately. Of the remaining thirty-two, an average of twenty led to convergence to the optimal solution. Although with every method tested there were some starting points which did not lead to convergence, the averaging of results over those starts which *did* converge is perhaps more meaningful with this example than with example B. This is because some *particular* starting points (about 15) led to convergence in almost all test runs, whereas others (about 8) did so infrequently or never.

5.10.2 Second-Derivative Methods

The results obtained for SD methods are summarized in Tables 5.30 to 5.34. As with previous examples, MQ-U with AT-QI search is given the base "labour" value, 1.00.



Key:

Stable Region (triangle) } each pole pair, k=1,2,3.
Starting Region (shaded) }

Optimal Pole Locations (crosses)

FIGURE 5.13 Example C. Coefficients Determining Pole Positions, All-pass Group Delay Equalizer

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	FAV-CI-C	AT-CI-C
GR-S	22	21	20	20	21	21
GR-U	22	18	17	18		
MQ-S	25	24	25	22	23	22
MQ-U	20	20	18	18		
GM	21	22	22	21	19	21
FF	18	19	21	16	21	21

TM-U-N: 19

TM-S-N: 25

TABLE 5.30 Example C. Second-derivative methods

Number of starts converging to the optimal solution (out of 32)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD					
	FAV-CF	FAV-QI	AT-QI	PQI	FAV-CI-C	AT-CI-C
GR-S	11.2	11.9	10.5	9.7	10.8	10.1
GR-U	12.3	12.4	10.2	9.5		
MQ-S	14.8	15.9	13.2	10.9	13.9	12.2
MQ-U	14.1	14.7	12.3	11.2		
GM	11.8	12.5	11.4	10.5	11.4	10.7
FF	12.4	13.5	12.0	10.2	12.7	12.0

TM-U-N: 15.7

TM-S-N: 17.9

TABLE 5.31 Example C. Second-derivative methods

Average number of iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	FAV-CI-C	AT-CI-C
GR-S	4.5	5.2	4.6	4.2	4.5	4.7
GR-U	6.0	6.1	4.6	4.4		
MQ-S	8.6	9.0	7.2	5.0	7.3	6.4
MQ-U	8.2	8.5	6.3	6.1		
GM	4.7	5.5	5.1	4.6	4.6	4.5
FF	5.1	6.7	5.2	4.4	5.4	6.0

TM-U-N: 9.2

TM-S-N: 11.2

TABLE 5.32 Example C. Second-derivative methods

Average Number of Pre-Newton Iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	FAV-CI-C	AT-CI-C
GR-S	18.4	16.2	21.0	38.5	15.1	21.8
GR-U	19.1	16.7	19.2	37.8		
MQ-S	20.0	19.7	25.1	49.2	17.1	24.3
MQ-U	19.0	17.9	23.3	49.1		
GM	26.0	20.1	25.7	45.4	16.5	22.4
FF	23.5	20.6	26.1	40.1	17.0	25.3

TM-U-N: 25.2

TM-S-N: 22.4

TABLE 5.33 Example C. Second-derivative methods

Average Number of Auxiliary Function Evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	FAV-CI-C	AT-CI-C
GR-S	0.88	0.91	0.86	0.97	0.82	0.84
GR-U	0.96	0.94	0.83	0.95		
MQ-S	1.12	1.19	1.08	1.15	1.04	1.00
MQ-U	1.07	1.09	1.00	1.17		
GM	0.99	0.98	0.96	1.09	0.88	0.89
FF	1.01	1.05	1.01	1.02	0.96	1.00

TM-U-N: 1.23

TM-S-N: 1.34

TABLE 5.34 Example C. Second-derivative methods

Relative Labour (CP time) Spent in Function Evaluation

In the matter of comparisons between line search methods, Tables 5.31 and 5.32 indicate that the number of iterations decreases as the accuracy of the line search is improved. However, as expected, the number of auxiliary evaluations increases (Table 5.33). From Table 5.34, the AT-QI search again represents the optimum from the point of view of total labour. The use of curvature information appears to be marginally beneficial, saving about 10% of the labour in the FAV algorithms, but somewhat less in AT algorithms.

The order of preference for search direction algorithms would appear to place the two GR methods first, followed by GM, then MQ-U and FF together, with MQ-S last. However, examination of Table 5.30 shows that MQ-S converges in significantly more cases than the others. Thus it could be that the averaging makes it appear worse simply because it converges from several "difficult" starting points for which the others fail. To test this hypothesis the averages were re-computed (for the AT-QI runs) over those 15 starting points which produced convergence with *every* method. Results are presented in Table 5.35, from which it is seen that MQ-S is actually performing as well as MQ-U and FF. However, a slight advantage remains with GM, and GR-U and GR-S are still the best. In this example GM and FF are shown in a better light than in previous examples. Further discussion is deferred to Section 5.14.

As with examples A and B, the true Marquardt SD methods are considerably less efficient than the best line search algorithms.

5.10.3 Gauss-Newton Methods

Tests were made of both true Marquardt and line search GN methods. Results are presented in Table 5.36. The findings are at variance with those for earlier examples in that the line

Method	Average No. of Iterations	Average No. of Auxiliary Evaluations	Relative Labour
GR-S	9.7	18.9	0.83
GR-U	9.8	18.3	0.83
MQ-S	11.3	20.7	0.96
MQ-U	11.7	22.1	1.00
GM	10.1	21.7	0.89
FF	11.1	23.6	0.98

TABLE 5.35 Example C

Performance of various SD methods with AT-QI line search

Averages over 15 identical starting points

	METHOD			
	TM-U-G	TM-S-G	MQ-U FAV-QI	MQ-U AT-QI
Number of Starts Converging	13	16	19	20
Average Number of Iterations	13.9	15.9	15.1	12.7
Average Number of Auxiliary Function Evaluations	25.8	24.9	20.8	24.1
Relative Labour (cf. Table 5.34)	0.85	0.93	0.86	0.79

TABLE 5.36 Example C

Gauss-Newton Methods

search methods seem to be at least as good as the TM methods. All cases of non-convergence were due to a constraints being encountered, rather than to the near-orthogonality of gradient and search direction or to the excessive size of the residuals.

In terms of total labour all methods tested rated somewhat better than the SD methods.

5.10.4 Quasi-Newton and Conjugate-Gradient Methods

The results obtained for QN and CG methods are summarized in Tables 5.37 to 5.40. The CG methods employed the "$\alpha = 1$" search variants, which for this problem were more efficient than "α-pass". This was because, for AT-QI search, the number of iterations was much smaller (77.9 vs 200.3 with CG-A) although marginally more auxiliary evaluations were required (548 vs 426). This confirms the finding with example A that underestimation of steplength can severely cripple the "α-pass" strategy with AT searches, although "α-pass" would presumably be advantageous almost always with a PQI search. In any case, the results indicate that even very careful tuning would be unlikely to make a CG method competitive with the other classes of algorithm.

Turning attention now to the QN methods, the BFGS and SR1-M techniques are again significantly better than DFP. The FAV-QI line search seems to be virtually as good as AT-QI, with PQI again not being worthwhile. (However, the AT search at least is needed to get the best performance from DFP). FAV-CF is inferior to FAV-QI although the number of iterations is smaller. This, presumably, is due to the greater number of evaluations needed to obtain an acceptable value of α following a matrix reset. The same effect was seen with example A.

SEARCH DIRECTION METHOD	LINE SEARCH METHOD ($N_p = 1$)				
	FAV-CF	FAV-QI	AT-QI	PQI	AT-QI $N_p=6$
DFP	7	15	18	15	
BFGS	7	17	18	15	18
SR1-M	7	16	17	16	
CG-A		15	17	17	17
CG-B		16	17	17	17

TABLE 5.37 Example C. Quasi-Newton and Conjugate Gradient Methods

Number of starts converging to the optimal solution (out of 32)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD (N_p=1) FAV-CF	FAV-QI	AT-QI	PQI	AT-QI N_p=6
DFP	54.9	67.3	28.8	22.0	
BFGS	26.1	26.5	24.6	21.9	24.9
SR1-M	26.3	27.2	22.7	23.1	
CG-A		121.7	77.9	77.5	78.5
CG-B		107.2	65.6	67.5	65.9

TABLE 5.38 Example C. Quasi-Newton and Conjugate Gradient Methods

Average Number of Iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD (N_p=1) FAV-CF	FAV-QI	AT-QI	PQI	AT-QI N_p=6
DFP	157.6	98.1	89.3	132.9	
BFGS	115.7	62.1	81.4	115.7	82.7
SR1-M	142.9	72.1	79.4	131.8	
CG-A		847.7	548.2	664.9	551.3
CG-B		718.8	452.8	573.7	454.4

TABLE 5.39 Example C. Quasi-Newton and Conjugate Gradient Methods

Average Number of Auxiliary Function Evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD $N_p=1$ FAV-CF	FAV-QI	AT-QI	PQI	AT-QI $N_p=6$
DFP	3.36	3.22	1.82	2.00	
BFGS	1.98	1.49	1.60	1.84	1.63
SR1-M	2.25	1.60	1.52	2.03	
CG-A		12.14	7.82	8.92	7.88
CG-B		10.43	6.51	7.71	6.53

TABLE 5.40 Example C. Quasi-Newton and Conjugate Gradient Methods

Relative Labour (CP time) in Function Evaluation

Restricting root pairing changes to every sixth iteration made virtually no difference.

The best QN methods require about twice the function evaluation time of the best SD and GN methods.

5.11 Example D - Simultaneous Optimization of Magnitude and Group Delay

5.11.1 General

The low-pass filter and group delay equalizer whose design was considered in Section 5.10 has an overall complexity of five biquadratic sections, or equivalently ten poles and ten zeros, involving twenty coefficients. The technique of using only two sections to shape the magnitude response and appending an all-pass group delay equalizer is convenient, but by no means represents the best that can be done with a filter of such complexity. This section deals with the optimization of an objective function which includes contributions from both magnitude and group delay responses, allowing all twenty coefficients to vary independently. In a problem of such complexity a large number of local minima could be expected. No attempt has been made to find the globally optimal solution, but only to seek a local improvement, taking the all-pass solution as a starting point.

The art in the formulation of such problems is in selecting relative weights such that errors in the group delay response are traded off realistically against magnitude response errors. Since in the present case the chief concern is with the performance of optimization algorithms, no attempt was made to

automate the choice of weights. Rather, they were selected on a trial-and-error basis.

With example C, the original filter considered was a two-section low-pass filter designed by the transformation of a "classical" s-plane design. In order to allow the use of optimization techniques throughout the present study, a similar filter was designed by reformulating the specification as a least squares problem. In anticipation of improved performance in the final design, the opportunity was taken at the outset to reduce the width of the transition band slightly, so that the passband is now defined by $\theta = 0$ to 1.57 and the stopband by $\theta = 1.86$ to π. A two-section low-pass filter (with zeros constrained to lie on the unit circle) was designed by least squares magnitude optimization (with unit weights) over the following set if 42 non-uniformly distributed frequencies:

(a) Passband (specified magnitude = 1.0)

$$\theta_i = 1.57 \sin \frac{\pi i}{40} \qquad i = 0, 1, \ldots 20 \tag{5.27}$$

(b) Stopband (specified magnitude = 0.0)

$$\theta_{21+i} = \pi - (\pi - 1.86) \sin \frac{\pi i}{40} \qquad i = 0, 1, \ldots 20. \tag{5.28}$$

The reason for the use of a non-uniform frequency set is the same as with example C. The filter coefficients found are detailed in column A of Table 5.41 (Sections 1 and 2). The magnitude response is shown in Figure 5.14. This is closely comparable with the response of the elliptic filter of Figure 5.11, and shows that with a suitable choice of frequency set a

Section Number	Coefficient	(A) Two-section lowpass filter with three-section group delay equalizer	(B) Optimized Filter
1	a	1.662506	1.498160
	b	1.0	0.908732
	c	-0.3428260	-0.728966
	d	0.2087280	0.560747
2	a	0.6742920	0.618187
	b	1.0	0.971763
	c	0.06114800	0.204273
	d	0.7942140	0.857456
3	a	-2.019779	-2.206958
	b	1.839906	2.249604
	c	-1.097762	-1.147933
	d	0.5435060	0.514874
4	a	-2.670584	-2.986846
	b	1.886287	2.368466
	c	-1.415789	-1.372330
	d	0.5301420	0.492890
5	a	-0.8661150	-0.889819
	b	1.695815	2.022727
	c	-0.5107370	-0.202822
	d	0.5896870	0.645425
		F = 19.53	F = 1.285544

TABLE 5.41 Example D. Coefficients of Original and Optimized Filters

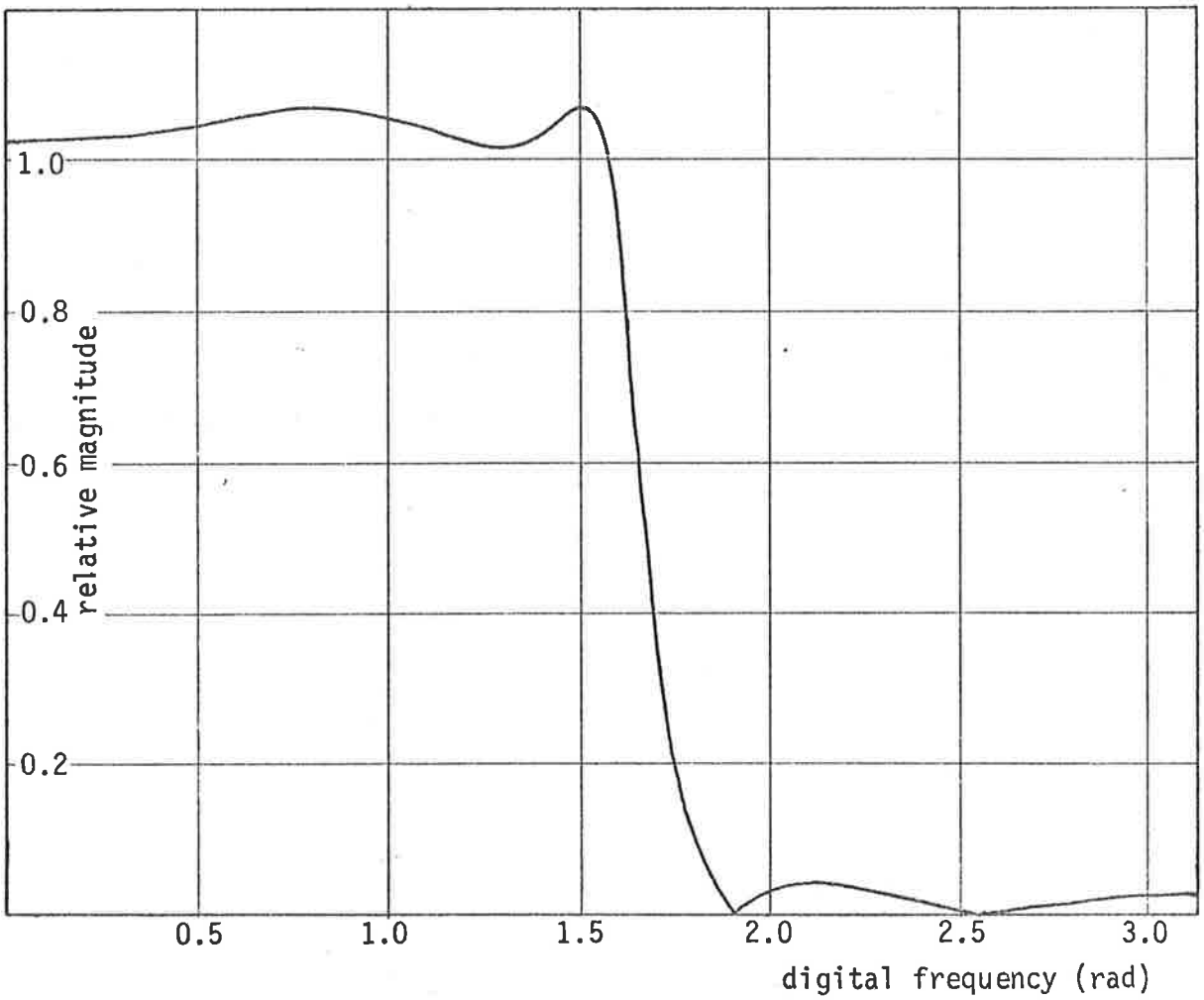


FIGURE 5.14 Example D. Magnitude Response of Two-Section Lowpass Filter, Designed by Least-squares Optimization

(42-point nonuniform discrete frequency set)

least squares design can achieve a response which is close to equiripple. The group delay response of this filter is shown in Figure 5.15, together with the group delay resulting after cascading a three-section all-pass equalizer designed by a unit-weighted least squares optimization over the passband frequency set. The coefficients of the equalizer also appear in column A of Table 5.41 (Sections 3, 4 and 5). The coefficients of the resulting five-section filter constitute the starting parameter vector for the simultaneous magnitude and group delay optimization.

By trial-and-error it was determined that a reasonable error tradeoff was obtained when all magnitude response weight factors were taken as 8.0 (that is, at each of the 42 frequencies of equations (5.27) and (5.28)) and all group delay weights were taken as 1.0 (at the passband frequencies of (5.27) only). In fact, with these values the final magnitude response is slightly better than that of the original filter while the group delay distortion in the passband is greatly reduced. Final response curves are presented as Figures 5.16 and 5.17, and coefficients in column B of Table 5.41. Different tradeoffs may be made by suitable adjustment of weight factors.

There is one further complication in the definition of this problem. When the optimization was attempted exactly as stated above, a solution was obtained which appeared to fit the specification very well indeed. However, it happened that an extremely lightly damped pole pair had been placed in the so-called "transition" band where no target response had been specified. This of course renders the structure useless as a low-pass filter. Accordingly, the objective function was modified

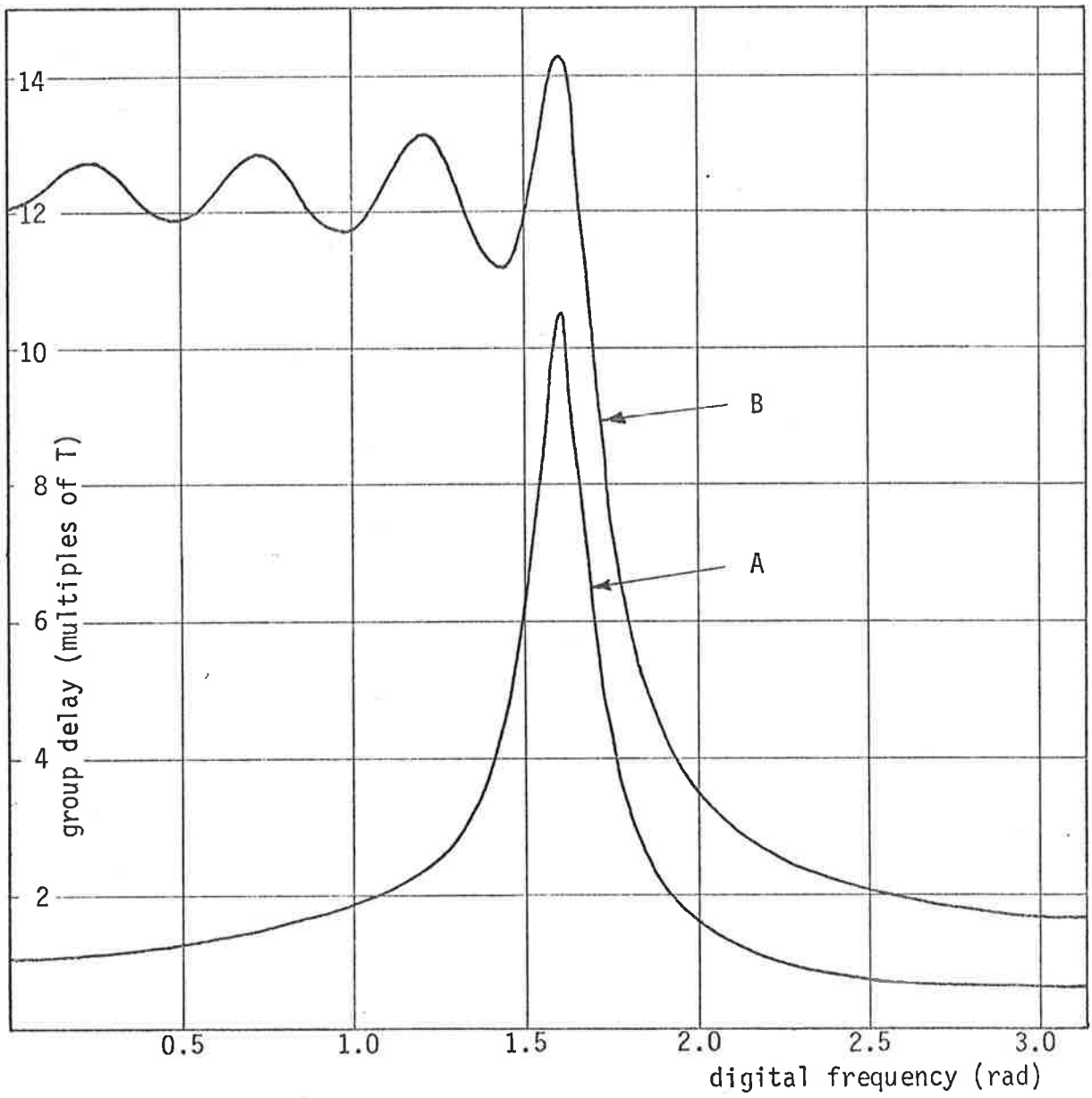


FIGURE 5.15 Example D. Group Delay Response of Lowpass Filters

(A) 2-section magnitude-optimized filter

(B) filter of (A) cascaded with 3-section allpass equalizer

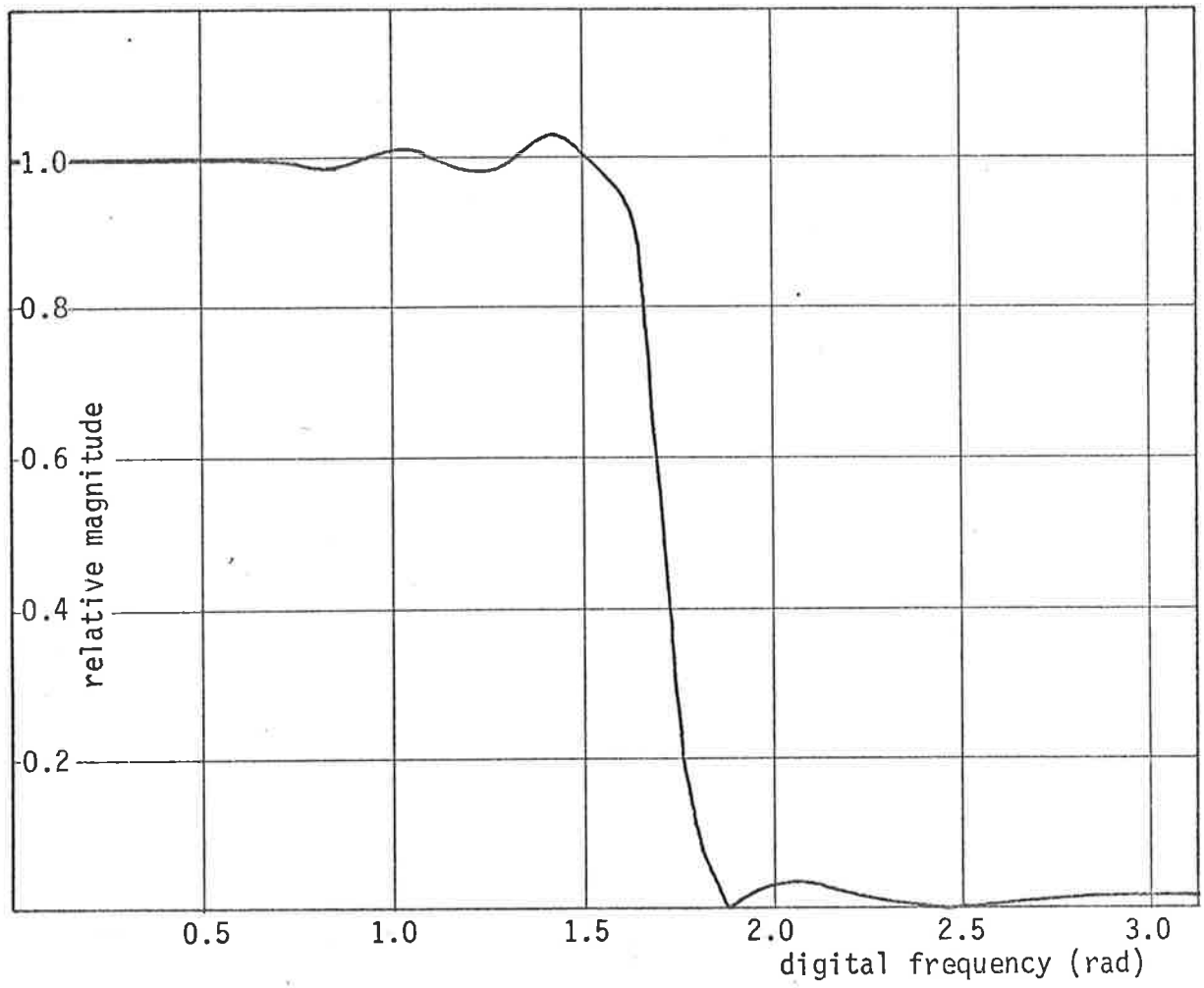


FIGURE 5.16 Example D. Simultaneous Magnitude and Group Delay Optimization. Magnitude Response at Final Solution

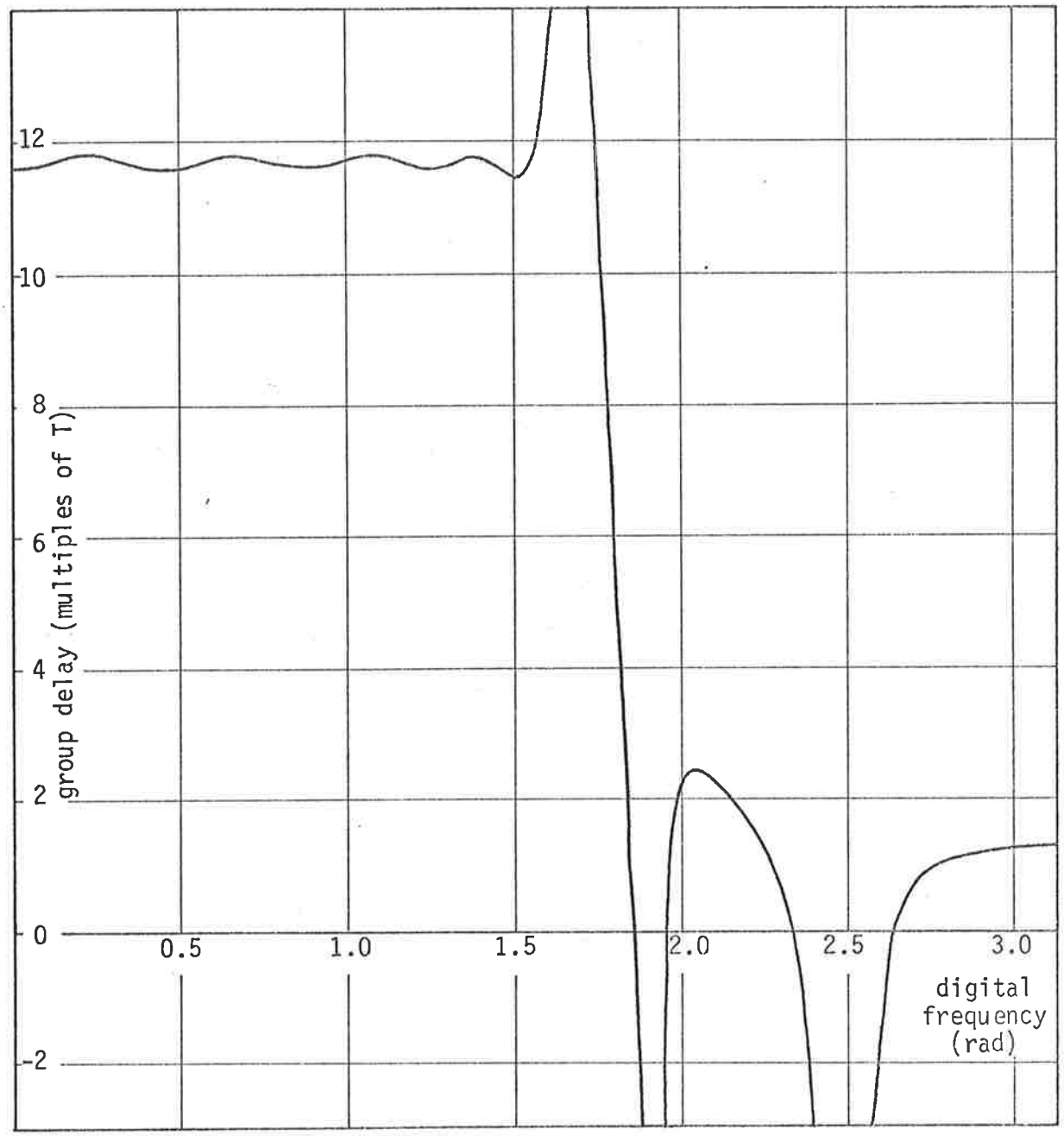


FIGURE 5.17 Example D. Simultaneous Magnitude and Group Delay Optimization. Group Delay Response at Final Solution

by the addition of plausible target magnitude values at five points in the transition band, as follows:

$\theta = 1.61$ mag = 0.95

$\theta = 1.66$ mag = 0.80

$\theta = 1.71$ mag = 0.51

$\theta = 1.76$ mag = 0.24

$\theta = 1.81$ mag = 0.08.

The weight factors applied at these frequencies were also 8.0. It is in fact when the objective function is augmented in this manner that the results of Figures 5.16 and 5.17, and Table 5.41, are obtained.

This matter just discussed illustrates a difficulty which often surfaces when complete design automation is attempted. If the objective function is badly formulated, the results obtained can be quite misleading. In the design of filters such as this, the optimization should probably be considered just as one phase of a human-machine interactive process.

5.11.2 Results

A representative selection of solution methods were applied to the 20-dimensional simultaneous optimization problem. All except CG methods converged to the same solution, which had an associated function value of 1.285544, as compared with 19.53 for the all-pass equalized filter. Convergence was deemed to have occurred when all components of both $\mathbf{g}$ and $\mathbf{p}$ were less than 10^{-3}. The CG methods were run to 600 iterations without the convergence criterion being met, but the function value had been reduced to 1.321 (for the AT-QI search) and to

1.287 (for the PQI search) with all coefficients being quite close to the optimal values. No real roots and so no pairing changes occurred with any method.

The results are presented in Table 5.42. Since for this problem only one starting point was used with each method, the performance could be evaluated both in terms of function evaluation time and total execution time. As the figures of Table 5.42 show, the evaluation time swamps the "overheads" so that the ranking of the algorithms is virtually the same with either measure. The one exception is, predictably, the Greenstadt SD method, which required eigensystem analysis (using more effort than all other operations combined).

As with other examples the computing "labour" is quoted relative to that of the MQ-U SD method with AT-QI line search, which was the third best method tested. The somewhat better performance of MQ-U with PQI was due to the need for two fewer iterations,and is probably fortuitous. Other SD methods were also efficient, as would be expected from the results for earlier examples. GM and FF methods, which are again worse than MQ-U, rate better in terms of total time than of evaluation time because they always require only one matrix factorization per iteration. With MQ-U (AT-QI) an average of 1.3 factorizations per iterations had to be abandoned after the matrices were proven indefinite.

The performance of GN methods is interesting. The line search version (MQ-U with AT-QI) was much more efficient than any other method tested, suggesting that in this example there are no problems associated with large residuals. However, TM-S-G was disappointing and the performance of TM-U-G was appalling. Further investigation (using the detailed printout facilities of

Method	No. of Iterations	No. of Pre-Newton Iterations	No. of Auxiliary Evaluations	Function Evaluation Time		Total CP Time	
				Actual (s)	Relative*	Actual (s)	Relative*
(a) SD methods							
MQ-U, FAV-CF	18	8	32	17.1	1.19	18.7	1.71
MQ-U, AT-QI	15	8	30	14.4	1.00	16.0	1.00
MQ-U, PQI	13	8	50	13.5	0.94	14.9	0.93
GR-U, AT-QI	15	11	35	14.6	1.01	36.8	2.30
GM, AT-QI	16	10	58	16.5	1.15	17.2	1.08
FF, AT-QI	21	16	61	21.0	1.46	22.2	1.39
TM-U-N	18	10	24	16.8	1.17	18.8	1.18
TM-S-N	16	9	18	14.8	1.03	16.8	1.05
(b) GN methods							
MQ-U, AT-QI	22	-	56	11.2	0.78	12.8	0.80
TM-U-G	140	-	357	71.5	4.97	86.7	5.42
TM-S-G	33	-	79	16.6	1.15	20.9	1.31
(c) QN methods							
DFP, AT-QI	189	-	416	50.3	3.49	55.6	3.48
DFP, PQI	68	-	484	32.2	2.24	33.7	2.11
DFP, LM (ε=1.0)	65	-	524	33.4	2.32	35.0	2.19
BFGS, FAV-CF	82	-	294	26.6	1.85	27.6	1.73
BFGS, AT-QI	69	-	200	20.4	1.42	21.7	1.36
BFGS, PQI	67	-	304	24.4	1.69	25.7	1.61
SR1-M, AT-QI	71	-	223	21.7	1.51	23.1	1.44
(d) CG methods							
CG-A, AT-QI, α-pass	600	-	1210	155	10.8	162	10.1
CG-A, PQI, α-pass	600	-	2949	228	15.8	235	14.7

* relative to values for SD method MQ-U, AT-QI.

TABLE 5.42 Results of Optimization Tests, Example D

the general test program) showed that in fact the fast "Newton-like" convergence rate had not materialized with any of the GN algorithms. The initial performance of all methods was good, the function value being reduced below 1.30 in 11 iterations with both TM methods and in 9 iterations by MQ-U with AT-QI. However, the subsequent convergence to the minimum was only at a "steepest-descent" rate. With both TM methods a non-zero value of the Marquardt parameter was needed to gain a function decrease right up to the final iteration. With the line search version the expected "ideal" steplength $\alpha = 1$ was seldom acceptable. Progress towards the minimum was, in all three cases by an inefficient "zig-zag" path. Thus it seems that the line search algorithm was able to meet the convergence criterion after only 22 iterations solely through good luck.

The "McKeown parameter τ/μ (see Section 3.4.12) at the optimal solution was calculated as 2.0×10^3. Although with example B (Section 5.9) a value somewhat larger than this (9.2×10^3) did not seem to lead to convergence problems, the value is in the range for which McKeown (1975) has observed poor performance.

The findings of this section confirm that GN methods have an unfortunate unreliable characteristic which is significant in practice.

The best QN method (BFGS) again came within a factor of 1.5 of the time performance of the best SD method, being slightly better than SR1-M and much better than DFP. AT-QI was the best line search with BFGS, while DFP benefited from the more accurate PQI search.

The CG methods can be said to have produced an acceptable

solution, but their execution time requirement makes them uncompetitive.

5.12 Example E - Time Domain, Synthetic Signal

5.12.1 General

This example and the next deal with the synthesis of a digital filter to match a given target impulse response, as was considered in Section 4.7. As such a process may have application to real-time system identification problems, the time efficiency of the optimization methods is of perhaps greater importance here than when applied to off-line filter "design" problems.

For example E the target impulse response was obtained by exciting a two-zero four-pole digital filter (whose parameters are given in Table 5.43, columns (A) and (B)) with a unit impulse. The filter output was then corrupted by the addition of a small-amplitude noise-like component which consisted of a fixed amount of 0.01 which was added to or subtracted from individual output samples on a "random" basis. The resulting target response of 31 points' duration is listed in Table 5.44. The optimization process used a unit-weighted least-squares "formulation B1" criterion applied to samples 2 to 31. The model fitted had the same configuration and the same complexity as the generator, and so N = 6, and a very good fit is expected.

Thirty starting points were used, generated in accordance with Section 5.2.2; that is, the poles and zeros were initially placed at "random" locations within the unit circle. The

(A) Poles and zeros of signal	(B) Coefficients used to generate signal	(C) Coefficients found (solution vector)
Two real zeros $z_1 = -0.635971$ $z_2 = +0.578804$	$a = 0.057167$ $b = -0.368103$	$a = 0.058279$ $b = -0.362211$
Complex conjugate pole pair $r = 0.666980$ $\phi = \pm 2.186282$	$c = 0.770168$ $d = 0.444862$	$c = 0.769033$ $d = 0.445077$
Complex conjugate pole pair $r = 0.93$ $\phi = \pm 1.244369$	$c = -0.596430$ $d = 0.864900$	$c = -0.595955$ $d = 0.865345$

TABLE 5.43 Details of Generating System (A,B) and Solution (C) for Example E

Time index, n	Sample value, h(n)
1	1.0000000000
2	-.1156699670
3	-1.1957868403
4	-.0980149892
5	.6928774269
6	.4792153955
7	-.1793129603
8	-.6241959185
9	-.1990071459
10	.4499489608
11	.4145827427
12	-.1350225276
13	-.4306779858
14	-.1536148918
15	.2875802248
16	.3026393464
17	-.0694032462
18	-.2992045032
19	-.1215566030
20	.1937761875
21	.2145460824
22	-.0391841023
23	-.2030007736
24	-.0957113010
25	.1234790942
26	.1566024976
27	-.0180016822
28	-.1462429895
29	-.0693455109
30	.0849345105
31	.1067009648

TABLE 5.44 Target Impulse Response for Example E

derivatives of the objective function were calculated by employing the recursive filter structures derived in Section 4.7.2, and so zeros (as well as poles) had to be constrained to lie inside the unit circle during the entire optimization process. As expected, the need to observe constraints led to many instances of failure to converge, but there was a sufficient number of successes to allow the relative performance of the algorithms to be evaluated.

The optimal coefficients are listed in Table 5.43 column (C). The slight differences between these and the generator coefficients are due to the effect of the additive noise. The minimum value of the objective function was 2.145424×10^{-4}. Convergence was deemed to have occurred when all components of the $\mathbf{p}$ vector became less than 10^{-3}, with the Euclidean norm of the $\mathbf{g}$ vector also being less than 10^{-3}.

5.12.2 Second-Derivative Methods

The results obtained for SD methods are presented in Tables 5.45 to 5.48. Many of the conclusions to be drawn are similar to those of earlier examples, and only the more important will be repeated here. The MQ-U and MQ-S approaches again outperform their competitors. The two GR methods and GM rank second (although GM would be more efficient due to lower overheads), while FF is a rather poor last. Interestingly, Table 5.45 indicates that the most efficient (MQ) algorithms also converge more often (particularly when compared with GM). A possible explanation for this runs as follows. We may consider that each local minimum of any objective function has a "capture area", which is that region of parameter space whose points may be

SEARCH DIRECTION METHOD	LINE SEARCH METHOD							
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C
GR-S		22	24	24	24			
GR-U	20	22	22	21	22	21	20	20
MQ-S		29	27	30	28			
MQ-U	29	29	27	26	26	30	29	29
GM	18	13	15	14	13	14	17	16
FF	21	20	22	19	21	21	19	18

TM-U-N: 29

TM-S-N: 28

TABLE 5.45 Example E. Second-derivative Methods

Number of starts converging (out of 30)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD							
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C
GR-S		13.8	12.1	11.3	11.9			
GR-U	12.3	13.8	12.5	11.7	11.8	13.1	13.9	12.0
MQ-S		13.2	10.9	9.7	9.5			
MQ-U	11.6	12.1	10.4	9.3	9.5	11.3	12.4	11.1
GM	13.0	14.3	11.7	10.6	10.3	13.7	15.6	13.3
FF	16.2	19.7	15.6	14.2	14.7	15.2	17.9	16.6

TM-U-N: 13.0

TM-S-N: 13.9

TABLE 5.46 Example E. Second-derivative Methods

Average Number of Iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD							
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C
GR-S		23.3	28.3	43.5	57.5			
GR-U	24.6	22.9	29.8	45.2	57.9	35.6	26.4	31.7
MQ-S		17.4	21.4	35.0	45.6			
MQ-U	16.0	15.4	20.2	32.2	44.1	16.8	17.4	24.0
GM	44.6	29.0	32.3	46.1	52.2	35.9	28.9	34.0
FF	47.7	36.7	41.8	60.2	71.3	37.8	31.0	41.7

TM-U-N: 15.3

TM-S-N: 16.9

TABLE 5.47 Example E. Second-derivative Methods

Average number of auxiliary function evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD							
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)	FAV-CF-C	FAV-CI-C	AT-CI-C
GR-S		1.29	1.21	1.31	1.49			
GR-U	1.19	1.29	1.26	1.35	1.49	1.36	1.33	1.24
MQ-S		1.19	1.05	1.10	1.19			
MQ-U	1.05	1.08	1.00	1.04	1.17	1.04	1.13	1.09
GM	1.45	1.39	1.22	1.28	1.32	1.41	1.49	1.36
FF	1.72	1.88	1.62	1.69	1.85	1.55	1.68	1.69

TM-U-N: 1.15

TM-S-N: 1.24

TABLE 5.48 Example E. Second-derivative Methods

Relative Labour in Function Evaluation

connected to that local minimum by paths which are entirely "downhill." In other words, it is the set of points from which a steepest-descent process employing infinitesimally short steps would be expected to converge. Owing to the nature of the present problem, the fairly small dimensionality, and the very good fit of the filter model at the optimal solution, it seems reasonable that the capture area of the global optimum is rather large; in fact, probably all of the 30 starting points are within it. If there are any other local minima their capture areas are likely to be relatively small. Although any search method employing *finite* steps has the ability to overstep a ridge, thus leaving one capture area for another, and although this may occasionally be beneficial, it seems intuitively desirable that this should not happen very often. The bias towards the steepest-descent direction introduced in the MQ methods is likely to lead to less ridge-jumping than more arbitrary descent directions, particularly those of the GM method. If it is given that the starting points *are* within the capture area of the optimal solution, then the MQ methods *should* converge more often (or, rather, diverge *less* often).

With all SD line-search methods the AT-QI search is the best. The use of curvature information to predict a step-length is not worthwhile. The true Marquardt methods are reliable (as the above reasoning concerning "capture area" would predict), but are again not as efficient as the best line search methods.

5.12.3 Gauss-Newton Methods

Results for GN methods are in Table 5.49. The true Marquardt methods are much more reliable and more efficient than the line search variants. As usual, the TM methods outperform even the best SD methods by a significant margin. Of course, in this problem the optimal model is so good that there are no convergence problems associated with large residuals. The "McKeown factor" is only 0.9185, indicating that even the classical undamped Gauss-Newton algorithm would probably converge. An interesting departure from earlier findings is that the line search algorithns seem to benefit from the added accuracy of the PQI search.

5.12.4 Quasi-Newton and Conjugate Gradient Methods

Results for QN and CG methods are in Tables 5.50 to 5.53. The CG algorithms used the "$\alpha = 1$" variants which proved more efficient than "α-pass" for the AT search. No CG method approached the QN or other methods in efficiency.

The BFGS and SR1-M methods (with AT-QI or FAV-QI) were for this example only slightly less efficient than the SD algorithms. However, all QN methods seem to diverge more often than the TM and MQ-U SD algorithms, perhaps due to "ridge-jumping" with overly large steepest-descent steps on early iterations.

	METHOD								
			GR-S			MQ-S			
	TM-U-G	TM-S-G	FAV-QI	AT-QI	PQI	FAV-QI	AT-QI	PQI	LM (ε=1.0)
Number Converging (out of 30)	29	28	15	14	15	14	14	15	16
Average Number of Iterations	11.2	11.2	17.7	17.3	11.6	17.1	17.3	11.6	13.3
Average Number of Auxiliary Evaluations	23.3	22.1	44.3	60.1	43.8	42.2	60.1	43.8	72.2
Relative* Labour	0.82	0.81	1.38	1.51	1.05	1.32	1.51	1.05	1.43

* Labour spent in function evaluation, relative to MQ-U, AT-QI, SD method

TABLE 5.49 Example E. Gauss-Newton Methods

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)
DFP	12	20	22	23	22
BFGS	19	23	24	23	24
SR1-M	18	24	26	23	22
CG-A		27*	30	29	29
CG-B		29	30	30	30

* only 27 starts run

TABLE 5.50 Example E. Quasi-Newton and Conjugate Gradient Methods

Number of Starts Converging (out of 30)

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)
DFP	44.2	79.5	32.6	22.1	21.0
BFGS	28.1	28.5	22.2	20.1	20.3
SR1-M	27.9	27.8	25.6	21.1	21.2
CG-A		132.8	78.8	75.6	64.4
CG-B		134.2	71.4	63.9	65.6

TABLE 5.51 Example E. Quasi-Newton and Conjugate Gradient Methods

Average Number of Iterations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD				
	FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)
DFP	73.1	96.3	73.8	119.3	128.9
BFGS	88.3	51.3	58.4	86.5	103.0
SR1-M	67.7	44.9	64.2	98.2	115.2
CG-A		528.1	319.8	422.0	416.3
CG-B		512.7	285.6	351.9	418.7

TABLE 5.52 Example E. Quasi-Newton and Conjugate Gradient Methods

Average Number of Auxiliary Function Evaluations

SEARCH DIRECTION METHOD	LINE SEARCH METHOD: FAV-CF	FAV-QI	AT-QI	PQI	LM (ε=1.0)
DFP	1.76	2.80	1.50	1.71	1.79
BFGS	1.54	1.17	1.10	1.34	1.51
SR1-M	1.33	1.09	1.24	1.48	1.65
CG-A		8.39	5.05	6.00	5.69
CG-B		8.27	4.53	5.03	5.74

TABLE 5.53 Example E. Quasi-Newton and Conjugate Gradient Methods

Relative Labour Spent in Function Evaluation

5.13 Example F - Time Domain, Natural Signal

5.13.1 General Discussion

Example F illustrates the use of time-domain optimization in the pole-zero modelling of a "natural" signal, that is, one not necessarily fitting any digital filter model. The target impulse response consists of a minimum-phase estimate of the human vocal tract impulse response during an utterance of the nasal consonant/m/. The curve was produced by homomorphic processing of the speech signal (to be discussed in Chapter Seven) and is shown in Figure 5.18. The work reported here does not represent a serious attempt to generate realistic models for speech sounds. The target response derived from speech merely provides a convenient example of a signal not derived from a simple known model.

The first choice to be resolved concerns the complexity (number of poles and zeros) of the model to be fitted. Any serious attempt to automate the system modelling process would have to develop criteria for making such a selection, and this would involve an analysis of the adequacy of the model for its intended purpose. In the case of speech this may involve such matters as the intelligibility of speech re-synthesized using the model. As the main object here is the comparison of optimization methods, a trial-and-error approach was adopted using visual comparison of achieved and target impulse responses.

Figures 5.19 to 5.21 repeat the target impulse response of Figure 5.18 (points not shown were zero-weighted in optimizing) and show the fits obtained with several simple models. In Figure 5.19 the very obvious low frequency damped oscillation is well modelled by a pole pair, but the model has insufficient complexity

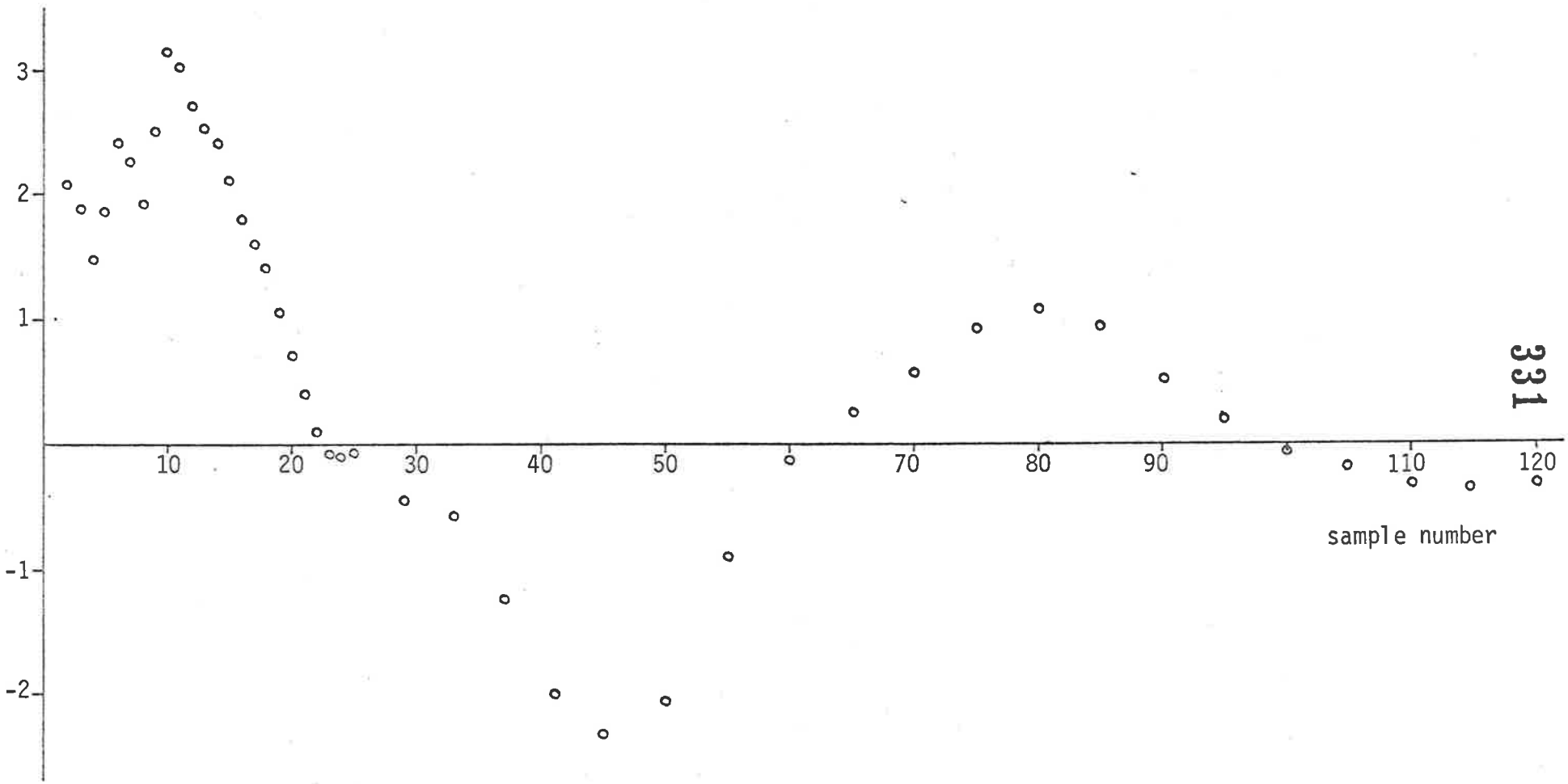


FIGURE 5.18 Example F. Target Impulse Response

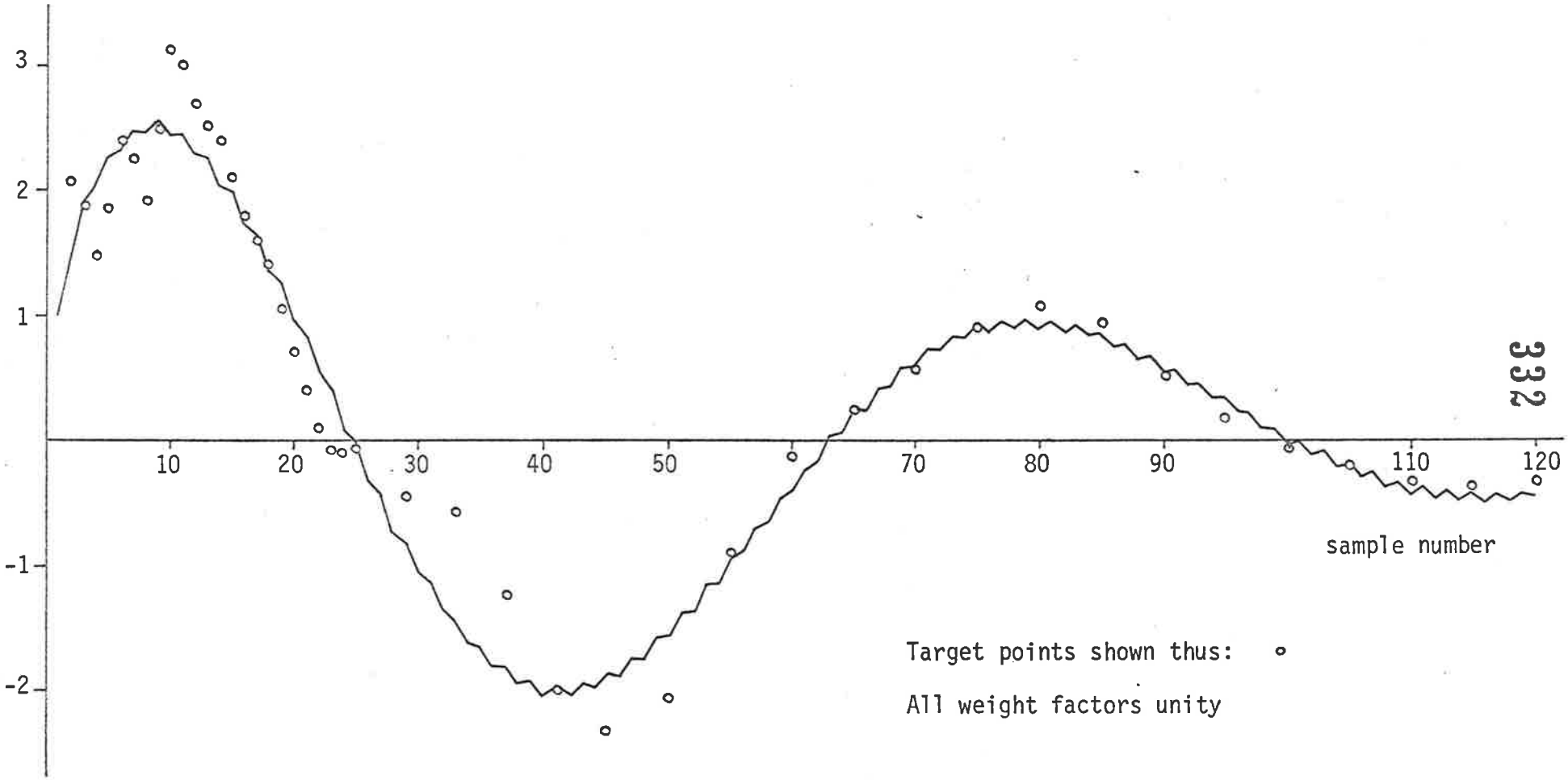


FIGURE 5.19 Example F. Impulse Response Matching. 4-Pole 4-Zero Fit

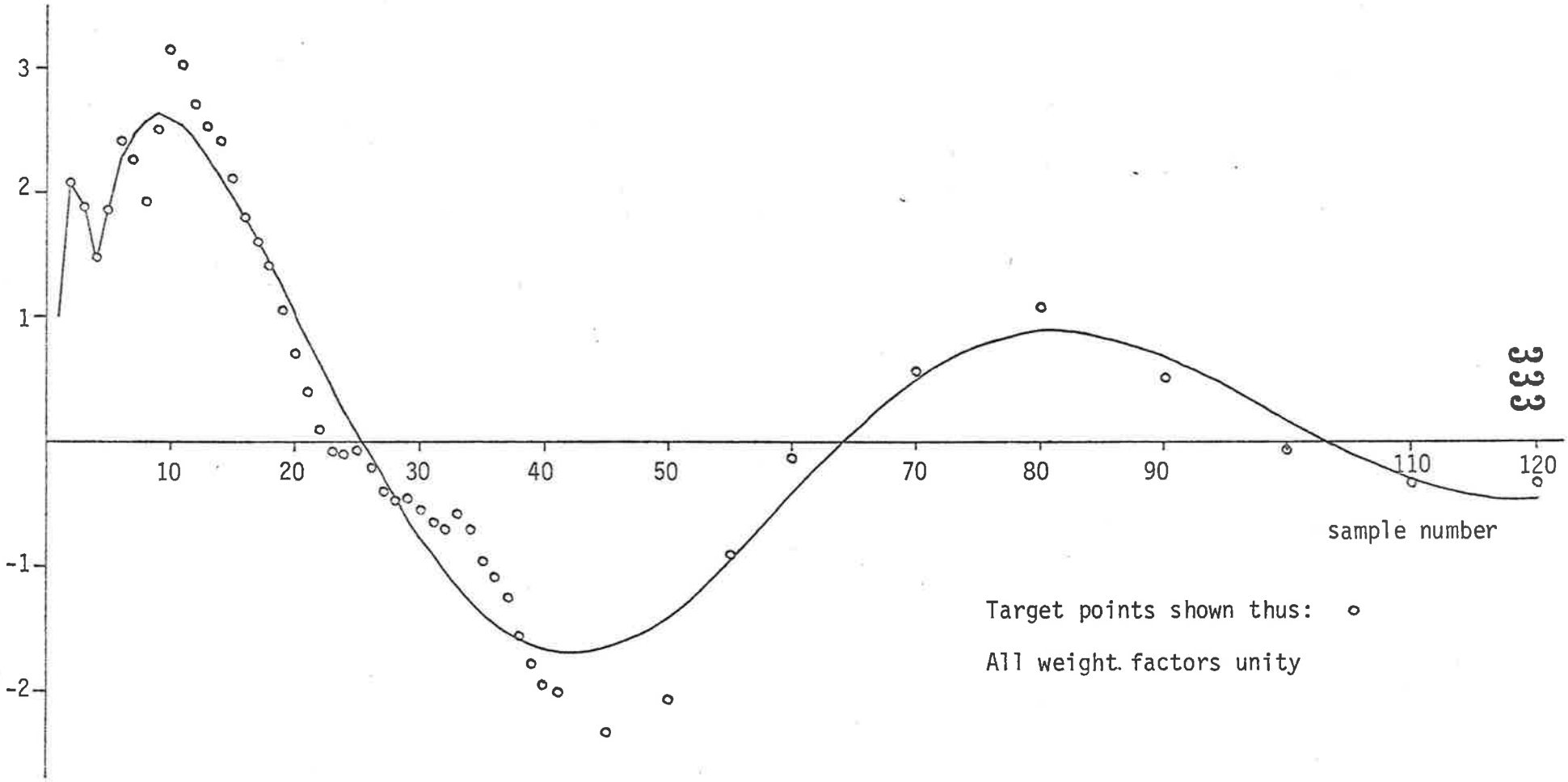


FIGURE 5.20 Example F. Impulse Response Matching. 4-Pole 6-Zero Fit

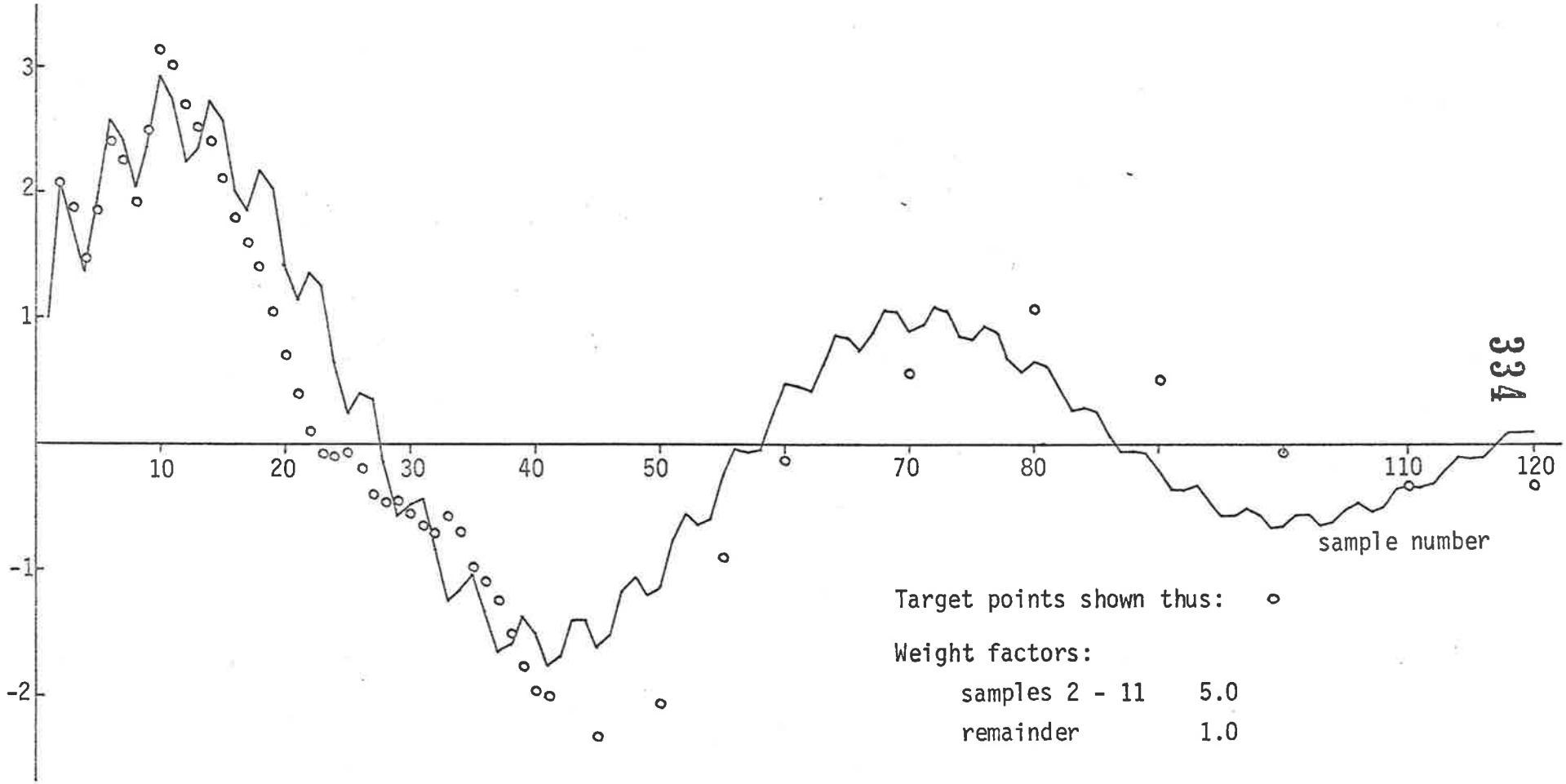


FIGURE 5.21 Example F. Impulse Response Matching. 4-Pole 6-Zero Fit

to match the complicated behaviour of the early samples. The oscillatory behaviour of alternate samples which persists up to (and beyond) sample 120 is due to a very lightly damped pole on the negative real axis. Its occurrence is annoying because such behaviour is not a characteristic of the signal being modelled. It is a case of a bad choice of model leading to quite strange results, which unfortunately often happens with optimization techniques. Figure 5.20 illustrates the effect of adding another pair of zeros and including more samples in the time index range 25 - 40 in the specification. The low-frequency oscillation and the behaviour at samples 1 to 6 are well modelled, and the spurious real-axis pole has disappeared, but the fit at samples 7 to 12 in particular could be improved. Figure 5.21 shows the effect of increasing the weighting on the early samples (2 to 11) to 5.0, which is to force this part of the response to be modelled by a complex pole pair. The resulting fit is good in this region, but the pole pair is very lightly damped and its effect persists to much later times where it has no correspondence with any characteristic of the real signal.

Several attempts to find other models failed because non-minimum phase zeros were called for. Since the gradient computation scheme used requires zeros to remain minimum phase, the optimization runs had to be abandoned whenever the zeros were driven into the unit circle. This could be a major failing of this method for modelling general signals (even when, as in the present instance, they are known to arise from minimum-phase considerations). For this reason, it is recommended that a general-purpose program

should use the alternative (but more time-consuming) scheme for gradient calculation. This was derived in Section 4.7, and involves omitting one filter section for each gradient component rather than cancelling its effect with an appended recursive section.

5.13.2 Problem Definition

Figure 5.22 shows the impulse response of the model finally selected for the studies of optimization methods. There are 5 zero pairs and two pole pairs. One pole pair adequately models the sustained oscillatory aspects of the signal whilst the zeros and the second pole pair (all complex apart from one pair of zeros) look after the shaping for low time index.

The dimensionality of this optimization problem (N) is 14. In a trial involving 17 starting points generated in the random fashion of Section 5.2.2, four cases of convergence to the optimal solution were recorded, and the remainder of the runs were abandoned when the minimum-phase constraint was encountered. This illustrates that it *is* feasible to generate such models from arbitrary starting points. However, it shows also that this would *not* be a robust technique for on-line applications. To compare the optimization algorithms it was decided to use a set of starting points in the near vicinity of the optimal solution. This may not be unreasonable because in many applications a good estimate may be available from an earlier "frame" of data or by using a method such as Shanks' (1967) or Kalman's (1958). In any case it should not upset the conclusions regarding the convergence of algorithms. Nine stable

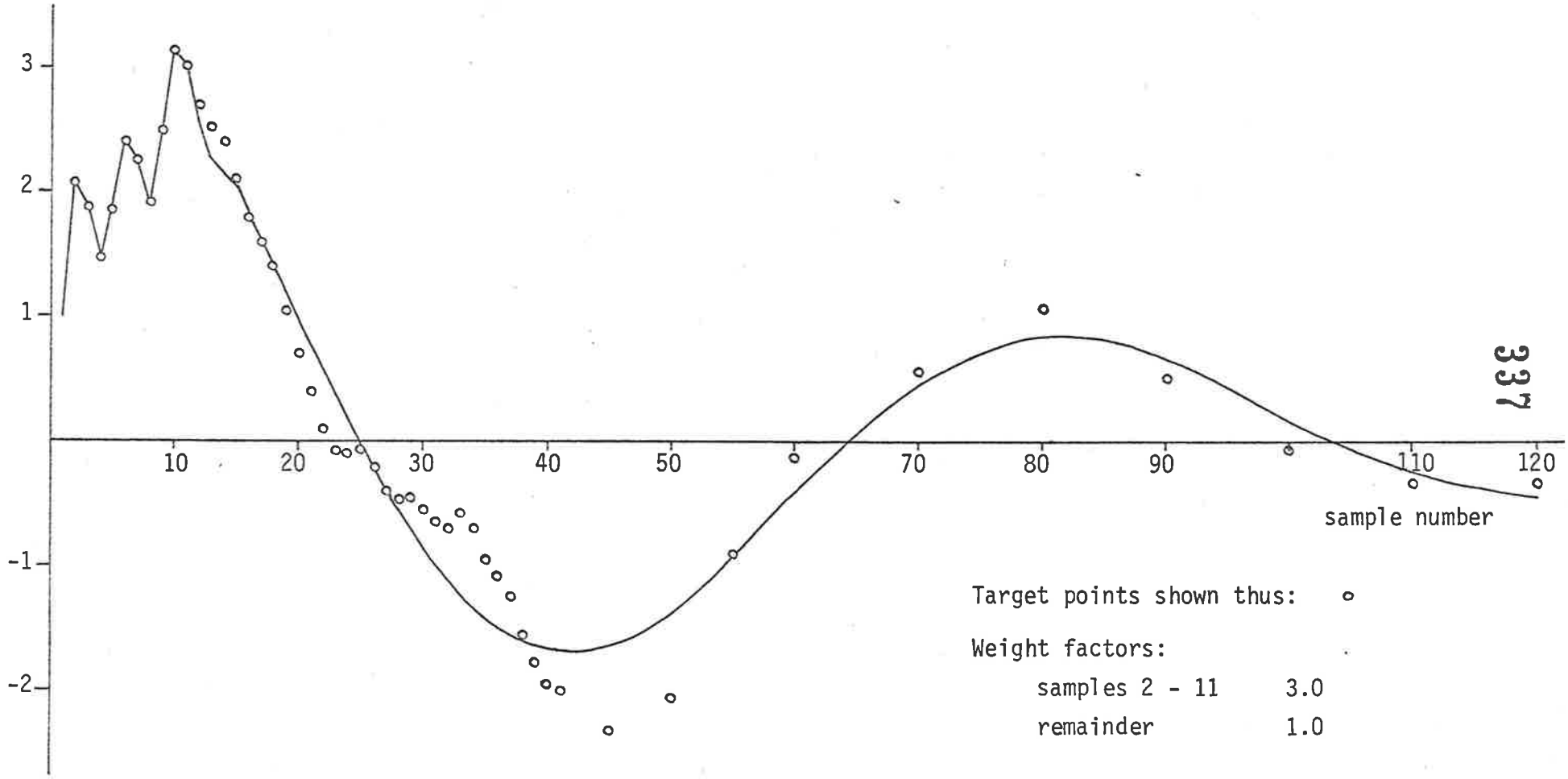


FIGURE 5.22 Example F. Impulse Response Matching. 4-Pole 10-Zero Fit

(and minimum phase) starting points were in fact used, generated as described in Section 5.2.3. "Centre values" of parameters were those at the solution. All "maximum deviations" were 0.05 except that the coefficients determining the lightly-damped poles ($r = 0.982$, $\phi = 0.079$) were given maximum deviations of 0.002.

The value of the objective function at the solution was 4.03. Convergence was deemed to have occurred when all components of $\mathbf{g}$ and $\mathbf{p}$ vectors were less than 10^{-3}.

5.13.3 Results

The results obtained are presented in Table 5.54, and are much as would be expected from earlier sections. All algorithms converged in all cases except for one case of failure with GR and FF SD methods and three with GM. Thus MQ-U seems the most robust, and GM the least robust, of the SD methods, confirming the findings with example E. MQ-U is also the most efficient of the SD methods, three different line search variants giving closely comparable figures.

The GN methods tested (both TM and line search) are "Newton-like" in behaviour and so surpass the SD methods in efficiency by a considerable margin.

The time requirements of the best QN methods (BFGS and SR1-M with AT-QI) again hover around 130% relative to the MQ-U SD method. QN methods are thus quite an attractive alternative to SD methods. In fact, if the more complicated scheme for derivative calculation (to allow for nonminimum phase zeros) were necessary, the SD methods would suffer disproportionately, and it is likely that QN methods would become more efficient.

Method	Number Converging (out of 9)	Average No. of Iterations	Average No. of Pre-Newton Iterations	Average No. of Auxiliary Evaluations	Relative Labour
(a) SD methods					
MQ-U, FAV-CF	9	10.4	4.8	11.2	0.98
MQ-U, AT-QI	9	10.1	4.0	16.9	1.00
MQ-U, PQI	9	9.2	3.8	28.7	1.02
GR-U, AT-QI	8	13.1	5.6	28.0	1.35
GM, AT-QI	6	14.2	5.7	33.5	1.48
FF, AT-QI	8	14.8	7.9	33.6	1.54
TM-U-N	9	11.3	5.4	16.8	1.10
TM-S-N	9	10.8	5.3	20.1	1.09
(b) GN methods					
MQ-U, AT-QI	9	10.1	-	12.2	0.62
MQ-U, PQI	9	9.3	-	22.2	0.66
TM-U-G	9	11.6	-	11.7	0.69
TM-S-G	9	11.3	-	11.4	0.67
(c) QN methods					
DFP, AT-QI	9	50.3	-	115.0	1.90
DFP, PQI	9	25.7	-	148.1	1.69
BFGS, AT-QI	9	28.0	-	94.9	1.31
BFGS, PQI	9	25.8	-	144.6	1.67
SR1-M, AT-QI	9	29.2	-	93.8	1.32

TABLE 5.54 Results of Optimization Tests, Example F

No tests of CG methods were performed with this example.

5.14 Overall Conclusions Regarding Optimization Methods

Many findings relating to the relative performance of the optimization methods tested have already been reported. Many of these are uniformly valid over the entire spectrum of filter design examples. In addition, some phenomena have occurred frequently enough to suggest general guidelines for the selection of an optimization algorithm:

(a) if the best speed performance is truly important, together with reliability, it *does* appear to be worthwhile evaluating second derivatives, at least in frequency-domain problems.

(b) the best SD method to use is probably one of the MQ-U type with line search. This has consistently performed better than other alternatives. In several examples where several starting points were used (B, E and F), MQ-U achieved convergence where others failed, owing to its bias towards the steepest descent direction. In addition to reliability, MQ-U has shown a speed advantage. Algorithms of the Greenstadt type have sometimes appeared better (examples B, C) in terms of function evaluation time alone, but their requirement for eigensystem analysis would still put them at a disadvantage. And in examples A and D GR methods were no better than MQ-U, and in E and F they were worse. GM and FF methods were almost always worse in speed, and often significantly less reliable. GM and FF have the advantage of requiring only

one matrix factorization per iteration, but MQ-U can be programmed efficiently enough that other factors are more important. MQ-S was compared with MQ-U in four examples. In three (A, C and E), MQ-U was more efficient and in the fourth the result was inconclusive. It is worth mentioning again that the MQ-U method as tested passes the successful value of the Marquardt parameter β on to the next iteration, except that if the *first* value tried succeeded in producing a positive definite matrix, 0.25 times this value is passed. Within an iteration, β is multiplied by 4.0 whenever $\mathbf{H} + \beta \mathbf{I}$ is indefinite. These updating factors are merely guesses, and it may be possible to obtain even better performance by selecting alternative values.

(c) The line search method which performed consistently best with SD algorithms was AT-QI. That is, it seems worth making one additional prediction for a minimizing value of α when the first acceptable value has been found, but not worth going to a more accurate line search. Whether there is any benefit from using curvature information at $\alpha = 0$ to aid the line search is uncertain, but if so it is sure to be small. An analysis of some tests with example A suggests that the best policy may be to try $\alpha = 1$ first, and then to use the curvature information to enable *cubic* rather than quadratic interpolation for the additional try, or to continue the search for an acceptable value of α if still necessary. The use of Fletcher and Freeman's (1977) "target function reduction" and quartic polynomial does not seem to be worthwhile.

If for some reason it is desired to use a very simple line search with an SD method, FAV-CF with a reduction factor of about 0.6 performs almost as well as AT-QI.

(d) True Marquardt SD methods are also reliable and avoid the need for any line search. However, they incur a speed penalty of about 10 - 20% (for TM-U-N) compared with the best line search methods in terms of function evaluation alone, and also have higher overheads due to the need to solve several sets of linear equations per iteration. TM-U-N was better than TM-S-N in four out of six examples.

(e) Gauss-Newton methods have achieved the highest speeds of all. In all cases except example C, the TM versions are more efficient than the line search variants, and the necessary solution of more linear equations would be worthwhile. In addition, the line search methods lack robustness, a failure mode involving small steps almost orthogonal to the gradient being observed with examples A and B.

However, a most serious defect is present with all the GN methods, in that the existence of non-zero residuals at the solution can in quite routine problems (seen in examples B and D) cause the convergence rate to be extremely poor. The unfortunate conclusion is that the "traditional" GN methods described in this chapter should not be used, except possibly in cases of a known good fit between model and target response. As mentioned in Section 3.4.12, however, there have been attempts to overcome this problem by combining the best features of GN and

QN methods (e.g. Dennis, 1973). Use of such a method *could* be advantageous.

(f) The best quasi-Newton methods have been consistently (but not vastly) slower than the SD methods. Except with example C, the BFGS matrix update was the best, being marginally better than SR1-M. Both BFGS and SR1-M were always clearly superior to the simpler SR1 and the very widely-used DRP methods. The best line search method to use with BFGS is either FAV-QI or AT-QI. Although FAV-CF is an acceptable alternative to AT-QI with SD methods, it should not be used with QN methods because (examples A, C, D and E) it can lead to an excessive number of function evaluations on early iterations far from convergence.

The efficiency of BFGS with AT-QI was worse than that of MQ-U with AT-QI by a factor of about 1.4 to 1.6 in the frequency domain examples (A,B,C,D) and only 1.1 to 1.3 in time domain examples (E,F). These figures are for function evaluation time alone, and *overall* figures would place the QN method in a slighly better light because of substantially lower overheads per iteration. The lower time penalties seen with time-domain examples are a direct result of the substantial time needed to generate second-derivative sequences. As mentioned in conjunction with example F, the scheme which was used for this (involving the cancelling of a zero pair by a pole pair) may not be sufficiently general for many problems, which require the treatment of non-minimum phase zeros. Because second-derivative computation using the alternative, general, scheme would be much more

time-consuming again, it seems that QN (BFGS) methods would in fact be the most time-efficient for such time-domain problems.

(g) In the implementation of QN methods where "root reflection" and/or reviews of real root pairing are required, the matrix should be reset to the identity after any such change. For problems of low dimensionality (6, example A) it seemed marginally better to allow reflection and pairing at every iteration, but in more complex (and realistic) problems it is definitely necessary to allow them only at less frequent intervals, perhaps every N iterations (example B).

(h) The conjugate gradient methods are not competitive.

(i) Example C, the optimization of the all-pass group delay equalizer, provides most of the exceptions to the general findings. With the SD methods, MQ-S converges more often than MQ-U, the FF method performs as well as MQ, and the GM method is even better. The line-search GN methods are as good as the TM variants. With QN methods, SR1-M is possibly better than BFGS. Whether these results are fortuitous or whether there is something unusual about the objective function when the parameters are the coefficients of all-pass sections, is not clear. However, none of the effects is sufficiently major to invalidate the general findings.

The conclusions to be drawn from this study may be summarized more tersely, in the form of recommendations regarding the choice of optimization algorithm. The algorithms recommended are MQ-U with AT-QI or FAV-CF (a second-derivative method) and BFGS with AT-QI or FAV-QI (a quasi-Newton method).

The SD method is recommended for frequency-domain problems. However, if the added complexity involved in programming the second derivative evaluation is regarded seriously, the QN method could be used instead, with a speed penalty factor considerably less than 2.

In the *time-domain* problems studied the SD method was again more efficient (but only very slightly) than the QN method. It is, however, anticipated that the "short-cut" procedure used for first and second derivative evaluation will prove insufficiently general for realistic time domain problems. With a more general method for derivative evaluation the QN method would become more efficient than the SD method; it is therefore recommended.

CHAPTER SIX

6. FINITE-WORDLENGTH DIGITAL FILTER DESIGN

6.1 Introduction

In any implementation of a digital filter there is a necessity to represent both data and coefficients as words with a certain (finite) number of bits. The effect of data quantization is (roughly speaking) that noise is added to the desired filter output, while coefficient quantization causes the frequency response (and impulse response) to differ from what the designer would like. The two phenomena are almost entirely separate, although Bogner (1977) has considered a tradeoff in which response accuracy is gained at the expense of a further added noise component. The coefficients are *dithered* (that is, randomly, or systematically, caused to assume two or more values at different times) so that in an "average" sense the coefficient can be thought to possess some desirable intermediate value.

The study of finite-wordlength effects is important for a number of reasons. Finite precision is inherent in any digital representation. However, when a digital filter is implemented on a general-purpose mainframe computer with no real-time requirement the computations are performed with sufficient accuracy that the bad effects are usually negligible. The considerations of short wordlength arise mostly in conjunction with high-speed dedicated equipment, which could be a general-purpose mini - or micro-computer, or (in the highest speed applications) digital hardware specially built for the purpose. When using, say, a 16-bit minicomputer with a hardware 16 by 16-bit multiplier the question may be simply whether 16-bit arithmetic is sufficiently accurate to realize the filter adequately. If, however, the multiplications

had to be performed with a software shift-and-add sequence, there would be a time advantage in shortening the coefficient wordlength because this would reduce the number of passes through the shift-and-add loop. If execution time were critical it would be desirable to determine the *shortest* wordlength which just allowed the filter to meet its specification. In the case of special-purpose hardware the money cost (particularly of multipliers but also of other components) would also be reduced along with data and coefficient wordlengths.

This chapter concentrates on the matter of coefficient quantization, which provides scope for the application of a type of optimization technique, namely, the minimization of a function of several variables under integer constraints on the variables. However, there is a problem resulting from *data* quantization to which another kind of optimization has been applied; namely, what is the best *ordering of sections* in a cascade form implementation of a digital filter? This matter is mentioned briefly in Section 6.2.

Research on the design of digital filters which must be implemented with short coefficient wordlengths has taken two main thrusts. The first is the development of filter *structures* such that the response is little affected by small changes to the coefficients, and thus by the inevitable quantization. The second is the development of *algorithmic techniques* for determining:

a) the particular set of coefficients of a given wordlength (to be used with a filter of a given structure) to best approximate a desired objective, or

b) the smallest coefficient wordlength(s) which will allow a filter of a given structure to meet a specification, and the appropriate values of the coefficients.

It is the algorithmic area which is of most interest here because the techniques may be termed "mathematical optimization methods." Their use is not usually confined to any particular filter structure. Nevertheless, the attainment of a sufficiently good filter response may require *both* a good structure *and* a good discrete optimization scheme, and so the matter of filter structures is treated, in Section 6.3. In addition, there has been some useful analytic work relating to coefficient quantization; this is reviewed in Section 6.4.

Section 6.5 reviews some of the techniques which have been advocated for discrete coefficient optimization. Section 6.6 deals with a new approach to the problem, previously introduced in publications by this author (Smith, 1977, 1979). Both of these papers are reproduced as part of Section 6.6, and further amplifying discussion is presented.

6.2 Optimization and Data Quantization

Reviews of the effects of data quantization are given by Oppenheim and Weinstein (1972) and by Claasen, Mecklebrauker and Peek (1976). The principal arithmetic operations in a digital filter are multiplication by a constant (coefficient) and addition. The exact product of an m-bit data word and a b-bit coefficient requires m + b bits for its representation. If this product were then multiplied by another b-bit coefficient, m + 2b bits would be required. Hence, if the filter contains recursive elements, the required wordlength grows without limit. There is thus a need to quantize (truncate or round) the results of multiplications,

and this introduces a fluctuating unwanted component superimposed on the "ideal" signal. This has usually been modelled as an additive random noise source following the multiplier.

The effect of the *additions* depends in a major way on whether fixed or floating-point arithmetic is being used. In fixed-point, two numbers may be added without incurring any error, provided that the result is not too large to be representable in the given number of bits, ie., provided the register does not *overflow*. The avoidance of overflow in pure fixed-point filters requires that the input signal level be kept below some limit, and so the *dynamic range* of the filter is considerably restricted. The lower limit for the input level is of course set when the input quantization noise due to digital down-scaling or to A/D conversion becomes intolerable.

When floating-point arithmetic is used, dynamic range considerations are usually unimportant due to the large range of numbers representable. However, rounding occurs frequently in the floating-point additions, and is a further source of noise.

When a high-order digital filter is implemented in fixed-point in the cascade form, overflow must be controlled at all internal nodes and the noise-to-signal ratio at the filter output simultaneously kept within reasonable bounds. This has implications relating to the *ordering* of the first- or second-order sections, and the *pairing* (or allocation of poles and zeros to individual sections). The use of floating-point arithmetic can render such considerations of minor importance (Oppenheim and Weinstein, 1972), but digital filters are often

implemented in fixed point for hardware simplicity or increased operating speed, or because a fixed-point mini or micro-computer is used. The optimization of section *assignment* (ordering and pairing) has thus received considerable attention.

It is possible to prescribe a limit for the input signal level such that overflow is prevented in the worst case (Oppenheim and Weinstein, 1972). However, such a procedure is normally unduly conservative, and gives a very poor utilization of the available number of bits with a consequently high output noise-to-signal ratio. It is usually considered better to allow the possibility of overflow in a small number of cases, but steps are then required to prevent the occurrence of self-sustaining *overflow oscillations*. The use of saturation arithmetic (representing an overflowed result by the largest possible number) is ordinarily sufficient.

Jackson (1970a) devises reasonable dynamic range constraints based on Lp norms of the input signal and the system transfer function. (The Lp norm of a signal x(n) is defined as

$$\| X \|_p = \left(\frac{1}{f_s} \int_0^{f_s} |X(f)|^p df \right)^{1/p} \qquad (6.1)$$

where X(f) is the Fourier transform of x(n), and f_s is the sampling frequency). In the context of a cascade form filter, for a given assignment and a given choice of norm (i.e., value of p) there will be an optimum set of scaling coefficients, one to be associated with each section, to optimize the overall noise performance. Jackson (1970b) gives expressions for the output

noise variance when such scaling is taken into account. Based on these expressions and considerable experimental evidence he then gives "rules of thumb" for deciding on good assignments for a given filter. Regarding choice of norm, Oppenheim and Weinstein (1972) state that constraining the $p = \infty$ norm of the scaled transfer functions to intermediate nodes is appropriate when the input signal is narrow-band, whereas the choice $p = 2$ applies to wide-band signals.

The use of optimization procedures to obtain good assignments began when Hwang (1974) showed that the generation of output noise could be considered as a stagewise process. The method of *dynamic programming* (Gottfried and Weisman, 1973, Ch. 8) is thus applicable to the generation of an optimal assignment. Hwang provides an example to illustrate this technique.

Liu and Peled (1975) point to the rapid increase in computation time which must occur as dynamic programming is applied to filters with larger numbers of sections. For example, the expression for the output noise variance must be evaluated 125 times for four sections, but 1695 times for five sections, the evaluation involving complex contour integrals. Liu and Peled suggest a "heuristic" optimization procedure which, they demonstrate, produces assignments which are "near optimum" and involve only a small fraction of the labour of the dynamic programming approach. Incidentally, they demonstrate the general effectiveness of Jackson's rules, but show that their own procedure usually finds somewhat better solutions.

Liu and Peled begin their optimization procedure by generating a random ordering for the poles and another for the

zeros. They then proceed to explore a set of assignments which may be considered to constitute a "neighbourhood" of this particular assignment. In particular, they leave the zero ordering fixed and interchange all possible *pairs* of pole sections, evaluating the output noise variance for each of these assignments. The process is repeated for a pre-assigned number M of random orderings, and the best assignment found during the whole process is used. The authors find that M = 5 is sufficient for many low-order filters, recommending M = 10 for a filter with 11 sections.

Liu and Peled leave open the question of whether the even simpler course of evaluating the output noise for an equivalent number of purely random assignments would have been more or less effective than their exploratory method. However, they do show that procedures based on random search can be more useful in practice than more formal mathematical methods (in this case, dynamic programming) which can guarantee finding a true optimum, but only with unrealistic amounts of computer time. A similar situation holds with respect to the coefficient quantization problem, to be discussed.

6.3 Filter Structures and Coefficient Sensitivity

6.3.1 Introduction

The necessity to use finite wordlength representations for coefficients causes the performance of the filter to differ from that of a related (but unachievable) "ideal" filter. In this sense the "ideal" filter is that which would result if the lengths of all coefficient words could be allowed to tend to infinity. It does not imply that such a filter achieves exactly

some desired frequency response (for this may be impossible for a filter with a finite *number* of coefficients), but it does imply that the poles and zeros may be positioned with infinite precision in the places suggested by the design procedure. In most cases, filters which are both designed and implemented in floating-point arithmetic on computers with a "large" wordlength, say 60 bits, will have a performance almost indistinguishable from the ideal. In particular, optimization procedures of the type considered in earlier parts of this thesis, when implemented on a general purpose mainframe computer, may be considered to produce the ideal filter coefficients.

Because the positions of the poles and zeros of the filter are functions of the coefficients, the need to use finite-wordlength coefficients implies that the poles and zeros may be sited only at certain positions in the z-plane. For a given filter structure and a given wordlength the possible poles (and zeros) may be plotted as the intersection points of some grid in the z-plane. When the "ideal" poles and zeros are superimposed they are most unlikely to line up exactly with grid points, and accordingly they must be *approximated* by a set of poles and zeros which *are* on the grid.

An obvious strategy is to position a pole (zero) at the *nearest* grid point to each ideal pole (zero). An even simpler method is to round off the ideal *coefficient* values to the nearest realizable figures, without even computing pole and zero positions. Whether these procedures (which may or may not produce the same filter) are adequate, depends on the stringency of the original specification, and on the distances by which the poles and zeros

have to shift in the process. For *any* filter structure and *any* wordlength the possibility exists that some filter *other* than one produced by rounding will have a better response, and in fact this is virtually always true. Methods for finding such filters are the subject of later sections of this chapter. The present section reviews a relevant but largely independent area, the matter of designing filter structures whose pole (zero) grids are suitably *fine* in the vicinity of the ideal poles (zeros). Use of a structure which is good in this regard obviously reduces the bad effects of coefficient quantization by reducing the magnitude of pole-zero displacements. It may even render any discrete optimization method unnecessary, but normally the selection of a good structure and the use of an optimization technique complement each other in the design of a good short-wordlength filter.

Much of the work relating to filter structures and coefficient quantization has concentrated on the complex poles of second-order resonators. The attention to second-order sections is justified by the fact that higher-order IIR filters are usually built by parallel or cascade connections of such sections. In other words, each pair of complex conjugate poles is implemented by a separate section. The reason for this lies itself with the coefficient quantization problem, as it is well known that the sensitivity of the roots of a polynomial to its coefficients usually increases with the degree of the polynomial (e.g. National Physical Laboratory, 1961). The matter was first mentioned in the digital filtering context by Kaiser (1965). *Complex* poles have been emphasized because it is *selective* filters (those with well-defined passbands and stopbands) whose response is most likely to be seriously upset by small coefficient changes, and these filters

normally employ complex poles.

Figure 6.1 (a) shows the ordinary direct implementation of a second-order resonator. In terms of the two coefficients c and d, its transfer function is

$$H(z) = \frac{1}{1 + cz^{-1} + dz^{-2}} \tag{6.2}$$

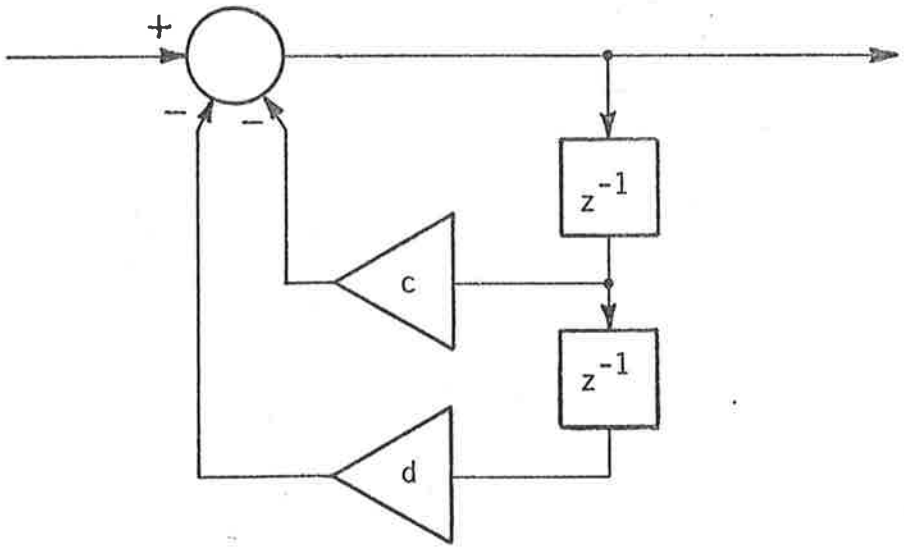
If the values of the coefficients are such that there are complex poles, these poles occur as a complex conjugate pair of radius r and angle $\pm\phi$, where

$$c = -2r \cos \phi \tag{6.3}$$

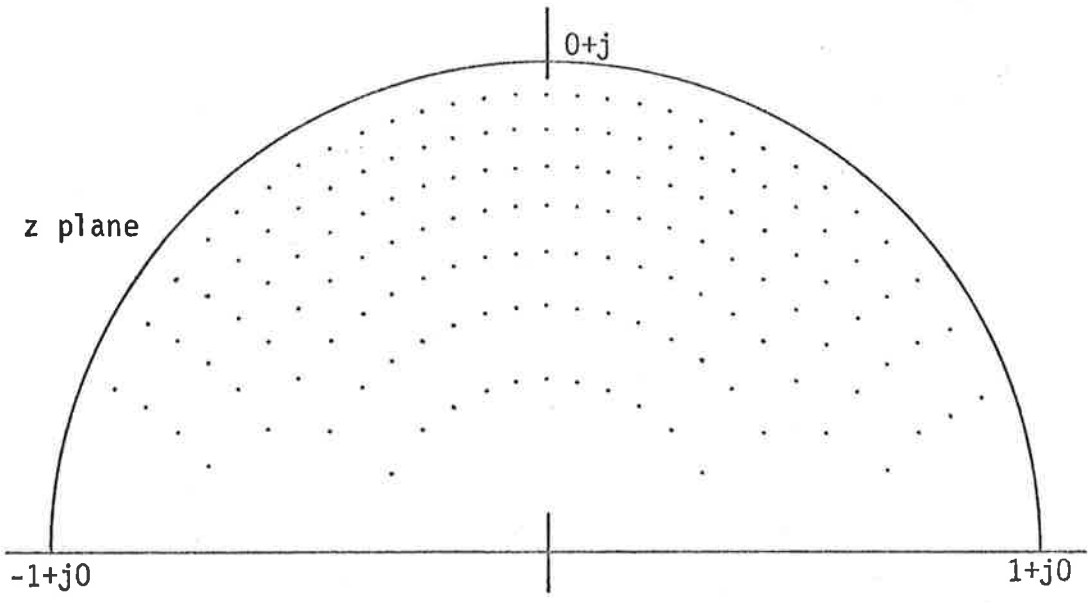
and

$$d = r^2 . \tag{6.4}$$

Thus for a given value of d the possible poles lie on a circle of radius $\sqrt{d}$, and for a given c they lie on a straight line parallel to the imaginary axis with abscissa $-c/2$. If, for example, c and d are to be represented as binary numbers with three fractional bits, all possible poles with positive imaginary parts are shown in figure 6.1 (b), readily recognised as the intersection points of the loci derived above. For smaller quantization steps there will of course be more positions, but their density will still follow the general pattern apparent in figure 6.1 (b). There is a sparsity of possible poles towards the centre of the unit circle, but this is not very serious because the response of a filter is seldom much affected by small changes to poles in this region. What is more important is the lack of allowable poles near the points $z = +1$ and -1, because these are regions of special importance for narrow-band lowpass and highpass filters respectively.



(a)



(b)

FIGURE 6.1 (a) Direct-form Second-Order Resonator

(b) Pole Positions, $q = 2^{-3}$ (upper half plane only)

6.3.2 Special Structures for Second-order Resonators

Several filter structures have been proposed which have pole grids more appropriate for realizing such filters. Avenhaus (1972a) derived two structures which have a *uniform* density of possible pole positions over the entire stable region of the z-plane. One of these (his "circuit C") is shown with its pole grid in figure 6.2. As with the direct-form second-order section, two zeros may be implemented in addition to the poles; their positioning is independent of that of the poles and has the same distribution. The complete structure has been shown in figure 4.8 (page 99). The structure is *canonic* in the sense that only two delay elements (storage locations) are required, but eight multipliers (in addition to a trivial multiplication by 2) are needed as opposed to only four for the direct form.

Rader and Gold (1967) proposed a circuit, shown in figure 6.3(a), which has come to be known as the "coupled resonator." This structure implements a complex conjugate pair of poles $r \exp(\pm j\phi)$. If the output is taken at Y_1, there is also a single real zero at $z = r \cos \phi$ which is clearly not independent of the poles. If the output is taken at Y_2 there is no finite zero, but the filter introduces a unit sample delay and a non-unit gain factor $r \sin \phi$. Since the actual coefficients employed are $r \cos \phi$ and $r \sin \phi$, the possible pole positions lie on a rectangular grid as shown in figure 6.3(b), and have twice the density of the Avenhaus "C" structure for the same wordlength.

The "D" structure of Avenhaus (1972a) provides the same pole distribution as the coupled resonator and also allows the implementation of an independent pair of zeros. The coupled

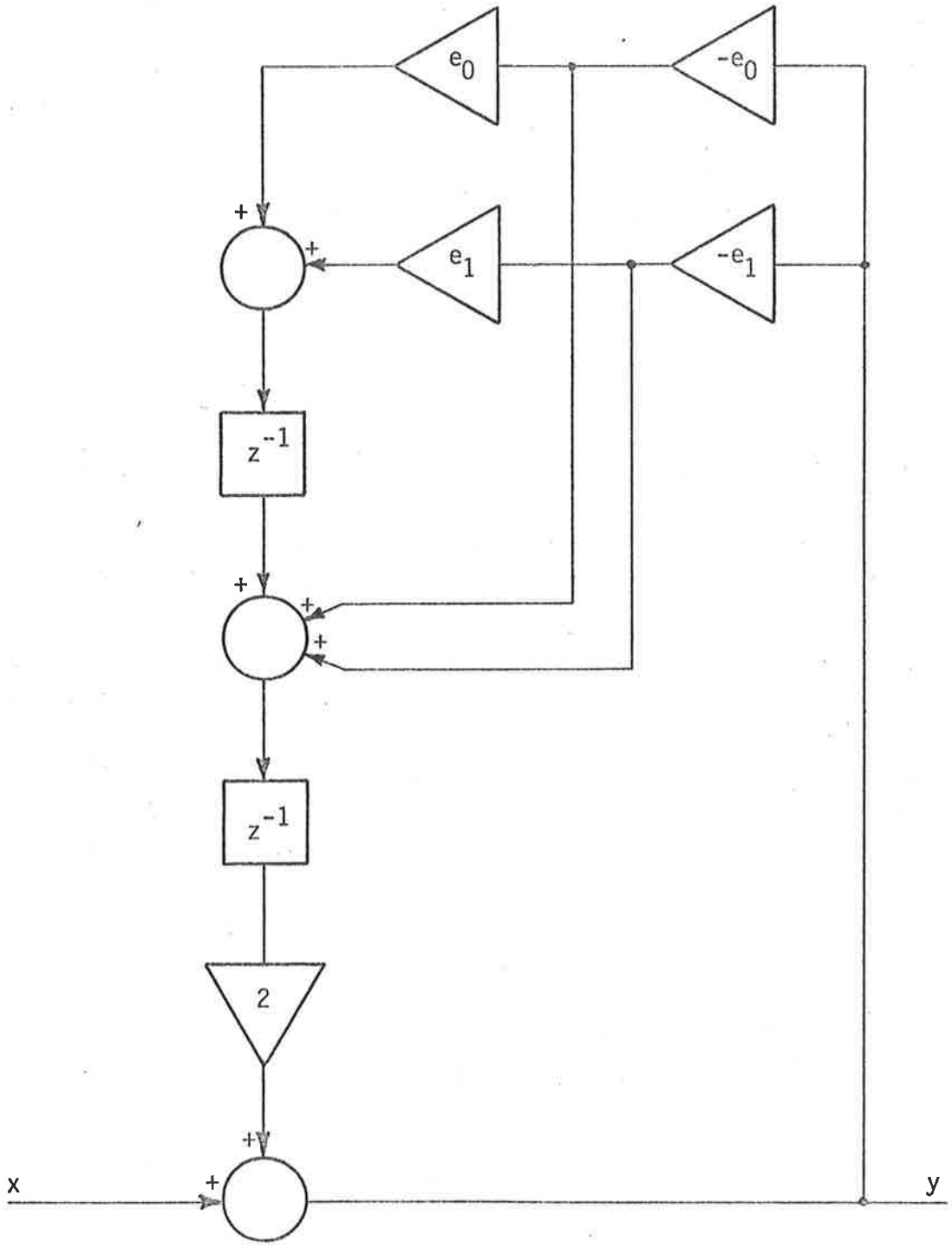


FIGURE 6.2(a) Avenhaus "Circuit C" Second-Order Resonator

$$H(z) = \frac{1}{1 + 2(e_1+e_0)z^{-1} + 2(e_1^{\,2}+e_0^{\,2})z^{-2}}$$

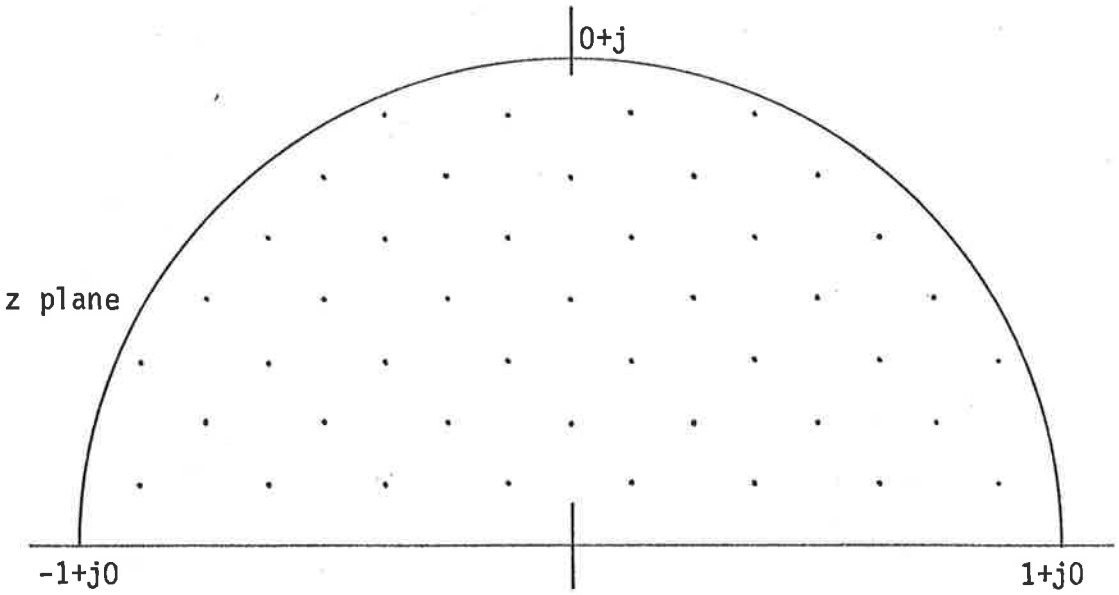
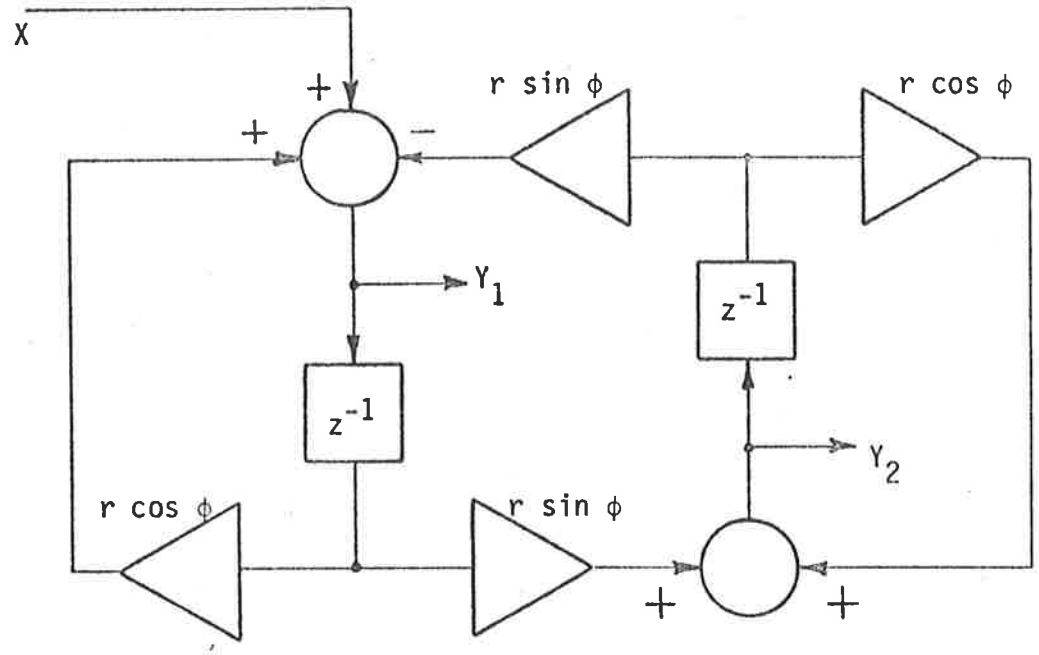
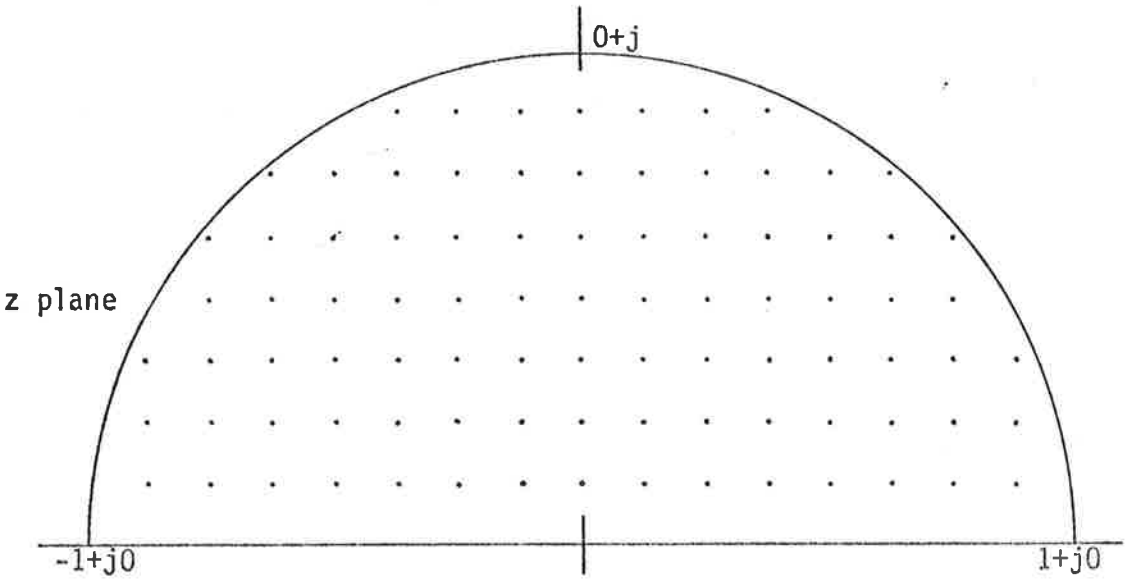


FIGURE 6.2(b) Avenhaus "Circuit C" Second-Order Resonator Pole Positions, $q = 2^{-3}$ (upper half plane only)



(a)



(b)

FIGURE 6.3 (a) "Coupled-form" Second-Order Resonator

(b) Pole Positions, $q = 2^{-3}$ (upper half plane only)

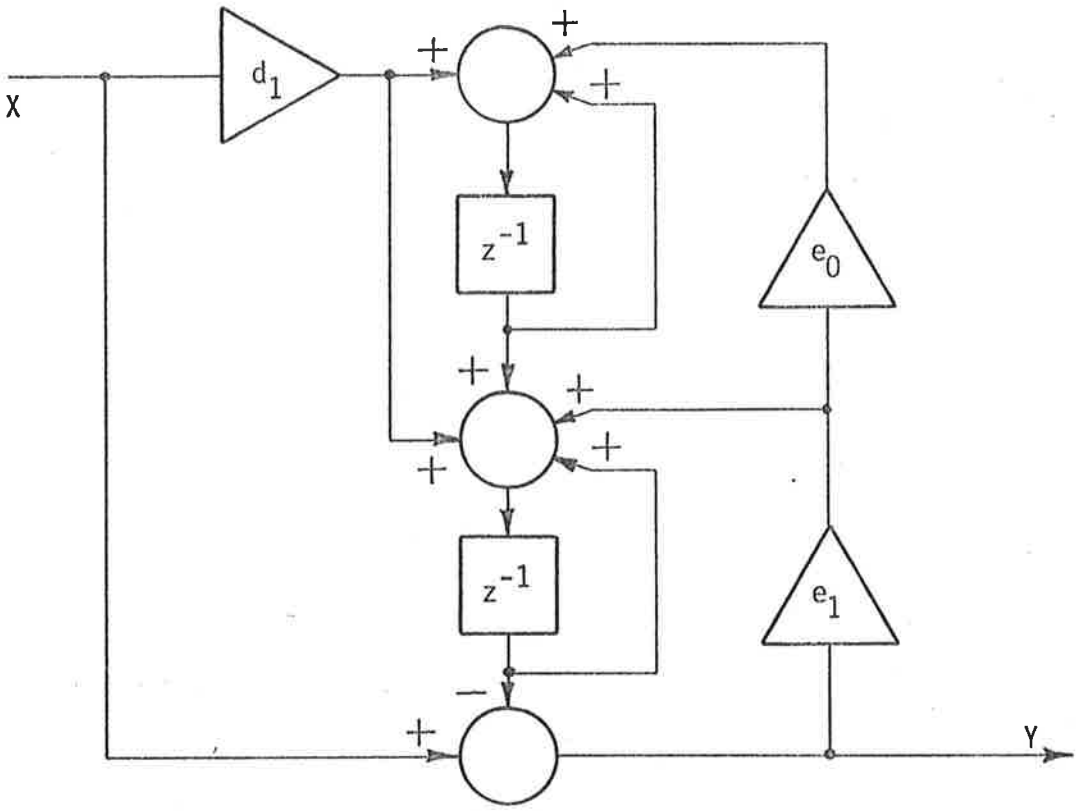
form and both of the Avenhaus structures require four multipliers (for poles only) as opposed to two for the direct form, but the coefficient *memory* requirements are no greater because each coefficient is used twice (with a possible sign change).

Regarding the roundoff noise performance of such circuits, Oppenheim and Weinstein (1972) quote results indicating that the coupled resonator may be superior to the direct form when r is close to 1 and ϕ near 0° or 180°, which is the situation in which it is likely to be used. There appears to have been no analysis of the noise performance of Avenhaus' circuits.

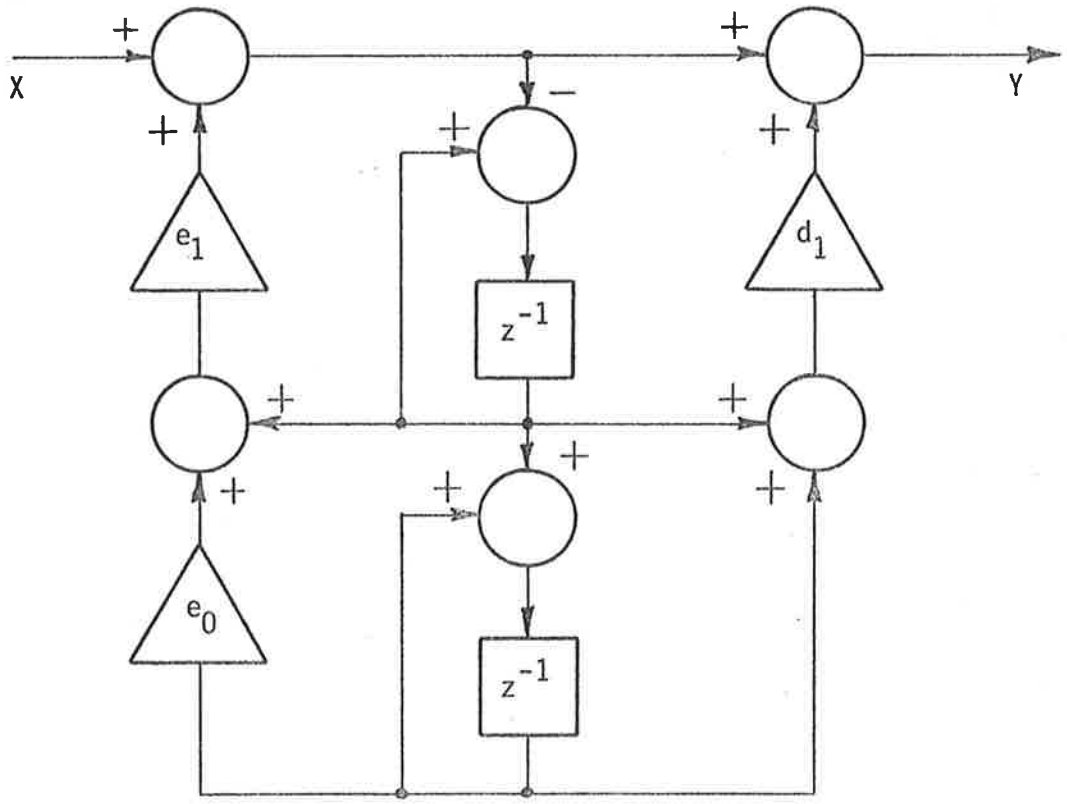
The use of a filter structure which has a uniform grid of allowable pole positions is simple in concept, but it is not necessarily the best way to provide a dense net of pole positions in a given region of the z-plane. Avenhaus (1972a) has derived a structure for which there are many poles near the unit circle for small angles ϕ, which is a useful distribution for narrow-band low-pass filters. Furthermore, this structure is canonic with respect to the multipliers (that is, requires only two multiplications per output sample) as well as with respect to delay elements. The circuit given by Avenhaus (his "circuit E") is shown in figure 6.4(a) and includes one further multiplier to allow the implementation of a pair of zeros on the unit circle, as is usual with selective filters. This filter could also be implemented in its adjoint form, as in figure 6.4(b).

The transfer function of the Avenhaus E structure is

$$H(z) = \frac{1 - (d_1 + 2)\, z^{-1} + z^{-2}}{1 + (e_1 - 2)z^{-1} + (1 - e_1 + e_1 e_0)z^{-2}} \qquad (6.5)$$



(a)



(b)

FIGURE 6.4 Second-Order Section for Narrow-band Lowpass Filters

(a) as given by Avenhaus (1972a) (b) adjoint form

so that the complex pole pairs r exp ($\pm j\phi$) are given by

$$-2r \cos \phi = e_1 - 2 \tag{6.6}$$

and

$$r^2 = 1 - e_1 + e_1 e_0 \tag{6.7}$$

The abscissa of either pole, x, is given by r cos ϕ, so that from (6.6)

$$x = 1 - \tfrac{1}{2} e_1 \tag{6.8}$$

The square of the ordinate is

$$y^2 = r^2 \sin^2 \phi = r^2 - (r \cos \phi)^2$$

$$= 1 - e_1 + e_1 e_0 - (1 - \tfrac{1}{2} e_1)^2$$

or

$$y^2 = e_1 e_0 - \tfrac{1}{4} e_1^2 \tag{6.9}$$

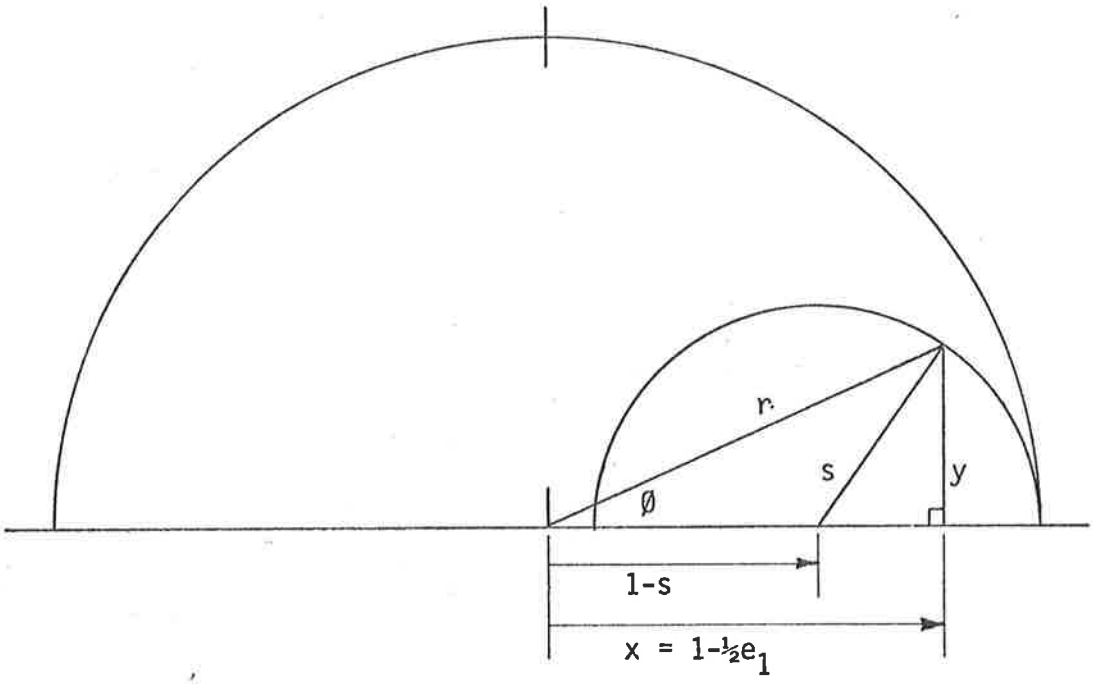
It is convenient (with hind sight) to determine the radius s of a circle centered at (1−s,0) on which the poles lie. The upper half of this circle is shown in figure 6.5(a). By Pythagoras' theorem

$$s^2 = y^2 + [x - (1 - s)]^2 \tag{6.10}$$

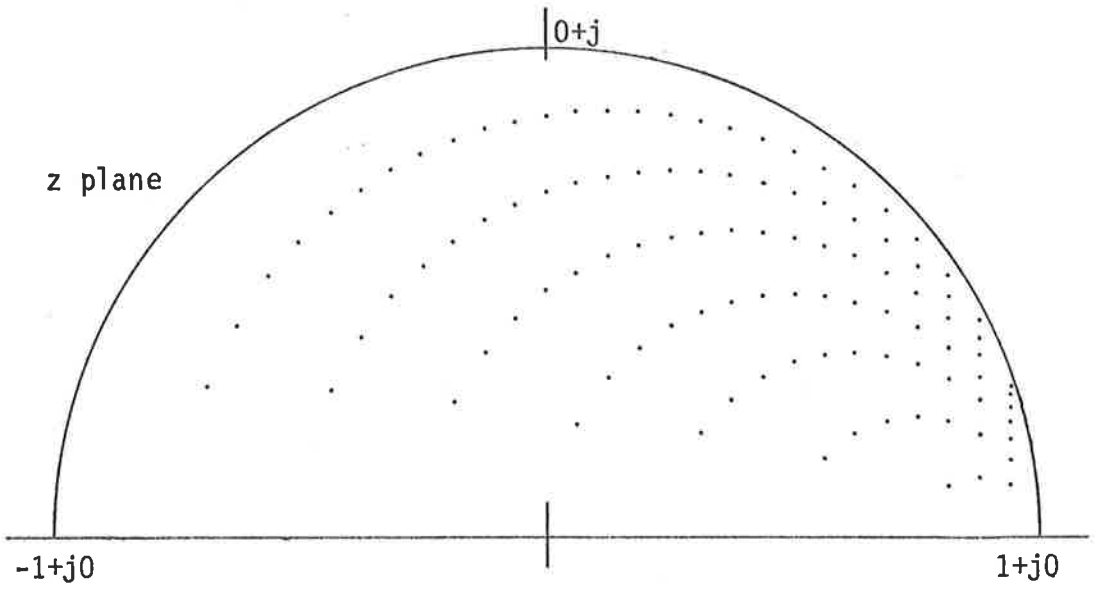
and substituting the known expression for x and y^2 we get

$$s = e_0 \tag{6.11}$$

Thus the allowed pole positions when the coefficients are quantized are determined by the intersections of equally-spaced lines parallel to the imaginary axis (with abscissa $1 - \tfrac{1}{2} e_1$) and circles tangent to the unit circle at z = 1 + j o, with radius e_0. All allowed poles with positive imaginary part are shown in figure 6.5(b) for a quantization step $q = 2^{-3}$. For a complete coverage of the unit circle the range of coefficients is 0 to +4 for e_1 and 0 to +1 for e_0. However, since this particular filter structure is likely to be used to realize lightly-damped poles with small angles, both coefficients can reasonably be restricted to the range 0 to 1.



(a)



(b)

FIGURE 6.5 Poles of Avenhaus 'E' Structure

(a) Definition of symbols

(b) Allowed Poles, $q = 2^{-3}$ (upper half plane only)

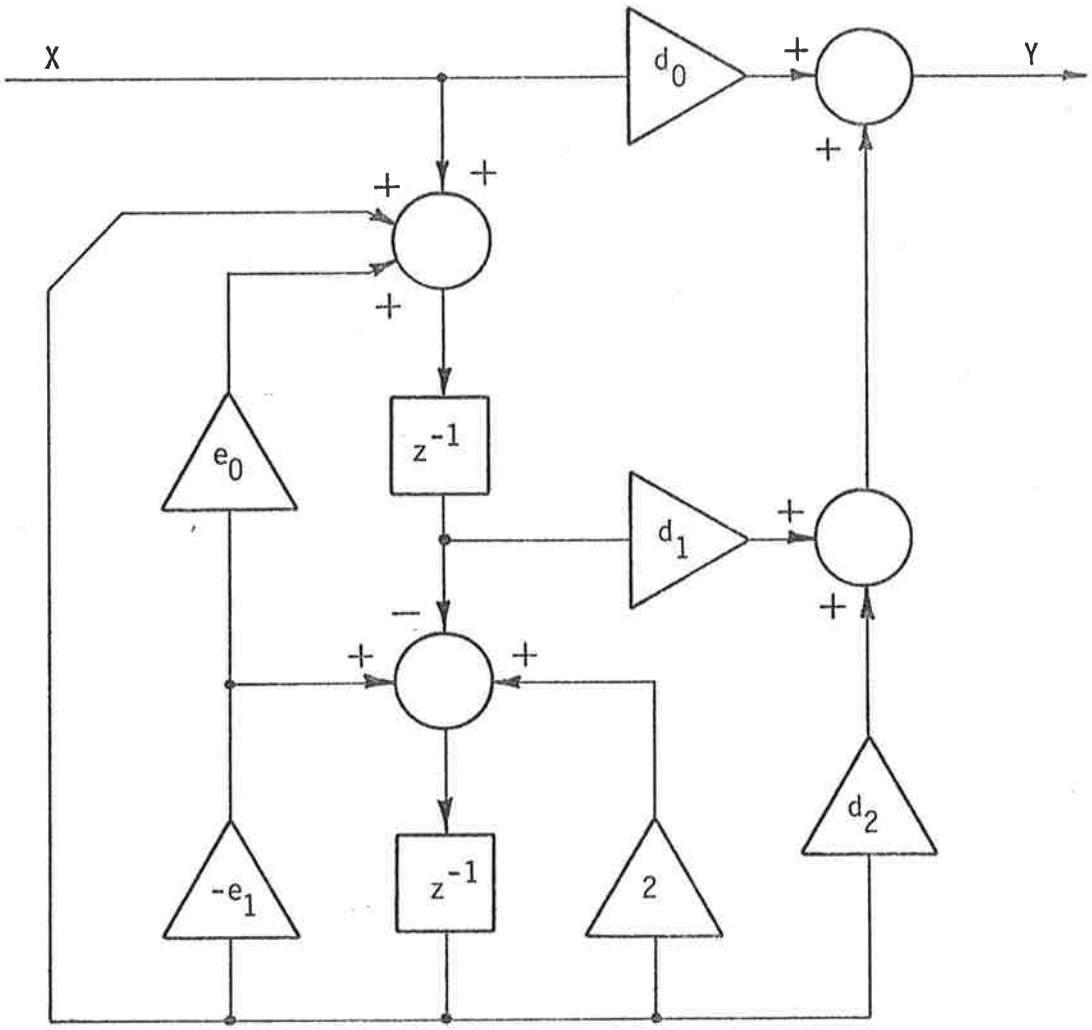


FIGURE 6.6 Lowpass Filter Structure of Rahman and Fahmy (1975)

Rahman and Fahmy (1975) have proposed another structure, shown in figure 6.6, which has the same distribution as Avenhaus' E circuit. With the configuration shown, a general pair of zeros may also be implemented.

For applications other than narrow-band low-pass filters it may be desirable to provide a dense pole grid in other regions of the z-plane. Structures similar to those above are available for narrow-band high-pass filters. Avenhaus (1972a) has also suggested a circuit suitable for narrow-band band-pass filters, in which the positions of poles and zeros are not independent. It provides a high density of possible pole positions near the unit circle and in the vicinity of the zeros.

6.3.3 Structures for FIR Filters

The structure of linear-phase FIR filters has also received some attention from the point of view of coefficient quantization. The "traditional" transversal realization has zeros which are determined by the coefficients of a high-order polynomial, and it is these coefficients themselves which are subject to quantization. Herrmann and Schüssler (1970) suggested instead a cascade realization whose elemental building blocks are of fourth order. Such a filter has sufffficient flexibility to place zeros on or off the unit circle and yet maintain the linear phase property. The zeros are likely to be less sensitive functions of the coefficients than with the transversal form because of the lower degree of polynomials involved.

6.3.4 Wave Digital Filters

A quite distinct approach to digital filtering is embodied in the *wave digital filters*, introduced by Fettweis (1971a,b).

Such filters have considerable advantages, related mainly to their coefficient quantization properties. Wave digital filters are digital signal-flow networks which imitate in structure the classical inductance-capacitance (LC) filters of ladder or lattice type. LC ladder filters possess very good coefficient sensitivity properties, and because of the close element-by-element correspondence the digital realizations were expected to have similar desirable properties under coefficient quantization. That this is true was illustrated with the example of a seventh-order Chebyshev low-pass filter by Crochiere (1972). This filter was implemented (for several different widths of the pass band) as a digital ladder structure, as an ordinary cascade-form recursive filter, and as a cascade-form filter in which the poles were implemented by coupled resonators. In all cases the coefficients were binary floating-point numbers with the mantissas rounded off to various numbers of bits. Crochiere used as a measure of the response degradation under quantization a "relative error" quantity; the increase in pass band magnitude response error relative to the pass band ripple specified for the ideal design. When the pass band edge frequency θ_p was 0.25, all three structures performed almost equally well under quantization. As θ_p was reduced the coupled form filter began to show some superiority over the direct form (as would be expected from our earlier description of the z-plane pole grids). The ladder structure, however, was significantly better than either cascade structure; at $\theta_p = 0.01$ and a design pass band ripple value of 0.2 dB, about 7 fewer bits were needed than with the coupled-form (and 9 fewer than with the direct form) to achieve the same

relative error. It was also observed that the number of bits required in the ladder filter for a given relative error was virtually independent of θ_p. In a wave digital filter the pole positions depend on *all* coefficients so that it is difficult to describe the sensitivity properties by a pole grid pattern as has been done for other structures.

Wave digital *lattice* filters were introduced by Fettweis *et al.* (1974). They require in general fewer multipliers than ladder filters and have even better passband sensitivity properties, but like their analog counterparts have high coefficient sensitivity in the stopband (Wegener, 1978). However, there are many different ways to achieve the (digital) realization of the lattice reactances, and Wegener (1978) has shown that, by taking advantage of this variety *and* by using a discrete optimization scheme, short-wordlength digital lattice filters are often possible.

6.4 Analytic Approaches to the Coefficient Wordlength Problem

6.4.1 General

The development of structures having dense pole distributions in the vicinity of the ideal poles represents a constructive, or *synthetic*, approach to the finite-wordlength coefficient problem. Several other authors have taken instead an *analytic* approach; that is, with a particular filter structure assumed, they have attempted to determine the minimum wordlength necessary to keep the degradation in filter performance within reasonable limits.

Since the z-plane coordinates of any pole (or zero) can

be expressed in terms of some or all of the filter coefficients, it is obviously possible to derive (by partial differentiation) expressions for the small changes to those coordinates which would result from small changes to the coefficients. For example, for the second-order resonator whose transfer function is given by (6.2) and whose (complex) poles are r exp ($\pm j\phi$), we have

$$\Delta r = \frac{1}{2r} \Delta d \tag{6.12}$$

and

$$\Delta \phi = \frac{1}{2r \sin \phi} \Delta c + \frac{1}{2r^2 \tan \phi} \Delta d \tag{6.13}$$

which are expressions accurate to the first order in small quantities. Similar expressions were derived by Rader and Gold (1967) and by Mitra and Sherwood (1974).

6.4.2 Worst-Case Design Approach

Mitra and Sherwood (1974) suggest a worst-case design approach for arriving at a coefficient wordlength which will guarantee that pole (zero) displacements are suitably bounded. Returning to the example of the last paragraph, and *assuming* particular tolerances for both radial and angular pole positions, namely

$$\left.\begin{aligned} |\Delta r| &< R^1 \\ |\Delta \phi| &< \Phi^1 \end{aligned}\right\}, \tag{6.14}$$

equation (6.12) provides an upper bound on the allowable change to the d coefficient, that is

$$|\Delta d| < 2r\, R^1. \tag{6.15}$$

Equation (6.13) can also provide bounds on the changes to both coefficients if it is assumed that the allowable angle change Φ^1 is apportioned equally between Δc and Δd, namely

$$\left.\begin{aligned} |\Delta c| &< r\sin\phi \; \Phi^1 \\ |\Delta d| &< r^2 \tan\phi \quad \Phi^1 \end{aligned}\right\} \qquad (6.16)$$

A sufficient wordlength for each coefficient may then be quoted by requiring that one half of its quantization step size satisfy the most stringent inequality (6.15 or 6.16) for that coefficient.

There are several objections to the approach outlined above. Firstly, there is no guide as to the magnitude of pole displacement (in either the r or the ϕ direction) which should be considered "acceptable", because there is nothing relating this to the filter response in either the time or the frequency domain. Secondly, the wordlength estimates will normally be unduly pessimistic. This is both because the contributions to pole displacement due to the individual coefficients need not add in the same direction, and because quantization will often perturb a coefficient by much less than one half of the step size. Because each pole pair is treated individually, there is of course no recognition that the bad effects of displacing one pair could be partially compensated for by moving another. The discrete optimization procedures of section 6.5 allow the designer to benefit from such effects, and usually result in the saving of several bits.

6.4.3 Statistical Approaches

Knowles and Olcayto (1968), while not expressly taking any advantage of mutually-compensating pole displacements (which would require an optimization method), presented a more soundly-based analysis of coefficient quantization effects. The

perturbations to coefficients due to quantization are taken to be *statistical* in nature, and their effects are related to a simple measure of frequency domain performance.

At the most basic level, the assumption that the rounding of coefficients produces errors which may be characterized statistically is obviously untrue, because for any given filter and wordlength the rounded coefficients are quite determinate. However, the method has in practice led to realistic wordlength estimates, probably because it provides a mechanism for *averaging* the bad effects of the coefficient errors. Such averaging would normally provide a closer description of a real situation than would the pessimistic assumption that errors add in the worst possible way.

Denoting the transfer functions of the ideal filter and of a finite-wordlength realization as $H_\infty(z)$ and $H(z)$ respectively, Knowles and Olcayto define a *mean-square convergence criterion*

$$\sigma_w^2 = \frac{T}{2\pi} \int_0^{\frac{2\pi}{T}} |H(e^{jwT}) - H_\infty(e^{jwT})|^2 \, dw. \qquad (6.17)$$

This provides a measure of the departure of the performance of the finite wordlength filter from that of the ideal in terms of the overall effect on the frequency response. Assuming that the quantization error in each filter coefficient is a random variable with a uniform distribution between $-\frac{1}{2}q$ and $+\frac{1}{2}q$, and that the errors in different coefficients are statistically independent, the authors are able to derive expressions for the *expected value* $\langle \sigma_w^2 \rangle$ for filters in direct and parallel form. (Here q is the quantization step size, assumed the same for each coefficient). q appears only as a simple scaling factor, so that once $\langle \sigma_w^2 \rangle$ has been found for some wordlength it may very simply be found

for any other.

To confirm their analysis, Knowles and Olcayto computed the actual value of σ_w^2 for a 22nd-order bandstop filter for several coefficient wordlengths, using both direct and parallel programming. Their predicted values correspond closely with those observed for wordlengths greater than 39 bits (direct form) and 10 bits (parallel form). With 8 and fewer bits in the parallel-form the values of σ_w^2 were somewhat less than the expected values, apparently due to the breakdown of the assumption that the effects of coefficient rounding combine statistically. This is not surprising for such severe quantizations.

The method of Knowles and Olcayto may be used as a design tool (to predict the necessary coefficient wordlength) if an estimate is available of the maximum allowable value of σ_w^2. The authors suggest that q be selected such that $3\sqrt{<\sigma_w^2>}$ is less than the maximum tolerable magnitude response error at any frequency. Unfortunately the method is not applicable to filters in cascade form.

Avenhaus (1972b) presented another statistical method for estimating a sufficient wordlength. He criticises the overall frequency-domain measure σ_w^2 used by Knowles and Olcayto because, for selective filters, it fails to express the stringent attenuation requirements in the stop band (s). He prefers to treat the performance at each frequency separately. Considering a statistical ensemble of filters (the ensemble whose coefficients are independent and uniformly distributed within $\pm \frac{1}{2}q$ of the ideal values), Avenhaus derives a "statistical wordlength function",

which is that wordlength which allows 95% of the members of the ensemble to meet the filter specification at a particular frequency. The maximum (over all frequencies) of the wordlength function is then taken as an estimate of the minimum wordlength necessary for the filter.

Avenhaus observed that the maximum value of the statistical wordlength function within the stopband was likely to differ considerably from that in the passband. In other words, the wordlength required for the filter was likely to be determined by the coefficient sensitivity in *either* the passband *or* the stop band, with the filter response in the "other" band being very little different from the "ideal". Crochiere (1975) pointed out that this effect was likely to result in wastage of bits. Typical specifications for a selective filter quote a maximum pass-band ripple δ_1, and a minimum stop band attenuation (which may equivalently be expressed as a maximum stop-band "ripple" δ_2). The order of the filter must be high enough to meet these requirements, and since it is normally taken as the next higher integer it is usual for the "ideal" filter to meet both specifications with some margin. In fact, to allow for coefficient quantization, there *must* be such a margin. For want of a better criterion, traditional ideal designs have often been such that passband and stopband specifications are met with *equal* margin. Crochiere suggested instead that the ideal parameter values be selected to equalize the maxima of the statistical wordlength function in the passband and in the stopband. He proposed two algorithms for doing this and illustrated their use with the example of an elliptic low-pass filter. Both are in

fact iterative optimization methods in a space of low dimensionality; the first uses two parameters, being the achieved pass- and stopband ripple maxima; the second uses four parameters, allowing variation also in band edge frequencies.

The equalization of statistical wordlength function is not guaranteed to result in a saving of bits but it is likely to do so. Crochiere has observed improvements of 1 to 3 bits in the actual wordlength required, and this is still with simple coefficient rounding. Crochiere further conjectured that his statistically-optimized ideal filters may provide superior starting points for discrete optimization by search techniques. This has been confirmed by Kwan (1979).

6.5 Discrete Optimization of Digital Filter Coefficients

6.5.1 General

Simpler statistical methods for predicting the wordlength requirement are useful as an intermediate stage in the design of a filter; the stage following the determination (by optimization or otherwise) of the ideal coefficients. Crochiere's procedure may be thought of instead as a way to incorporate a penalty for short wordlength into the objective function to be minimized by the ideal design. However, the estimates of the actual wordlength requirement are usually still pessimistic, although less so than those derived by worst-case analyses. Rounding of all coefficients very seldom produces the "best" filter of a given wordlength, and discrete parameter optimization can often result in the saving of several bits. Avenhaus (1972b) gave a specific example, an

eighth-order bandpass filter, realized in cascade form. A worst-case design procedure suggests that 14 bits would be necessary for the coefficients, while his statistical method reduces this to 12.5 bits. In the event, the specifications could just be met by coefficients of 11 bits when simple rounding was employed. After application of an optimization process it transpired that 8 bits were sufficient.

Many different procedures have been proposed for carrying out the discrete coefficient optimization, and those known to this author are reviewed in this section. In section 6.6 a new method is proposed which is thought to have some advantages. At the outset it should be remarked that the term "optimization" is something of a misnomer in this context, because most of the methods cannot specifically seek the global discrete optimum, even in some restricted region where the continuous function is known to be unimodal. Their aim is to improve the coefficient set as found by rounding the ideal values by as great a margin as possible, within reasonable constraints on computer time. If the discrete optimum *is* found there is no indication of its optimality.

The question arises as to what function, of what variables, is to be minimized (or reduced). The simplest case, conceptually, is the case where the ideal coefficients have themselves been found as a result of the continuous minimization of some objective function, and it is known in advance that all coefficients are to be represented with some particular wordlength. In such a case, the original objective function is still taken as a true measure of the merit of the filter. The finite-wordlength requirement has however, modified the feasible region so that it now consists

only of the intersection points of a grid in the N-dimensional parameter space. A feasible point is sought which minimizes (or at least has a low value of) the objective function. The objective function value of the ideal filter of course provides a lower bound on the value achievable by the finite-wordlength filter, and the value associated with rounded coefficients is an upper bound. When the problem is cast in this form it becomes purely one of mathematical optimization, any obvious connection with digital filtering being lost. Most of the methods proposed *do* fall into this category, and so they could have application to other, unrelated, problems.

The same "general" methods may be applied in an attempt to *minimize* the coefficient wordlength by performing a series of discrete optimizations with differing numbers assumed for the wordlength. The optimal objective function value will be a monotone decreasing function of wordlength, although not necessarily *strictly* monotone. This is because all feasible solutions for a given wordlength are also feasible solutions for all greater wordlengths. The wordlength minimization process requires a judgement to be made that one "optimal" filter is acceptable but that the one of the next lower wordlength is not. With many objective functions there is no natural value which separates acceptability from unacceptability, and so the final filter is seldom "optimal" in any rigorous sense, quite apart from any shortcomings in the discrete minimization procedure itself. Indeed, the objective function itself may not reflect the true aims of the designer of the filter. Nevertheless, minimization procedures, and, further, discrete minimization

procedures, are useful because they do produce effective digital filters of suitably small wordlength.

6.5.2 Integer Programming Approaches

Discrete optimization procedures of the "general" type referred to above are those of Avenhaus (1971, 1972b), Steiglitz (1971), Suk and Mitra (1972), Charalambous and Best (1974), Kwan (1979), Kodek (1980) and the present author (Smith, 1977, 1979, and section 6.6 of this thesis). Because the connection with digital filters is superfluous to the discussion of these methods, the "coefficients" will be referred to by the term more widely used in optimization literature, "parameters". Furthermore, it will be assumed that all parameters have been scaled up by a factor equal to the quantization step size so that the feasible solutions are those for which all parameters assume *integer* values.

Charalambous and Best (1974) applied the branch-and-bound integer programming approach, which was outlined in section 2.3.5 of this thesis. This is a true "optimization" procedure in the sense that finding the global minimum is guaranteed if the objective function is convex. In the case of a differentiable objective function (as arises from a least squares or least pth criterion, but not from a minimax criterion), the function will be well-behaved in a neighborhood of the continuous local minimum. In fact, it will closely resemble a quadratic function in N variables, which accounts for the success of most of the gradient methods for continuous optimization dealt with in earlier parts of this thesis. In the case of the digital filter problem the quantization stepsize is often small enough that the

region in which the function is well-behaved (closely quadratic, and therefore convex) extends outwards to many times the feasible mesh size. This implies that the branch-and-bound approach is almost certain to find the true discrete optimal parameter vector. However, the method is extremely time-consuming because an inequality-constrained N-dimensional continuous optimization is required at each branch point in the tree search. The constraints are linear, and Charalambous and Best take advantage of an efficient feasible-direction method to solve the sub-problems. Nevertheless, they report that (working with an IBM 360/75) CP times of 43, 42, 35 and 31 seconds were required for the entire tree search, in problems with only five parameters. The approach would certainly be unrealistic in terms of computer time if the number of parameters were significantly greater than this. Kodek (1980) applied branch-and-bound to FIR filter design with up to 40 parameters and reported using "huge" amounts of computer time, although he did not quote figures.

The methods of Avenhaus (1971, 1972b), Steiglitz (1971), Suk and Mitra (1972) and Kwan (1979) are similar in that some form of search is applied in the discrete parameter space. No continuous re-optimizations are applied, resulting in a great saving of computer time when compared with branch-and-bound. All function evaluations are made with feasible (integer) values of the parameters. Although this implies that almost all of the computer time is spent in testing parameter vectors which are candidates for the optimum, it means that only rudimentary advantage may be taken of the well-behaved nature of the function. In the opinion of the present author, this is a great sacrifice, and this belief provided the motivation to develop the method of

section 6.6. However, discrete search methods do not rely on differentiability, and so are applicable also to maximum modulus objective functions.

Three basic types of search have been proposed. These are a) the enumerative search, b) the univariate search, and c) the random search. The first is conceptually very simple; trials are made of all possible (feasible) parameter vectors within a restricted region of parameter space. Either all parameters may be varied, or some may be held fixed (that is, the enumerative search region may be restricted to a sub space). In theory, the global optimum may always be found by such a search, but to tackle the complete problem by this method alone is prohibitive in terms of computer time. For instance, in the first example of Charalambous and Best (1974) (the differentiator, with N = 5 and q = 6) two of the optimal coefficients were removed from the rounded values by four quantization steps. To perform an exhaustive search out to that distance in all directions from the "rounded" parameter vector would require 9^5, or 59049, function evaluations, and there is still no guarantee of optimality. Hence, when an enumerative search is used it is usually concentrated in a sensitive subspace and is only begun when another technique has reached an impasse.

The second type of search is the "univariate search", a term introduced by Avenhaus (1971). The search begins at a feasible parameter vector, for example, that produced by rounding the coefficients. One parameter is then perturbed by some integer (often 1). If the function value decreases, the step is accepted and the search is continued from the new point.

If the move fails, a further trial is made by perturbing the same parameter in the opposite direction. This process is repeated for all other parameters in turn, with any improvements being accepted. The search terminates when eventually it cycles through all parameters without any improvement being recorded, except that if the original steplength were greater than unity, some further improvement could be sought with a reduced steplength. The pure univariate search reaches an impasse, when no unit step in any direction produces a decrease in the function value. Such *discrete local minima* can be very numerous and often quite poor in terms of function value. Point A in figure 6.7 is an example of such a discrete minimum in two dimensions. A univariate search reaching this point would terminate and so the true discrete optimum (point B) would not be found.

Because of the inadequacy of the univariate search by itself, several embellishments have been suggested, or may be worth trying:

(a) The process may be repeated from several starting points, thus producing a variety of discrete local minima from which the best may be selected. The starting points could be generated at random, or some of the unfavourable points arising in the search could be used (Avenhaus, 1972b).

(b) Because improvements are accepted whenever they are encountered, the progress of the search is sensitive to the order in which the parameters are enumerated. A variety of candidate solutions could be produced with different random shufflings of the integers 1, 2 . . . N. (Steiglitz, 1971).

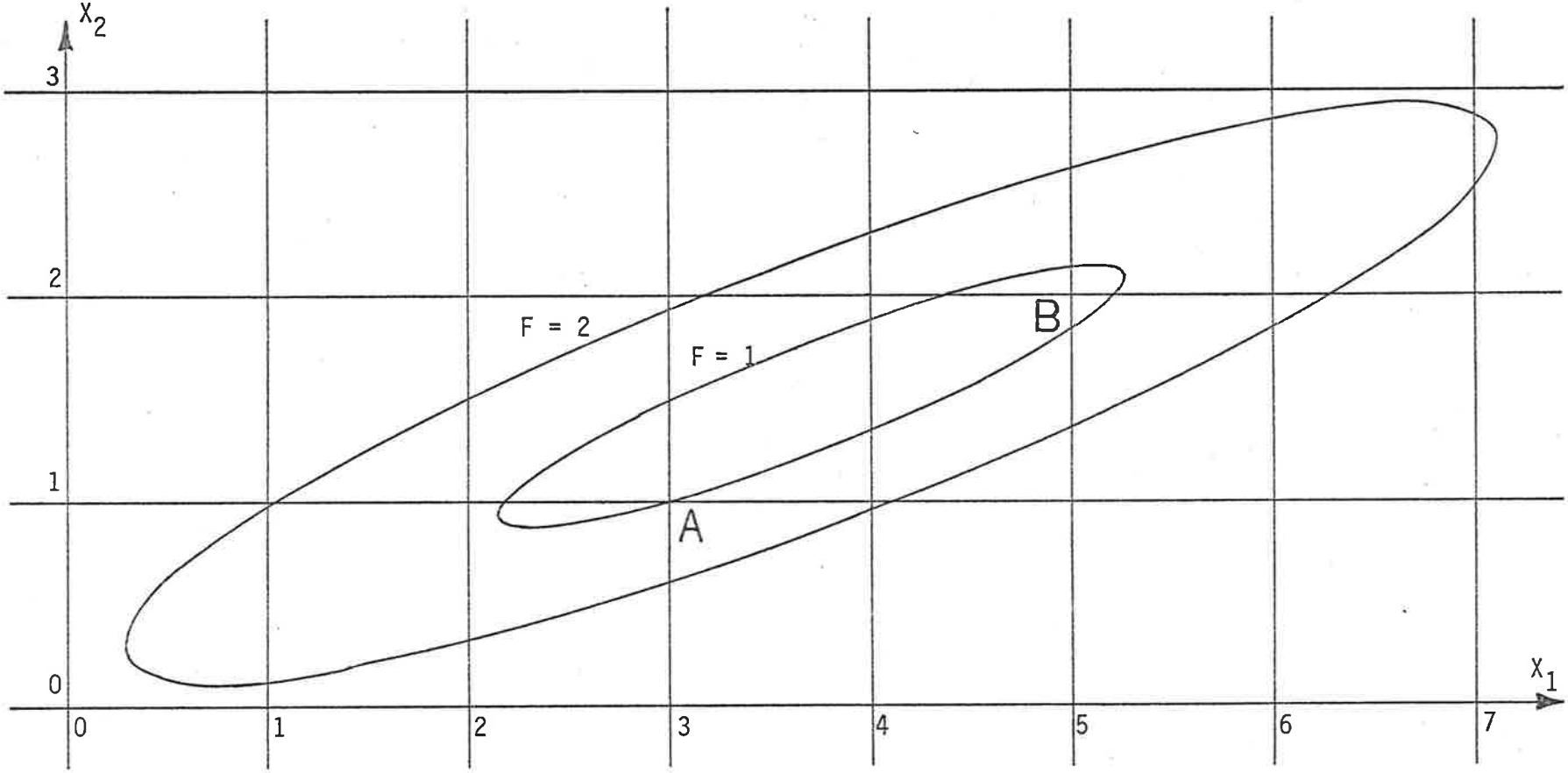


FIGURE 6.7 Two-dimensional Example of Discrete Local Minima

A Sub-optimal B Globally Optimal

(c) When a local minimum is found from which the univariate search cannot escape, the power of the local search may be increased. For instance, the parameters may be perturbed in pairs instead of singly (Steiglitz, 1971), or an enumerature search may be performed over a hypercube centred on the point in question. Avenhaus (1972b) used such a search, restricted to the subspace of the four most sensitive parameters, the length of an edge of the cube being 2. Note that neither of these strategies (which, in two dimensions, are equivalent) would yield an improvement from point A in figure 6.7.

The algorithm of Steiglitz (1971) in fact uses the univariate search as a component in a larger strategy, which is a generalization of the pattern search method of Hooke and Jeeves (1961) to a discrete space. The univariate search is used in an exploratory phase about a *base point*. It is not necessarily carried to completion, but only until all parameters have been perturbed once. Steiglitz has used an ordinary univariate search, but prefers a modified search ("TWO-OPT"). Should any parameter fail to improve the function when perturbed alone, simultaneous perturbations are made to that parameter and each succeeding parameter, until either an improvement results or all parameters have been exhausted. The parameters are shuffled into a new random order each time the exploratory search is used. When this exploration is finished an attempt is made to gain further improvement by extrapolating from the base point to an equal distance on the other side of the "forward" point (the final point of the exploratory phase). The rationale for this "pattern move" is as for the original Hooke-Jeeves method, i.e. it is

assumed that the exploration has discovered a good direction for progress. In the Steiglitz algorithm the new point is still feasible, and is taken as the starting point for another exploration. Should this phase yield a lower function value than the forward point, the pattern move is deemed successful and an attempt is made to extrapolate even further. This provides for rapid acceleration whenever a very good search direction has been identified. When a second or subsequent pattern move fails, retreat is made to the last successful exploratory pattern. If the first attempt at a pattern move fails, the entire process is repeated from the best point using a smaller steplength.

In contrast to the original Hooke-Jeeves algorithm for a continuous space, the steplength cannot be reduced indefinitely, and the process eventually terminates at a discrete local minimum which may still be quite unfavourable in terms of function value. Steiglitz emphasized the necessity for re-starting the process several times, using different seeds for the random number generator which determines the parameter ordering. In this way a variety of candidate solutions are produced, from which the best may be selected. He also found that large initial steplengths tended to produce poor local optima, and that the most effective initial value was 2, or even 1, which is the smallest possible value.

The acceleration capability of the Steiglitz algorithm would appear to give it the ability to converge from points remote from the final optimum. However, in such a mode it is most unlikely to be better for locating the approximate region of the

solution than one of the efficient gradient-based (continuous) methods. In fact, Steiglitz does not use it in this way. In each of his quoted examples the process was begun by rounding the coefficients at the continuous optimum. In practice, the discrete optimum will usually lie within a few unit steps of this point, and so it is difficult to see that successive pattern moves could succeed very often. The acceleration feature is therefore probably largely superfluous. Such success as the Steiglitz method has achieved, is probably attributable to the occasional determination of a good search direction through the use of TWO-OPT.

Wegener (1978) has also used a discrete search method based on the Hooke-Jeeves pattern search, in conjunction with wave digital lattice filters.

The algorithm of Kwan (1979) represents another attempt to combine the univariate search with an enumerative search in a subspace. As with the Steiglitz method, the parameters are shuffled into a random order for the univariate search, so that different seeds for the random number generator will lead to a variety of solutions. The univariate search incorporates also a "successful move scheme"; if any (single parameter) step reduces the function value, a further step in the same direction is tried before proceeding to the next parameter. The objective functions used by Kwan are derived from rigid specifications on maximum passband ripple and minimum stopband attenuation, and have the property that filters having this value less than 1 are acceptable, while those having a greater value are not. Hence, the univariate search is stopped as soon as an acceptable

parameter vector is found, and the whole procedure repeated to try to find an acceptable filter with the next shorter wordlength. If, however, no acceptable filter is located by the univariate search, an enumerative search is tried in a region within the subspace of the three most sensitive parameters. (The trials associated with the preceding univariate search enable the parameters to be roughly placed in order of sensitivity). If this also fails, the three parameters are given the best values found and a search performed in the subspace of the next three parameters. This sequence is continued if necessary until all parameters have been varied. Kwan recognizes that the search could still be trapped at a poor local minimum, and so he allows successively shorter wordlengths to be investigated until failure to produce an acceptable filter occurs twice running.

Kwan has also demonstrated the usefulness of Crochiere's (1975) "statistical wordlength equalization". In each of three examples (elliptic lowpass filters), Kwan's algorithm found acceptable filters of 9-bit wordlength when equalization was employed in arriving at the ideal coefficients. Without equalization, no solutions were found for fewer than 10 bits. The validity of Crochiere's idea is thus established when used in conjunction with an optimization procedure, as well as with simple rounding. However, the 9-bit solutions still *exist* regardless of the starting point for the search, and so it may be considered a shortcoming of Kwan's algorithm that it did not find them.

The third basic type of search in discrete space is the random search. This is not to be confused with methods such as

Steiglitz's which merely randomize parameter ordering to allow a variety of solutions to be found by otherwise deterministic searches. In the simplest application trial points could be generated at random within a given region as an alternative to a straight enumerative search. There would then be no guarantee of finding the optimum but there would usually be a good probability of finding a "reasonable" solution with far fewer evaluations than needed for the enumerative search. There is scope for adaptation in such a random search; for example, the probability density function could be made non-uniform with a peak centred on the best point found so far.

Suk and Mitra (1972) describe one approach to a random search, which operates in two phases. In the first phase, trial steps are made from the current best point by perturbing each parameter by 0, +h or -h, where the three possibilities are randomly selected and equally probable. Any improvements found are accepted (the centre point for the search is re-defined). The stepsize h is initially set to some value H_o, and if no improvements are recorded after a certain number of evaluations it is reduced by 1. This process continues until h is reduced to some preset value H_1, when the second phase begins. This is a random search with all points *within* the hypercube of edge length $2H_1$, being equally probable. Again, any improvements re-define the centre point of the hypercube.

There appear to be several weaknesses in the method as originally described by Suk and Mitra. In phase one, they require that *every possible* step be tested with a given value

of h before the stepsize is reduced. As the number of possibilities is 3^N, it is an unrealistic requirement in a space of even moderate dimensionality due not only to time but to the size of the look-up table necessary to record which steps have been tested. A similar requirement that the second-phase hypercube searches be exhaustive (and begin afresh whenever an improvement is found), is even less realistic, because the number of points to be tested is $(2H_1+1)^N$, an even larger number if $H_1 > 1$. Additionally, it is not obvious that the search would *ever* terminate with any given algorithmic "random" number generator.

In fact, Suk and Mitra performed the searches only in a subspace with N = 2. The procedure is then manageable but not very powerful. Such success as they achieved seems due to the large number of parameter vectors which *are* substantially better than the "rounded" vector, in the examples that they chose. Improved solutions are therefore quite easy to find, even with a bad method. In no case reported was the true discrete optimum found.

Random search methods in general have the advantage of being insensitive to major irregularities in the contours of the objective function. They may thus be the most suitable methods for integer optimization problems in which such irregularites occur over distances smaller than the feasible grid size. However, when this is not the case (as seems true for most digital filter problems), methods which include some means of discovering good *directions* in which to move in parameter space, would usually be more efficient. Of such methods there are those which perform evaluations only at

feasible points, and that of Steiglitz (with TWO-OPT) may be close to optimal. However, if the objective function is differentiable the method to be discussed in section 6.6 is applicable, and should be even better.

6.5.3 Successive Discretization Approaches

All of the methods for discrete coefficient optimization considered up to this point have been of the general "integer programming" variety; the design of the algorithm is based on the general behaviour of functions, and the connection with digital filtering is largely incidental. In contrast, several other methods have been proposed in which this connection is kept to the fore.

The methods of Boite, Dubois and Leich (1974) and Brglez (1978) are similar in that parameters are rounded off one or two at a time, and a continuous optimization of the remaining parameters is then performed in the reduced-dimensional space resulting. Boite *et al.* consider cascade-form selective digital filters in which all zeros are on the unit circle, and treat only the magnitude response. The transfer function is then

$$H(z) = A_0 \prod_{k=1}^{K} \frac{1 + a_k z^{-1} + z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (6.18)$$

The authors observe that the sensitivity of the magnitude response to parameter changes is much smaller in the stopband than in the passband. (A similar effect was found by Avenhaus (1972b) in his derivation of statistical wordlength functions). Their first step, then, is to round off all numerator coefficients

a_k (which determine the stopband zeros) to the nearest allowable values and re-optimize the c_k and d_k coefficients in the continuous 2K-dimensional space. The discrete optimization procedure proper then begins. One denominator coefficient (c_k or d_k, for some k) is rounded off and the remainder re-optimized in continuous space. Boite *et al.* claim that the stopband magnitude characteristic is little changed during this process, so that the objective function may possibly be evaluated on a passband frequency set only. The cycle of rounding and continuous re-optimization is repeated until all coefficients have been "discretized". Coefficients are rounded off in decreasing order of sensitivity.

The procedure of Brglez (1978) is more involved, but more general in that it is applicable to filters with arbitrary pole-zero distributions (not only to piecewise-constant magnitude characteristics). The filters considered are of the cascade form, with transfer function

$$H(z) = A_0 \prod_{k=1}^{K} \frac{1 + a_k z^{-1} + b_k z^{-2}}{1 + c_k z^{-1} + d_k z^{-2}} \qquad (6.19)$$

Both of the coefficients which determine a pole or zero pair are discretized simultaneously (that is, either a_k and b_k, or c_k and d_k, for some k). However, they are not merely rounded off to the nearest allowable values for a given wordlength. Assuming that a_k and b_k are selected for discretization, several (up to 16) discrete pairs in the (a_k, b_k) plane are selected for trial. These pairs are the feasible points which form some neighborhood of the current continuous optimal pair (a_k, b_k), such as the four nearest

points plus the twelve additional points surrounding those four (as shown in figure 6.8). The coefficients ^{a}k and ^{b}k are fixed at some or all of these points in turn and a continuous re-optimization of all remaining coefficients is performed for each trial. The (a_k, b_k) pair which gives rise to the smallest objective function value is accepted and a new pair is then selected for discretization. A variety of rules may be applied to determine which of the neighborhood points to try, since trying them all would require excessive computer time. Firstly ("rule 1") the objective function may be evaluated for each point (without continuous re-optimization of the remaining coefficients)and any desired number (say 3) selected on the basis of lowest value. This rule involves substantial computation but Brglez regards it as essential for arriving at the best possible solution. A simpler alternative is to evaluate the coordinates of the relevant pole or zero for each discrete coefficient pair, and select those for which the pole (zero) perturbation from the ideal is least. Brglez (his "rule 2") has used not the actual pole perturbation but the *change to the distance* between the pole and some "critical" point on the unit circle, such as a band edge, for a selective filter. A third, and even simpler, alternative is to use just the nearest coefficient grid point (i.e., rounding), or the nearest two (or more).

Brglez considered two criteria for deciding the order of coefficient pairs to be discretized. Again, he regarded a fairly complex rule as being essential to obtain the best results, with arbitrary pole-zero distributions. The objective function value is calculated (as for "rule 1" above) for all points in

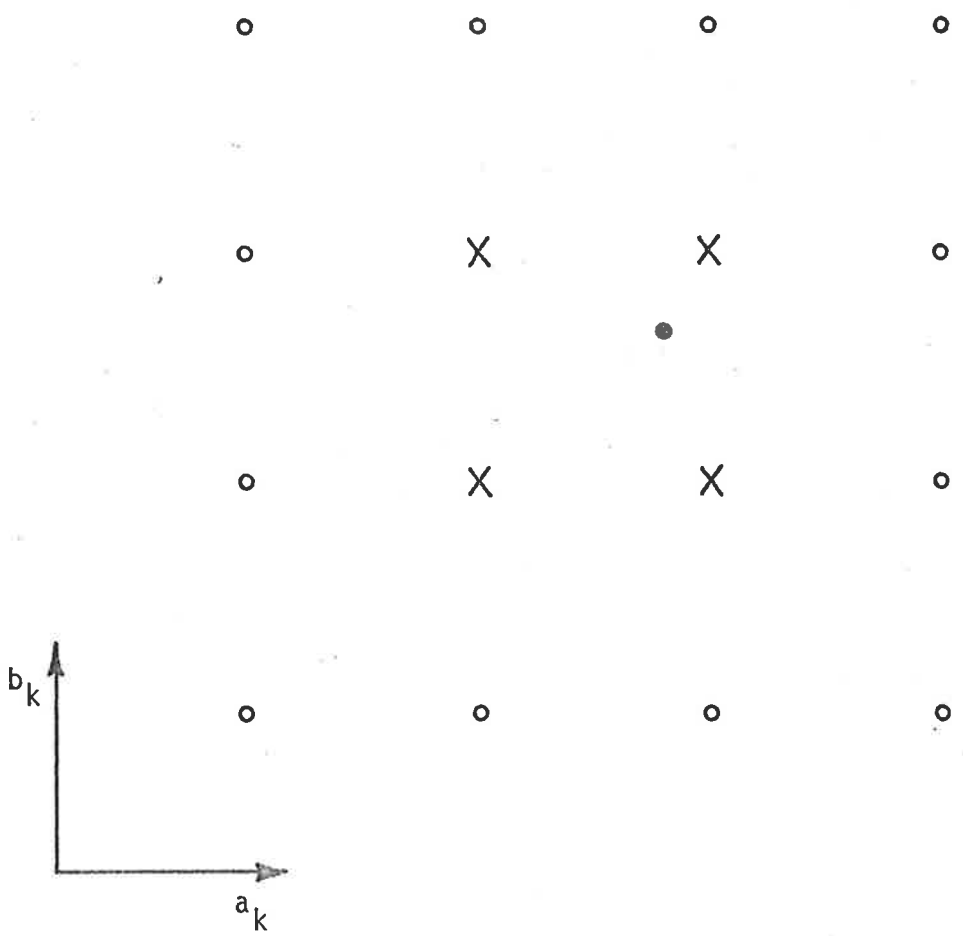


FIGURE 6.8 Trial Points in (a_k , b_k) Plane, Brglez Algorithm

- ● continuous optimal point
- X 4 nearest discrete points
- o 12 additional discrete points

the discrete neighborhood set for every possible pole and zero pair. Then for each such pair the four lowest values of objective function found are averaged, and the pole or zero pair with the *highest* average is chosen to be discretized next. A simpler rule is sufficient for selective (piece-wise constant) filters. The numerator coefficients (connected with the zeros) are discretized first, and in increasing order of the distance of the zero from the mid-band frequency (zero for lowpass filters, π for highpass, and the arithmetic or geometric mean for bandpass filters). Denominator coefficients are then discretized, in increasing order of the distance of the relevant pole from the unit circle. Brglez found that both criteria yielded identical results with typical selective filters.

A final step is suggested by Brglez to further improve the filter. This is an enumerative search over all combinations of discrete coefficients which were considered during the process. If, say, only 2 feasible coefficient pairs were tried at each stage, the total number of combinations would be 2^{2K}, and the enumerative search may be a practical proposition.

6.5.4 Minimization of a Function of Wordlengths

The optimization methods considered so far in this section all have in common the characteristic that the objective function is a measure of the deviation of the actual filter response from some desired response. In the case of spectrum shaping filters the *ideal* (infinite wordlength) design will usually have itself been produced by a (continuous) optimization process, and the objective function used in that process serves

during the discretization phase too. This type of filter is also characterized by having no particular function value which separates "acceptability" from "non-acceptability", so that the designer must make an *a posteriori* decision regarding the wordlength to use after he sees the best that can be achieved for each wordlength. On the other hand, a typical piecewise-constant (or "brick wall") filter usually has rigid specifications on passband ripple and stopband attenuation. The ideal design may or may not be produced by optimization but in any case an objective function (usually of the "maximum-modulus" type) may easily be defined for use in the discretization phase. In this case, some particular value of the function corresponds to the specification being met with zero margin, so that a true *minimum-wordlength* solution exists. However, the methods for wordlength minimization considered up to this point proceed stagewise, with some particular wordlength being assumed at each stage. The solution eventually found satisfies the constraint of equality of all coefficient wordlengths. In the case of very-high-speed hardware digital filters, there may be no need to include such a constraint; if each coefficient had to have its own dedicated multiplier there would be a cost benefit in using the smallest satisfactory wordlength for each individual coefficient. Bandler, Bardakjian and Chen (1975) have considered this problem. The magnitude response of the filter does not appear in the equation defining the objective function; rather, the magnitude at each frequency is *constrained* by the specifications. The objective function is a function of the coefficient wordlengths (such as their sum). This is minimized subject to integer constraints on the wordlengths and on the (scaled) coefficients, and to inequality constraints on the frequency response (at a reasonably dense set

of spot frequencies). The method of Bandler *et al.* is classifiable as a "formulation A" approach (as defined in chapter four of this thesis) whereas all others considered are "formulation B". The Bandler *et al.* method is applicable only when the response has rigid upper and lower specifications, which in practice usually includes only selective ("brick wall") filters. The method uses a technique of Bandler and Charalambous (1974) to transform the continuous constrained minimization problem into an unconstrained problem, and Fletcher's (1970) quasi-Newton method to solve the latter. Discretization is performed by a branch-and-bound search. Because of its complexity this method is unlikely to be practical except for filters with a very small number of coefficients; the authors quote computation times of up to 5 min. for five coefficients. The usual problem of increasing dimensionality is compounded in this approach because the parameter vector contains the wordlengths in addition to the coefficients themselves.

6.5.5 Interactive Approach for Very Severe Quantization

In order to complete this survey of methods for discrete optimization of digital filter coefficients, it is worth mentioning one more. Hadjifotiou and Appleby (1976) have proposed a direct graphical technique applicable when the quantization is very coarse. The ideal pole and zero positions in the z-plane are computed and overlaid on the grid of allowable poles (zeros). An interactive computer system incorporating graphics is suggested as the ideal environment. Since with very severe quantization none of the poles (or zeros) are

likely to wander by many steps from the ideal values, candidate pole (zero) positions can be readily picked manually. The computer system would include also a facility for displaying graphically the achieved response after such a selection to enable the designer to evaluate the results. The authors further propose that the four candidate poles nearest to each ideal pole be tried in turn, and the best one (in terms of a maximum modulus error criterion) kept when moving on to discretize a new pole (zero). This could usefully be automated, so that an interactive implementation, though desirable, is not strictly essential. This procedure is obviously applicable to any of the special filter structures considered in section 6.3 (apart from wave digital filters); the difference is in the grid of allowable poles (zeros) on which the ideal points are overlaid.

6.5.6 Summary

In summary, a great many methods have been proposed for the discrete coefficient optimization problem. They range from general integer programming techniques which are virtually guaranteed to find the true optimum for a given wordlength (but which require vast amounts of computer time), to those in which coefficients are rounded off one-at-a-time or by simple searches in two-dimensional subspaces. The latter methods are usually guided by "reasonable" assumptions regarding the response behaviour of digital filters.

The most practical methods seem to be of intermediate complexity, involving some kind of randomized search procedure. There is no guarantee of finding the true optimum, but a variety of "slightly sub-optimal" solutions can usually be found

in a reasonable time. Randomization is essential to ensure that the procedure does not become trapped at a poor local minimum, but some kind of a "learning" or "direction-finding" search is probably preferable to a pure random search. In the remainder of this chapter, a new method of this type is considered.

6.6 A New Method for Discrete Coefficient Optimization

6.6.1 Published Presentations

A new method for the coefficient optimization problem was introduced by this author at a conference (Smith, 1977) and later described more fully in a paper (Smith, 1979). Both items (to be referred to briefly as the 1977 paper and the 1979 paper) are reproduced as the next few pages of this thesis.

Smith, N. I. (1977, August). Optimum quantized coefficients for digital filters. In *Convention Digest, institution of radio and electronics engineers, Australia, 16th International Convention and Exhibition.* (p. 289-291). Melbourne, Victoria.

Smith, N. I. (1979). A random-search method for designing finite-wordlength recursive digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 27*(1), 40-46.

The essential features of the new method have been expounded in most detail in the 1979 paper. The 1977 paper has been reproduced here also because it is the first publication of the method and because the emphasis is slightly different. The remaining parts of section 6.6 are to further clarify certain points and to place the new method in the overall framework established in this chapter.

6.6.2 Basic Philosophy - Comparison with Other Methods

The aim of the new method is to attempt to exploit the well-behaved (and further, quadratic) nature of the objective function which, at least in theory, prevails near the continuous minimum (provided that the function is twice differentiable). For this approach to be fruitful the quadratic approximation would have to be valid over a region including a large number of feasible (integer) solutions and preferably including the true discrete optimum. Intuitively, this assumption will become more valid with increasing wordlength, because the feasible mesh size is continually refined while the contours of the objective function on which the mesh is superimposed remain the same. However, for long wordlengths discrete parameter optimization is largely superfluous. The theory of (Taylor series) function approximation has guided the formulation of the method, but it is *experiment* which indicates that it is indeed valid for usefully short wordlengths.

No previously-published methods have so directly considered the behaviour of the objective function in a region of space surrounding the continuous optimum. Branch-and-bound methods

use the well-behaved nature of the function implicitly; they usually succeed because the function *is* convex in a neighborhood of the continuous optimum.

On the other hand, search methods which rely only on feasible evaluations (such as those of Avenhaus (1972b) and Steiglitz (1971)) virtually ignore the well-behaved nature altogether. They are essentially unable to discover and adequately to use, information about the manner of variation of the function, except when remote from the (continuous) optimum. To see why this should be so, consider the following intuitive argument:
In a sense, the feasible mesh is always too coarse for a univariate search, regardless of the wordlength. As may be seen from figure 2 of the 1979 paper (for example), if a discrete optimum exists and is separated from the continuous optimum by several quantization steps, and the function *is* quadratic, then the contours *must* be elongated. This is a two-dimensional example, but in problems of higher dimensionality the contours must still be elongated at least in some directions, because otherwise the discrete and continuous optima would be closer.
It follows that many univariate steps from points near the long axis (or axes) of the contour system will "jump over" the contour axis, and for this reason many of these points will be discrete local minima. Roughly speaking, the more elongated the contours, the more local minima there will be, and the smaller the likelihood that a univariate search method will ever reach the true discrete optimum. This is not to say that the univariate search methods are not useful, but they are useful because many of the *suboptimal* vectors yield adequate designs, not because the true optimum is usually found.

The above intuitive objection is based on elongated contours, and would be removed if they were not elongated. However, since the sensitivity of the filter response to each coefficient is usually quite different, such a situation would be exceptional. Furthermore, it is just the situation in which discrete optimization would be of least value, because the rounded solution would be quite reasonable, if not optimal.

There is no problem with contour-jumping with the new method. The procedure is not really a "search", but a sequence of random trials. Discrete local minima go unrecognized as such and cause no special problems. In fact, the efficiency of the subsequent computation is probably increased as each better one is found, because the bounding contour shrinks. The discovery of the discrete optimum is a matter of chance, but with the new method the probability of finding it is *not* reduced by the presence of large numbers of suboptimal local minima.

6.6.3 Area of Applicability

It has been mentioned that the new method requires the objective function to be differentiable. Since the Hessian matrix is used, it must actually be twice differentiable. A further requirement is that the continuous optimum have been located to a fairly high degree of accuracy, because this determines the origin of all the search vectors and is of central importance. The method is thus ideally suited to problems (for example, the "formulation B" approximation of an arbitrary magnitude response) which use a least squares or

least pth error criterion and a quadratically-convergent gradient method for minimization.

The method is not directly applicable to filters (usually of the "brickwall" variety) which have been designed via the bilinear transformation of a classical s-plane design or by minimizing a maximum modulus objective function. This may seem a very severe limitation, but there are ways to reformulate the approximation problem to take advantage of the new optimization method. What is required is that a suitable differentiable objective function be formulated, and that the existing ideal design be modified slightly (if necessary) to minimize the new function. Construction of the function is a trial-and-error process, but with experience it is readily possible to "concoct" functions for which the minimizing design is very little different from the original design. The function could be of least pth form to maintain a close simulation of the original minimax objective, but with a good selection of the discrete frequency set on which the function is defined, a sum-of-squares function could serve just as well. An example was seen in chapter five, example D, for which a least-squares-optimal filter closely similar to an existing elliptic filter was produced. One guiding principle in this work is that there should be an approximately equal number of frequency samples per cycle of the response error function.

In using such a "secondary" objective function with the discrete minimization algorithm it is tacitly assumed that "good" discrete solutions of the secondary problem are also good solutions of the original approximation problem. It is

thought that in many cases this would be true, but this is a matter of conjecture. A further "hybrid" approach may also be worth trying in cases for which a maximum modulus objective function is defined. The secondary (differentiable) function could be used to determine the search directions and to predict the value of the maximum useful radial distance d_{max}. However, the results of the trials at feasible points would be sifted on the basis of the maximum modulus function.

6.6.4 Adequacy of the Quadratic Approximation

Assuming that a differentiable objective function is defined, the success of the new method in finding the discrete optimum (and to a lesser extent, in finding good suboptimal discrete solutions) would seem to be dependent on the quadratic approximation's being adequate out to distances of several units from the continuous optimum. All of the examples for which Charalambous and Best (1974) quoted branch-and-bound results were repeated using the new method, and in all cases the optimum was found. This provides an indirect indication that the approach is valid.

More direct verification of the quadratic nature of the function may be shown by actual evaluation along lines which pass through the continuous optimum. Figure 6.9 shows an example for the design by least squares of a single-section wide band differentiator. This is similar to example 1 of the 1979 paper, except that the dimensionality is reduced to 4 by choosing the gain factor optimally for any given values of the coefficients. (the gain is not constrained to take an integer value). The

distance is measured in multiples of the quantization step size for 8-bit coefficients (i.e., six fractional bits). In this example the discrete optimum has coefficients $a = {}^{-23}/_{64}$, $b = {}^{-41}/_{64}$, $c = {}^{53}/_{64}$ and $d = {}^{6}/_{64}$, an objective function value of 2.934×10^{-4}, and is located at a (Euclidean) distance of 3.429 units from the continuous optimum. Figure 6.9 shows the actual variation of the function along a line joining the continuous and discrete optima, and extended in both directions (solid curve). The dashed curve is the quadratic function found from equation (9) of the 1979 paper. It can be seen that the two curves correspond quite closely out to several unit radii.

Although the "quadratic approximation" provides the basis for the development of the present method, and although the approximation has been shown by example to hold fairly closely over a usefully large region, it should be mentioned that the algorithm includes several features which counteract the effect of errors in this approximation. Firstly, regarding the angular distribution of search vectors; the information in the Hessian matrix is merely used to *bias* the probability density function, not to restrict the choice of vectors. Thus no region of the space is excluded from the search, and the lower the value of the bias exponent (q) the less reliance is placed on the quadratic approximation.

The assumption of quadratic variation of the function with radial distance is used in another way; in the determination of d_{max}, the distance to which it is expected that a vector should be followed in order to flush out any likely candidate discrete solutions. However, this calculation is not based on

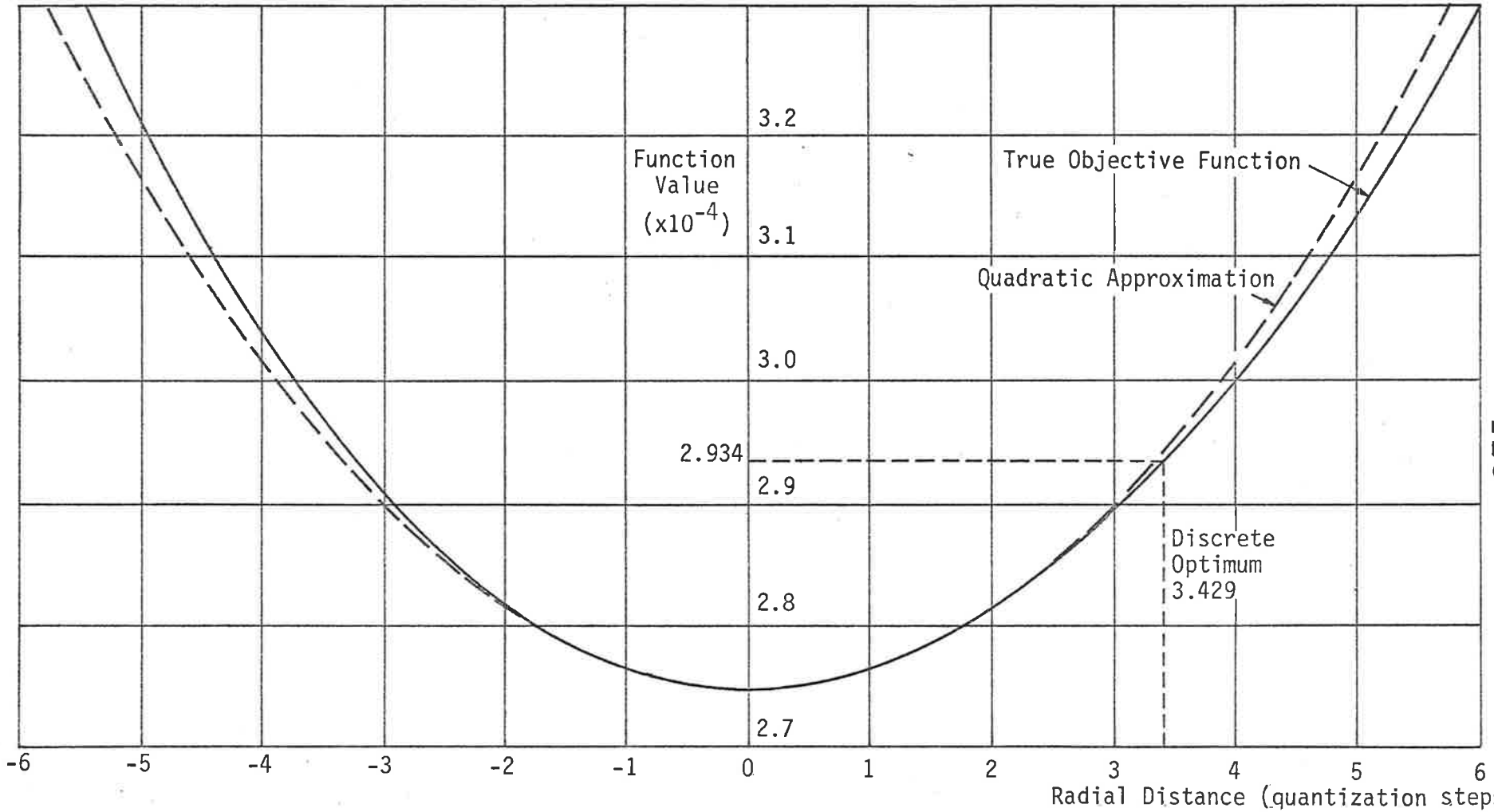


FIGURE 6.9 Objective Function and its Quadratic Approximation

4-dimensional example, $q = 2^{-6}$

the Hessian matrix but on an actual evaluation of the function at a finite distance L from the continuous optimum. (L is equal to the distance to the "rounded" feasible solution). ^{d}max is thus derived from a quadratic fit over a *finite* region of (one dimensional) space, calculated afresh for each search vector, and the error will usually be smaller than that resulting from the truncated Taylor series.

A further feature makes errors in ^{d}max less important. A factor g is introduced by which the predicted ^{d}max is increased before it is used. Thus all radial searches are carried outwards to a somewhat greater distance than that predicted to be necessary. In all the work reported this value g has been taken as 1.1, but in some problems it may be advantageous to make it larger. The price to pay for so gaining immunity from approximation errors is in increased computer time, because more trials will be made per vector.

6.6.5 Calculation of the Hessian Matrix **H**

In the examples reported in the 1979 paper the continuous minimization was performed by Powell's (1964) process using the value of the function only (no derivatives). The Hessian matrix was then calculated by a finite-difference (perturbation) technique again involving function value only. The success of the method indicates that such a procedure is adequate, although it is necessary to introduce a safeguard against the appearance of negative eigenvalues, as described in section V of the 1979 paper.

In that paper it was also mentioned that the eigenvalues and eigenvectors of **H** could be more easily and accurately found

from the positive-definite approximation to $\mathbf{H}^{-1}$ which is available after convergence of the Davidon-Fletcher-Powell algorithm, if that had been used instead. The same of course applies to any quasi-Newton method. A third alternative would be to calculate **H** directly from the analytic expressions derived in chapter four of this thesis. This is unlikely to be significantly superior to the use of the quasi-Newton iteration matrix, and if a quasi-Newton method had been used the additional programming to calculate **H** would not be justified. However, **H** could readily be calculated directly if a second-derivative minimization routine were in use.

6.6.6 Selection of the Bias Exponent q.

The search vectors are computed as random, but biassed, linear combinations of the N eigenvectors of **H**, according to equator (10) of the 1979 paper. A single parameter (an exponent) q has been used to control the amount of bias towards eigenvectors corresponding to small eigenvalues. This method of biassing has been chosen because of its simplicity, and it seems effective, but it may be that other strategies could be developed which would improve the angular distribution of vectors.

There is probably no value of q which is optimal for all problems. Generally, values less than 1 seem appropriate because the eigenvalues of **H** usually vary over several orders of magnitude and it seems ill-advised to concentrate the search in just one direction, especially if the true optimum is sought. However, it is quite likely that substantial improvements over the "rounded" solution would be produced very quickly by searching just the subspace of one or two "sensitive" eigenvectors. This would improve the efficiency of any subsequent, less concentrated,

search, because the area explored shrinks with every improved solution. Hence, it is thought that q should begin large and become smaller. A useful measure of the appropriateness of the value of q is the proportion of vectors which are abandoned after the first exploratory evaluation. If there are too many of these, computer time is obviously being wasted. If there are too few, or none, it could be that the search is too concentrated and is ignoring regions of the space where the solution could lie. If the purpose of the search is the discovery of the true optimum, it is recommended that q be set such that a steady proportion of about 20% of "abandoned" vectors is being produced. This "fine-toothed-comb" phase of the search should follow an introductory phase with a larger value of q, whose purpose is to shrink the bounding contour quickly. Such a strategy should increase the likelihood of finding the discrete optimum. However, there is no guarantee of finding it; if in fact it is found there is no indication of optimality.

In the more realistic situation where all that is desired is an "almost optimum" discrete parameter vector, the fine search may not be required at all, because the value of the objective function at the continuous optimum provides a known lower bound on possible discrete solutions. If the procedure is implemented on an interactive computer system, with a pause after each improved solution found, the designer can readily make the decision to stop or to continue.

6.6.7 Selection of Radial Steplength s

The intent of the radial search method is that a discrete parameter vector be tested as a candidate solution if a search vector passes "close" to it (that is, enters the unit cell of which it is the centre). To implement this without undue computational burden, it is necessary that the radial vector be explored in an outward direction in a stepwise fashion and the trial feasible points be generated by rounding off all parameters at each step. If the steplength were too large, many trial points could be missed. On the other hand, if the steps were minute, many successive steps would lead to trials at the same point. At first sight it would seem important to optimize the choice of s; it would also seem that the optimum value would depend on dimensionality, perhaps varying as $\sqrt{N}$ (because $\sqrt{N}$ is the maximum corner-to-corner distance of the unit hypercube). However, it is probably far better to use a value of s small enough that trials are virtually never missed (say 0.1), and build in a test to prevent a re-trial whenever second and subsequent steps round off to a discrete point which has already been tested. Quite small steps are justifiable because the computer time involved in the radial step calculations is negligible compared with that for a function evaluation.

6.6.8 Storage of Previously-tried Solutions

It is obviously wasteful of computer time to perform several function evaluations at the same point. In the context of a single search vector it is very easy to prevent this (as explained above) because it is only necessary to test that the

rounded coordinates do not duplicate those *last* tried. However, to extend the safeguard against repetition to *all* vectors, a lookup table is required, containing the coordinates of points already tried. The feature produces a net saving provided that successful table searches *do* occur, and do so sufficiently often that the average time spent in table searching is less than the time necessary for a new function evaluation.

For a given problem (and a given value of q) there will be an optimum table size (beyond which new entries should not be added). Furthermore, the *order* of entries in the table is important; there will be more likelihood of repeated evaluation at points close to the continuous optimum (because the search vectors will be less widely spread) and at points in the subspace of the sensitive eigenvectors (because more search vectors will explore this region). Such entries should occur *early* in the table to decrease the average search time. Fortunately, this may be ensured in an approximate sense by searching the table in the order in which it was created.

The matter of optimum table sizes has not been studied extensively, but several observations may be made. Firstly, it *does* seem to be worthwhile to store at least the first few points tried. Secondly, it may be a sensible strategy to allow the table to grow without limit (except that imposed by the computer memory) but periodically to adjust the proportion searched, and to determine this by monitoring the success rate. Such dynamic optimization of the algorithm could readily be carried out automatically.

The discrete optimization problem has a very useful characteristic which enables the "previous trials" table to be implemented in a very efficient manner, from the point of view of memory requirements as well as search time. The discrete optimum will seldom lie more than a few unit steps distant from the "rounded" discrete parameter vector, in any coordinate direction. Thus any point tried may be represented by a vector of small integers (to be added to the rounded vector as a base). If a range of, say, ± 16 units is allowed, then only five bits are required per coordinate. If, for example, a 60-bit computer is used, all the information for a given point may be packed into a single computer word when the dimensionality is less than 13. Two words would allow problems with N up to 24. The table search requires only equality comparisons which are easily done over the entire word or words, with no unpacking required.

6.6.9 Multimodality

All of the foregoing discussion has implicitly assumed that the discrete optimum lies in a small region surrounding *the* continuous optimum. However, if the objective function is not unimodal it is of course possible that the discrete optimum is associated with one of the suboptimal (continuous) local minima. Thus if several local minima have been found in the continuous domain it may be necessary to repeat the radial random search for each one. In such cases it is advantageous to begin with the *best* local minimum, because as soon as a discrete solution is found with a lower objective function value

than a suboptimal local minimum, it becomes unnecessary to perform the search about that minimum. No discrete solution found in such a search could possibly be better than that one already found.

In all practical cases for which multimodality has been observed by this author, one local minimum has been so clearly superior that only one discrete search has been necessary.

6.6.10 Constraints

Several remarks were made in the 1979 paper regarding inequality contraints, such as the filter stability requirement. Firstly, the continuous optimum could easily lie just inside the boundary of the feasible region and it could happen that search vectors cross the constraint boundary. This need cause no special problems when optimizing filters (particularly in the cascade form), because it is easy to test for constraint violation and simply to abandon the search vector after the infeasible region is entered.

Further remarks were made about the difficulties associated with constraints which are *binding* at the continuous optimum. In such a case some of the first partial derivatives of the objective function are nonzero, and the quadratic approximation is not valid. Two suggestions were made to get around the problem - a penalty function approach and an enumerative search. It is now realized that a situation like this will not normally occur with stability constraints, because the continuous minima of interest are always interior to the feasible region (see section 2.3.2). However, there may

in some cases be constraints on the coefficients which arise from other considerations. As an example, consider the all-pass group delay equalizer of example C in chapter five. The ideal (infinite precision) filter found by continuous optimization had several coefficients which were greater than 2.0, but only slightly so. The representation of the value 2.0 (or any greater value less than 4.0) requires an extra bit, so that it may be beneficial to seek instead a solution for which all coefficients are constrained to be less than 2.0. The continuous optimum would then involve binding constraints and one or other of the modified solution methods would be necessary.

6.7 Summary

In this chapter the wide field of finite-wordlength digital filter design has been reviewed. The matter of quantization of the *data* was touched on briefly in section 6.2 and indicated as an area for the application of a type of optimization method.

The amelioration of the problems arising from *coefficient* quantization has seen much fruitful application of optimization ideas. Published work in this area has been reviewed in section 6.5. Sections 6.3 and 6.4 were not specifically connected with optimization but served to introduce further ideas relevant to coefficient quantization.

In section 6.6 a new method for coefficient optimization was dealt with. This had been previously introduced in published papers (reproduced in that section). It is thought to be superior in many ways to alternative methods which have been proposed. In the latter part of section 6.6 some further discussion relating to the new method was presented.

CHAPTER SEVEN

A HIGH-SPEED IMPLEMENTATION OF SYSTEM IDENTIFICATION BY OPTIMIZATION TECHNIQUES

7.1 Introduction

This chapter differs somewhat in flavour from the preceding ones. Its purpose is to illustrate (by means of an example) some of the considerations involved in the implementation of system identification by optimization techniques using fast, cheap, short-wordlength digital hardware. The work to be described includes the development of both the hardware and the software.

At various places in this thesis reference has been made to the possible application of optimization techniques to real-time system identification problems. In a typical application, the output signal from some physical system would be sampled and digitized, and stored in a memory. The concurrent *input* to the system could be a special test signal or the signal occurring during normal operation. If it were the latter, and if it were available to a sensor, it would also be sampled, digitized and stored. Digital processing would then be performed, with the stored time sequence(s) as input data. The aim of the processing would be to identify the parameters of a "digital filter" model of the system under study; that is, to find appropriate values of the parameters such that the filter model, when fed with the known or measured input sequence, would produce in some sense an approximation to the observed output sequence. The *form* of the model normally would be decided in advance, and would be based on whatever was known about the physical principles on which the system operated.

The principal use of such processing is in keeping track of the parameters of a system (or, rather, of its model) when the system characteristics are changing with time. The usual assumption necessary with system identification algorithms is that the characteristics do not change appreciably over the time necessary to accumulate sequences which are long enough for reliable analysis (the *frame* time); however, with many signals of interest (speech, for example) the changes are sufficiently rapid that data must be accumulated continuously and analysed in contiguous or even overlapping frames. This means that if changes to the system are to be followed in real time, the processing must be performed in at least the same time as (and concurrently with) the accumulation of the next frame.

Real-time system identification is desirable for many different applications. When the signal is speech (i.e. when the system being identified is the human speech production apparatus) possible applications are in compressed-bandwidth telephony (the model parameters can be digitally encoded with far fewer bits than the signal itself), in talker verification, and in automatic speech recognition (for verbal entry of computer commands, for example). If the system were some controlled "plant" (e.g. an industrial process) whose parameters were liable to change with time, the system identification would be a necessary component of the overall control strategy.

Requirements for very high speed in system identification greatly influence the choice of the computer hardware and the complexity of the algorithms which may be considered for the task. The other major factor is cost. Seldom would a large mainframe

computer be available for dedicated operation; on -line system identification is much more likely to be mini- or micro-computer based. Long-wordlength floating-point arithmetic (having high precision and large dynamic range) may not be available (owing to cost) or may have to be avoided (because it is too slow).

A study has been done of the implementation on fast, cheap, short-wordlength computer hardware of system identification by a gradient-based mathematical optimization method. The computer used is the "General Arithmetic Signal Processor" (GASP), built in the Department of Electrical Engineering at The University of Adelaide during the years 1974-1978. In addition to writing the programs for the optimization study, I have, during my period of Ph.D. candidature, made major contributions to the design, testing and documentation of the hardware and "firmware" of this machine. Further details are given in Section 7.2.

An initial aim of the study reported here, to produce a system to perform useful analysis of speech signals by optimization in real time, has not been realized. However, the study is still of interest because:

a) It goes some way towards indicating the order of hardware speed up (through increased paralellism and/or improvements in logic speed) required to bring such speech processing within reach.

b) The system as developed may be useful for less demanding (slower) system identification problems.

c) It represents the only attempt known to the author to implement a sophisticated gradient-based optimization algorithm on such a machine.

Most previous optimization studies (including those of earlier parts of this thesis) have used the high-precision, floating point capabilities of large, general purpose mainframes. Conversely, attempts to make filters (or system models) which adapt to changes in signal characteristics in real time, usually employ less sophisticated optimization techniques (e.g. variants of steepest descent). The present work shows that it is indeed possible to implement matrix arithmetic and the other operations necessary for more sophisticated algorithms on simple short-word-length hardware, and to do so in an efficient manner.

The study is incomplete. No examples of useful application can be quoted. Nor are the computation times observed (with test signals) necessarily a reliable indication of what may be achieved. Speedups may possibly be gained from re-design of sections of the software, and certainly by minor changes to the machine's hardware.

In further developing a system of the type discussed here, considerable advantage could be taken of more up-to-date microelectronic technology, including LSI and VLSI. The existing machine, GASP, has some interesting architectural features which could usefully be incorporated into a future version, and the present study has shown that there are also ways in which it could be substantially improved. Specific comments along these lines are made in Section 7.9.

7.2 The General Arithmetic Signal Processor (GASP)

7.2.1 Description of the Machine

GASP is a flexible, programmable signal processor, incorporating several novel features particularly with regard to data paths and the organization of the arithmetic section. It was built in the Department of Electrical Engineering at the University of Adelaide in the years 1974 to 1978. The technology is predominantly Shottky-clamped bipolar integrated circuit logic at small and medium-scale levels of integration (SSI and MSI).

Most of the original design of the machine was carried out by Dr. J.A.V. Rogers. He was able to supervise the wiring and testing of most of the arithmetic section and data memory and to produce detailed circuit diagrams for the construction of the control section before he left the project (and the University) in June, 1976. Subsequent work on the machine was performed by B.D. Ackland (who designed and built the cache program memory system) and by D.S. Fensom and myself (who together built and tested the remainder of the hardware, and wrote system "firmware" (assembler, loader, etc.) and applications software.

The machine as finally commissioned is described by Fensom, Smith and Ackland (1979) in a paper reproduced as the next few pages of this thesis. Substantial re-design of certain parts of the machine already built by Dr. Rogers was found to be necessary; they proved inadequate when the machine was tested as a whole. Although much of this work was done in co-operation with D.S. Fensom (and some was done entirely by him), I claim certain aspects as being wholly or substantially my own work. These are listed in Section 7.2.2.

Fensom, D. S., Smith, N. I. & Ackland, B. D. (1979). GASP: a fast general purpose signal processor. *The Institution of Engineers, Australia; Electrical Engineering Transactions, 15*(1), 26-30.

7.2.2 Specific Contributions of the Author

My specific contributions to the hardware, firmware and documentation of GASP are listed below. The type of involvement is indicated by letter codes, as follows:

O: Origination of concept

D: Detailed Circuit design

BT: Building and testing

SR: Substantial re-design

SD: Software design

P: Programming and debugging

(1) Radix-4 floating point hardware (D, BT)

(2) The normalizer/shifter (a sub-processor running under its own clock control)(D, BT)

(3) Data memory control logic (SR, BT)

(4) Indexed and bit-reversed addressing (O,D, BT)

(5) Store instructions for counters (to enable re-use in inner loops) (O,D, BT)

(6) Interface to the host NOVA minicomputer (including both programmed transfers and direct-memory access). I/O protocol design. (D, BT)

(7) Digital-to-analog converter (O,D, BT)

(8) Real time clock (O,D, BT)

(9) Instruction decoder (SR, BT)

(10) GASP Relocatable Linking Loader (O, SD, P)

(11) GASP Library Directory Editor (O, SD, P). "Library Directories" are disc files searched by the loader program to determine the locations of the relocatable binaries required to be linked.

(12) GASP Overlay Editor (for examining and modifying portions of object programs and program memory dumps) (O,SD, P)

(13) GASP "Executive" Operating System (O, SD, P). A short (320_8 word) program, GEX, is resident in GASP lower program memory at all times during normal operation; user programs are loaded in the address space above this. A suite of short NOVA programs (which are invoked by console commands or by FORTRAN or machine language calls) enable interruption of any GASP user program (returning control to GEX) and then interaction with GEX to perform various functions including:

a) examination and modifying of any location in program memory.

b) dumping of any portion of program or data memory to a (NOVA) disc file, or to the console.

c) loading of a new GASP overlay (user program) from a disc file.

d) starting GASP program execution at any address.

e) examining the contents of selected GASP registers at the time of the last user-programmed breakpoint (that is, deliberate return to GEX).

f) re-starting GASP execution at the address following such a breakpoint.

The executive operating system provides both flexible run-time control of GASP and comprehensive program debugging facilities.

(14) Utility program PGF (O, SD, P)

This program provides facilities for conversion of numeric

portions of GASP dump files to decimal format for printout, and also allows plotting of such information on a flat-bed plotter or graphics display terminal (primarily a debugging aid).

(15) The GASP Simulator GSIM (O, SD, P). GSIM is a NOVA software simulation of GASP. It enables debugging of GASP program segments without actually using GASP (limited to program segments not involving GASP/NOVA interaction).

(16) The "GASP Utility Package" (O, SD, P). A unified framework has been established within which GASP may be used as a high-speed signal processing peripheral to NOVA, performing common well-defined tasks (such as autocorrelation calculations and Fast Fourier Transforms (FFT)). Users do not need to be GASP programmers, and use the machine through simple FORTRAN calls. The present facilities of the package involve mainly the FFT: however, any proficient GASP programmer could expand the facilities within the established framework, which handles data transfers, error codes etc.

7.3 Aims and Overall Description of Study

7.3.1 General

The initial aim of this study was the exploration of the use of optimization methods in formulating pole-zero models of speech. As much advantage as possible was to be taken of the fast processing facilities of GASP.

The "core" of such a processing program has been developed and tested. However, results are presented here only for a much-simplified synthetic test signal with some speech-like features. Successful application to real speech would involve much further program development to overcome a number of difficulties, which are alluded to in Section 7.8. In addition, the speed performance is inadequate to enable speech modelling in real time. It is likely that such real-time processing will remain out of reach except perhaps with the development of very sophisticated special purpose processors employing a much greater degree of hardware parallelism.

7.3.2 Speech and Speech Models

Very much success has been achieved with *all-pole* models for speech, obtained by linear prediction methods (Markel and Gray, 1976). Although the effects of (a) radiation at the lips and (b) the finite-duration glottal excitation pulse for voiced speech in theory require zeros for their representation, a total of 3 or 4 *poles* (over and above those necessary to represent vocal tract resonances) seems to serve just as well (Rabiner and Schafer, 1978, p. 419). However, for *nasal* speech (the consonants /m/, /n/ and /ŋ/, and the nasalized vowels which precede or follow

them), and for some unvoiced speech, zeros are considered to be much more significant perceptually (Flanagan, 1972). An all-pole model can still approximate the spectral characteristics to an arbitrary degree of accuracy, but the number of poles required may be quite large (Makhoul, 1975). This is undesirable from a data compression point of view, and several workers have investigated techniques for the more general *pole-zero* modelling.

As mentioned in section 7.1, system identification normally requires knowledge of both input and output signals. However, in the case of speech, only the "output" (the recorded speech waveform) is available. Linear prediction has the property of matching the spectral *envelope* of a signal without being sensitive to fine structure caused (e.g.) by periodic excitation (Makhoul, 1975) and so it may be directly applied to an arbitrarily chosen segment of speech. However, identification of individual poles and zeros requires either (a) frequency-domain methods in which a model is fitted to some "smoothed" version of the speech spectrum, or (b) some method for estimating the notional "impulse response" of the overall excitation-vocal tract-radiation system. Olive (1971) has used an optimization method to model speech in the frequency domain, although he considered only an all-pole model. In principle, the optimization methods considered in section 4.6 for magnitude-only design of IIR digital filters are applicable. (It is normally considered unnecessary to preserve phase information in models for speech, owing to the relative phase insensitivity of the ear (Flanagan, 1972).)

The time-domain methods fall into two classes:
(a) those using pitch-synchronous analysis (voiced speech only) and
(b) those estimating an impulse response by homomorphic decon-

volution. Steiglitz and Dickinson (1977) give an example of pole-zero modelling by pitch-synchronous means, but point out that such a procedure is very difficult to automate because no satisfactory means for locating the instant of glottal opening has been devised.

7.3.3 Homomorphic Processing and Speech Modelling

Homomorphic processing (Oppenheim *et al.* 1968) provides a means for "deconvolving" the periodic (or white noise) excitation and the filtering system which (notionally) shapes the spectrum of the speech signal. In the most general form, the "complex cepstrum" (the inverse Fourier transform of the complex logarithm of the Fourier transform) of a sampled speech segment is computed. The complex cepstrum is a *real* sequence because of the Hermitian symmetry of the Fourier transform of a real sequence. Because the effects of the excitation and the filter are combined by *convolution* in the time domain, they are *multiplied* in the spectrum and so are *additive* in the logarithmic spectrum and in the complex cepstrum. Furthermore, since the filter determines the slowly-varying spectral *envelope* its effect appears at low values of the cepstral independent variable (called the *quefrency*). Periodic, or random, excitation causes *fine structure* in the spectrum and therefore appears at high quefrency. This disjoint nature of the opstral contributions allows ready separation of the two components by means of a "window" which selects only high, or only low, quefrency values.

Since preservation of phase information is unnecessary in most speech applications, it is usual to use (instead of the

complex cepstrum) a sequence called simply the *cepstrum*. In place of the complex logarithm, only the (real) logarithm of the *magnitude* of the signal spectrum is computed, greatly simplifying the calculations. The cepstrum found in this way is an *even* sequence; in fact (Rabiner and Schafer, 1978) it is equal to the even part of the complex cepstrum, and is equally useful in deconvolving the signal components. From the low-quefrency part of the cepstrum an estimate can be made of the filter impulse response; the magnitude spectrum of the signal is preserved (in smoothed form) in this estimate, but since phase information has been lost some *particular* phase must be assumed. It is usual to assume *minimum phase* since this is easy (Section 7.4.4) and corresponds most nearly to the phase of real speech (Oppenheim, 1969).

In the computation of the cepstrum, the Fourier transform operations are usually approximated by Discrete Fourier transforms. This has the effect of introducing some aliasing in the cepstrum, but this can normally be kept within reasonable limits by using a DFT of length 512 or more (Rabiner and Schafer, 1978, p. 365). Time domain techniques for pole-zero modelling of speech employing the homomorphically derived impulse response estimate have been considered by Kopec *et al.* (1977), by Steiglitz (1977), and by Cann and Steiglitz (1978). Models have been produced in the form of a ratio of two high-order polynomials (of order about 10) because there are effective non-iterative means for estimating the parameters of models in this form. Kopec *et al.* considered such a model as their final form, while Steiglitz (1977) suggested that substantial improvement of the parameter

values could be gained by several iterations of the Steiglitz-McBride (1965) "iterative prefiltering" technique.

7.3.4 Features of the Present Approach

The foregoing two sections have provided some background in speech modelling techniques. The technique of homomorphic deconvolution appears most promising because it avoids the need for pitch synchronization and is equally applicable to unvoiced speech. Following the homomorphic processing, the fitting of a model could be done in either the time or the frequency domain because both a smoothed log spectrum and an impulse response estimate are available.

Optimization techniques do not appear to have been used in this application because they are iterative and therefore time-consuming. However, Steiglitz (1977) found that substantial improvement to a pole-zero model could result from application of an iterative technique. Optimization techniques should also be useful in this regard (perhaps more so, as they can expressly seek a reduction in a sum-of-squares error measure, a function not obviously performed by Steiglitz's "iterative prefiltering"), and it is possible that they may be no less efficient computationally. Other possible advantages of the general optimization approach are:

(a) The model may be formulated other than in direct (rational polynomial) form. For example, the cascade form may be desirable from the point of view of efficient speech coding, since it has better quantization properties.

(b) It is possible to use the model parameters from one frame as starting values in the computation for the next frame. This may be more efficient than using one of the non-iterative approaches, and convergence may often be obtained in a few iterations because speech parameters are likely to change little from frame to frame.

It is still necessary to decide whether to employ the optimization method in the time or the frequency domain. In the present study a time-domain approach was investigated. This allows the value of the objective function and its derivatives to be computed by straight forward digital filtering operations (Section 4.7) which are readily implemented on GASP. (However, this is not to say that an optimization of (say) log magnitude could not be implemented and would not be more efficient or more robust. This question is not answered here). A second reason for adopting a time domain approach is that a minimum-phase impulse response estimate may be derived directly from the cepstrum by a recurrence relation, which (at least when implemented on GASP) is more efficient than the DFT which is necessary to obtain a smoothed log spectrum. This computational process is discussed in Section 7.4, and it is believed that this is the first suggestion of using this well-known recurrence relation for this purpose; both the original "homomorphic vocoder" of Oppenheim (1969) and the recent system of Cann and Steiglitz (1978) use the alternative system of DFT - complex exponential - DFT. A third reason for the use of a time-domain approach is that it is not necessary to include a variable gain factor during the optimization, because the "target" impulse

response can readily be normalized to have a unit first sample.

The general features of the present approach to pole-zero modelling may be summarized as follows:

(a) A signal segment is chosen without regard to the location of pitch pulses. It is windowed and its short-time magnitude spectrum calculated by means of an FFT algorithm. The logarithm of the magnitude spectrum is then found and the cepstrum computed from this by an inverse FFT.

(b) A minimum-phase impulse response is computed from the low-quefrency samples of the cepstrum by means discussed in Section 7.4.4. This sequence is denoted Y_m, m = 0,1,2 . . . M (M being some suitable value), is normalized such that $Y_0 = 1$, and is referred to as the "target" sequence. Steps a) and b) are together referred to as the "deconvolution phase".

(c) The parameters of a cascade-form digital filter are found by an optimization method in such a way as to minimize the sum of squares of differences between the target sequence and the impulse response of the model filter B_m, m = 0,1, . . . M. That is, the objective function F is defined by

$$F = \sum_{m=1}^{M} (B_m - Y_m)^2 \qquad (7.1)$$

(where the lower limit of the summation is 1 since $B_0 = Y_0 = 1$ automatically). The parameter values used to start the optimization procedure are taken from the preceding frame. No alternative means for generating starting values has been incorporated; it is a serious

omission and much work needs to be done in this area. The matter is further discussed in Section 7.8.2.

7.4 The Deconvolution Phase

7.4.1 Data Pre-processing

The deconvolution processing for each frame of data is handled by GASP subroutine DCONV, which in turn calls other subroutines to perform the manipulations associated with the Fast Fourier Transform (FFT). Each data frame, of 256 samples, is transferred (in the test cases) from a disc file to a buffer in NOVA core memory, whence it is accessed by GASP, using (NOVA) direct memory access. The frame length, representing 25.6 ms of speech data sampled at 10 kHz, is somewhat shorter than that customarily employed for homomorphic analysis (Oppenheim, 1969) in which at least two pitch periods must be included to allow pitch determination by location of a cepstral peak. However, it is sufficient with the test signals used (Section 7.7), and in any case it is capable of easy extension because DCONV simply fills out the data segment to 512 samples by appending zeros. There are two data pre-processing options available in DCONV - either the mean value of the segment may be calculated and subtracted out, or the signal may be first differenced (that is, pre-emphasized with a filter of the form $1 - z^{-1}$). Following this, the signal segment is multiplied by a Hamming window (of length 256 points) and filled out to 512 samples with zeros. Fixed-point operations are employed in de-meaning (the 32-bit ALU allowing exact accumulation of the sum of all 16-bit data values with no possibility of overflow) and in pre-emphasis. However,

the 32-bit words output from the windowing stage are normalized to retain maximum precision in 16 bits before storage in data memory. A further pass over the data is then made, aligning all values to a common exponent. A block floating-point representation results; the data at this stage has a full-word representation of maximum accuracy even if the raw data was of low level and utilized the 16-bit NOVA wordlength poorly. The use of GASP's limited floating-point hardware capability in this way is very useful in avoiding the scaling problem (that is, the necessity both to retain precision and avoid overflow). It involves shifting words left (to normalize) and then right (to re-align) and so there is a time penalty when compared with a pure fixed-point implementation. However, the shifting is hardware-controlled rather than programmed and can run in parallel with both program activity and data memory access (the normalizer/shifter is an independent processor). In addition, the data may be stored in contiguous locations in data memory allowing maximum use to be taken of autoincrementing pointers. Such block floating-point operations are thus quite efficient. Another example is considered in Section 7.5.3(a).

In a real application (such as a vocoder) the necessary measure of signal level would be made at the data pre-processing stage. In the remainder of the processing the data representation is "centralized" to minimize the chance of floating point over or underflow; the actual signal level is ignored.

7.4.2 Fast Fourier Transform and Associated Operations

The windowed data sequence of 512 real-valued samples is transformed to a short-time spectrum (of resolution about 20 Hz,

assuming 10 kHz sampling) using a subroutine, FFT8, which performs the 256 (complex) point Fast Fourier Transform (FFT). Several well-known auxiliary data manipulations (Coates, 1975) are also performed to allow this "basic" FFT routine to be used with a real sequence of twice the length. When the spectrum has been calculated, the magnitude squared (sum-of-squares of real and imaginary parts, considered sample-by-sample) is computed and the logarithm of each of the resulting 257 samples evaluated as described in Section 7.4.3. Further data manipulations are then made and FFT8 called again (at a different entry point, so that the *inverse* FFT is computed) in order to calculate the 257-point non-negative quefrency part of the cepstrum.

The "auxiliary manipulations" required are of a "vector" nature in the sense that the same operations are required on a large number of pairs of operands. They are also in-place operations - the results may overwrite the operands in storage (except for some minor end-point anomalies which require a few words of temporary storage). The computations are accordingly organized to use a single 512-word array in data memory and to take advantage of autoincrementing pointers in "pipelining" the operations. Unfortunately, some array elements must be fetched in *decreasing* order of address, and GASP possesses no autodecrementing pointer facility (one could usefully be incorporated). Time must therefore be spent in repeatedly re-loading data pointers. However, rather than using the ALU to *calculate* values for the pointers (which would be both slow and difficult, since it is set up to perform the data arithmetic) the pointer load values are *themselves* fetched from an auxiliary data

memory array (of 128 words) which is accessed in autoincrementing fashion. Use of this area of data memory is avoided by all other sections of the program and it needs to be set up once only, before processing begins.

Extensive use is also made of the block floating-point representation as described in Section 7.4.1, both in the FFT routine itself and in the auxiliary data manipulations. The arithmetic elements (ALU and sum-of-products unit) may be used essentially in fixed-point mode since all data have a common exponent; however, the block pre-alignment guarantees high-precision working with absence of overflow.

FFT8 is a decimation-in-time algorithm (Coates, 1975) and prior to its use the 256 complex-valued samples must be shuffled into bit-reversed order of addresses. Ideally, this would be achieved with an autoincrementing data memory pointer whose contents were interpreted in bit reversed order. However, the original design of GASP did not specifically call for efficiency in implementing the FFT; there is no such pointer and to provide one easily would have involved the sacrifice of a useful "ordinary" pointer. It was simpler to incorporate bit-reversed addressing for *program* memory arrays by allowing reverse interpretation of an index register, and this has been incorporated into the machine. Data must therefore be buffered in program memory (a somewhat inefficient operation), but justifiable since such manipulations take only a few percent of the total execution time of FFT8.

The values of sines and cosines required in the FFT computation are held in a program memory lookup table, which

is readily accessed in an indexed addressing mode. The index register doubles as a loop counter; a single instruction (a conditional jump) both updates this and tests for completion of the program loop. Three quadrants of the sine function, tabulated at an interval of $\pi/128$ are included, and so no appeals to symmetry are needed in accessing the table, simplifying the program and improving execution speed. Shorter tables, at tabulation intervals of $\pi/32$ and $\pi/8$ are also included, taking little extra space but improving the efficiency of the early stages of the 8-pass FFT. GASP has a fairly large program memory (8K) and fairly large lookup tables may readily be used with most programs in the interests of execution speed.

The later passes of the decimation-in-time FFT are fairly well suited to GASP implementation since standardised operations are required on sizeable "batches" of operands which may be fetched using autoincrementing data pointers (Coates, 1975, p. 111). However, on earlier passes data pointer load operations take up an appreciable fraction of total computation time. Economy could be achieved here if data pointers which automatically incremented by 4, 8 or even 16 were available. As they are not, the pointer re-loading is unavoidable, but considerable use of lookup tables reduces the amount of associated calculation required.

Beyond the incorporation of a bit-reversed index register, GASP has no facilities which are specifically designed for efficiency in performing FFT's, and it cannot be expected to match "hardware FFT processors" in execution speed. Nevertheless,

most features of GASP can be used to some advantage, and a reasonably efficient algorithm results. The matter of speed is considered further in Section 7.9.

7.4.3 Logarithms

A 257-point magnitude squared spectrum results after the first application of the FFT algorithm; the operation $\frac{1}{2} \log_e$ must be performed on each of these values. Since, however, logarithms to different bases are simply related by a constant factor, the GASP program could be arranged to work to any base, with the appropriate factor (including the $\frac{1}{2}$) applied at the end. Since GASP uses a radix-4 exponent representation, it is natural that the central routine use the base 4. Only normalized, positive mantissas need be considered (since operands may be pre-normalized). The value of the operand's exponent is ignored for most of the processing, and then added as an integer to the value found for the logarithm of the mantissa.

The method selected for the calculation of logarithms was the summation of a Chebyshev series by the recurrence relation (National Physical Laboratory, 1961, p. 77). The range of normalized mantissas to be considered is 2.0 to 8.0(-), but since

$$\log_4 (2x) = \log_4 (x) + \tfrac{1}{2} \qquad (7.2)$$

only the range 2.0 to 4.0(-) requires representation by the series, returning a positive, fractional logarithm in the range 0.5 to 1.0(-). (Operands in the range 4.0 to 8.0(-) are first shifted right one place and $\frac{1}{2}$ is added to the result at the end.)

It is desired that the logarithm of *any* normalized (positive) number be representable in a single 16-bit fixed-point word; the (integer) characteristic requires the leading four bits, hence it is pointless to use a Chebyshev series which calculates the result to more accuracy than 2^{-12}. Thus it is feasible to use 16-bit fixed-point arithmetic in computing the recurrence relation, and this is certainly desirable from the point of view of speed. Simulation on a NOVA minicomputer (testing every possible operand) has shown that the use of a five-term series never gives any errors greater than ± 1 in the least significant bit (2^{-12}). Furthermore, the use of more terms gives no improvement, unless more precision is used in the arithmetic. A five-term series thus gives the most accurate "simple" routine. However, the simulation also showed that a *four* term series never gave any errors greater than ±2 LSB. Since this is substantially faster in execution, the four-term series was adopted.

7.4.4 Calculation of Target Impulse Response

As mentioned in Section 7.3.3, the cepstrum of a signal is an alternate representation of the magnitude spectrum of the signal; it is an even sequence, and is equal to the even part of the *complex cepstrum* (which itself retains both the magnitude and phase spectra of the original signal). A variety of complex cepstra may be constructed from a given (even) cepstrum by adding arbitrary *odd functions* to it; signals reconstructed from such complex cepstra (which is in general possible by the sequence DFT - complex exponential - IDFT) all have the same magnitude spectrum. In speech modelling

and synthesis applications phase is not considered to be of central importance and a variety of methods of estimating an impulse response from a cepstrum may be considered.

The simplest odd function is that which is identically zero. If this is added to the cepstrum (i.e., the complex cepstrum set equal to the cepstrum) a zero-phase impulse response is reconstructed. An alternative is to force the complex cepstrum x_m to be zero for $m<0$, that is to define x_m in terms of the cepstrum C_m by the formula:

$$\left.\begin{array}{lll} x_m = 0 & m<0 & \\ x_o = C_o & & \\ x_m = 2C_m & 1 \leqslant m \leqslant N_c & \\ x_m = 0 & m > N_c & \end{array}\right\} \quad (7.3)$$

The upper limit of the cepstral "window" N_c must be at least low enough to remove the effect of the excitation. The lower the value used, the more "smoothing" is applied to the spectrum; typical values for speech applications are 30 to 60 - in the present study $N_c = 48$ was used. The impulse response constructed from x_m, generated according to (7.3), is "minimum phase" (may be thought of as arising from a "filter" with all poles and zeros inside the unit circle). Tests by Oppenheim (1969) show that minimum-phase speech synthesis is usually preferable perceptually to zero-phase and other alternatives.

Oppenheim *et al.* (1968) have also shown that the impulse response Y_m is for minimum phase systems related recursively to the complex cepstrum X_m by

$$Y_o = \exp(x_o) \tag{7.4}$$

$$Y_m = \frac{1}{m} \sum_{k=1}^{m} kx_k \, Y_{m-k} \qquad m > o \tag{7.5}$$

In the GASP context this relation may be used to calculate the elements of Y_m considerably more efficiently than the alternative scheme involving two more FFT's, since (7.5) can be programmed as a fixed-point sum-of-products operation.

The approach taken is as follows. The overall signal level is irrelevant since the data has been normalized anyhow (Section 7.4.1). Thus there is no benefit in calculating the initial sample Y_o by (7.4) - some convenient fixed-point value may instead simply be assumed. The value used must be small enough for there to be no overflow in the sum-of-products unit in computing any of the remaining terms by (7.5). It is also used as the "unit impulse" during the optimization phase (Section (7.5.3) and must be small enough to prevent overflow at that stage as well. However, selection of an unnecessarily small value would lead to loss of arithmetic precision. In the tests to be reported a suitable value for impulse height (given the symbol IMPI) has simply been selected by trial and error.

In (7.5) each term of the complex cepstrum x_k always appears multiplied by its quefrency index k, and there is no need to perform this multiplication repeatedly. Accordingly, an "associated cepstrum function" γ_k (equal to kx_k) is computed from the cepstrum at the outset, according to

$$\gamma_k = 2kc_k \qquad k = 1, 2 \ldots N_c \qquad (7.6)$$

and stored in *decreasing* order in a data memory array. The value IMPI is placed in the lowest address of a second array. The two operands for each step of the sum-of-products operation of (7.5) are fetched, one from each array, using two autoincrementing data pointers. The scheme is the same for each impulse response term Y_m; only the initial data pointer load words and the loop count must be changed. The reciprocal 1/m is taken from a program memory lookup table and multiplied by the accumulated sum. The result is then placed in the next higher impulse reponse array location, completing the step.

The scheme may be used to calculate a "target" impulse response of any desired duration, independently of the value N_c assumed for the cepstral window. The duration M is selected to encompass a substantial portion of the total signal energy.

7.5 The Optimization Phase

7.5.1 Choice of Optimization Algorithm

The optimization algorithm chosen for implementation was a true Marquardt method. A logical switch in the program allows selection of either the Hessian matrix **H** or the Gauss-Newton matrix **R** .

The studies of chapter five have suggested several reasons why a true Marquardt method is probably an inappropriate choice for real-time system identification by time-domain optimization methods. These may be summarized briefly as follows:

a) If the Gauss-Newton (GN) variant (i.e., the **R** matrix) is used, fast "Newton-like" ultimate convergence is unlikely unless the fit between the model and target impulse responses is very good. Such a good fit could hardly be expected in general with natural signals.

b) If instead the second-derivative (SD) variant is used, there should be no convergence problems due to large residuals. However, as seen with example F of chapter five, the "economical" method of calculating first and second derivatives may not be sufficiently general due to the requirement that zeros remain minimum phase during the entire process. This can be a problem even when the impulse response being modelled *is* minimum phase (as is guaranteed by deriving it by homomorphic processing). If the more general method of derivative calculation is necessary, it has been shown that second-derivative methods are likely to be less time-efficient than quasi-Newton methods.

c) True Marquardt SD methods are usually less efficient than line search SD methods.

However, the decision to implement a true Marquardt algorithm was taken before the results of chapter five were available. The chief motivation was that it was unnecessary to program any line search routine. In the light of the results of chapter five this is of little concern, because simple line searches perform well. However, programming of a *complicated* line search (such as PQI) would have been extremely intricate in GASP assembly language.

One point in favour of the method adopted is that it is easy to study the performance differences between the SD and GN variants when operating on the same classes of signal. In cases for which the GN method *does* converge at a "Newton" rate, it is likely to be the fastest method available (chapter five).

7.5.2 Implementation - Main Program

A simplified flow diagram of the portion of the main program following the determination of the target impulse response is shown in figure 7.1. Calls are made to a number of subroutines, the functions of which are briefly discussed in Section 7.5.3. Several other points relating to the main program require clarification; lower case letters in brackets in figure 7.1 refer to the following notes:

a) CMARQ is the first value of the Marquardt parameter to be tried at each iteration. At the start it is set to some suitable small value FMARQ.

b) The "starting values" for the filter (model) coefficients could be obtained by time domain regression methods (Shanks, 1967; Kalman, 1958; Steiglitz, 1977) but a polynomial rooting routine would also be required to find the coefficients of the cascade-form filter. Another possible approach could be based on peak picking in the cepstrally-smoothed log spectrum. However, one of the chief motivations for trying optimization approaches to the system identification problem is that convergence in a given frame may often require only a few iterations *because the coefficients found for the preceding frame are a close approximation.*

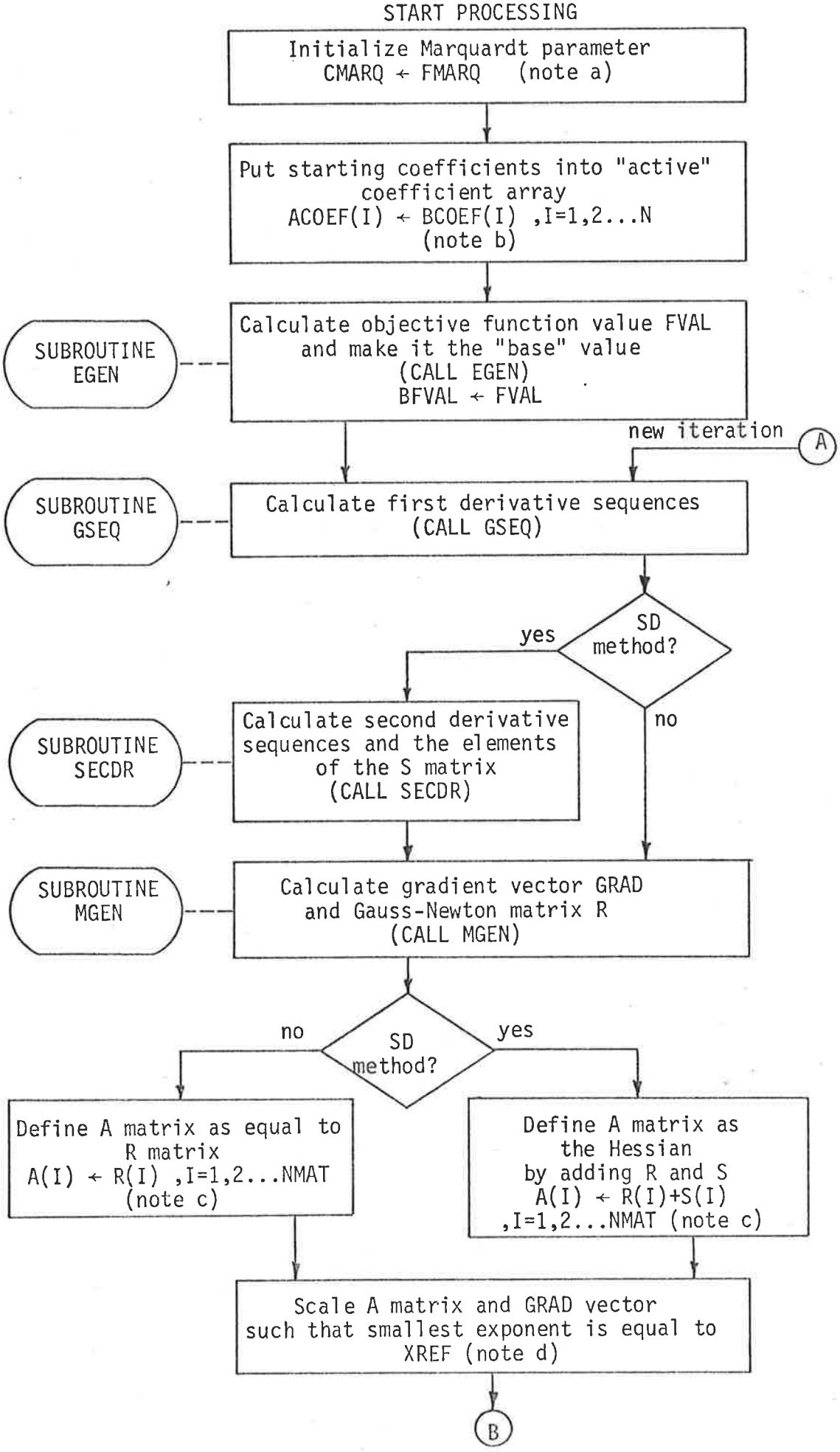


FIGURE 7.1 (sheet 1 of 3) ..Flow Chart - Optimization Phase..

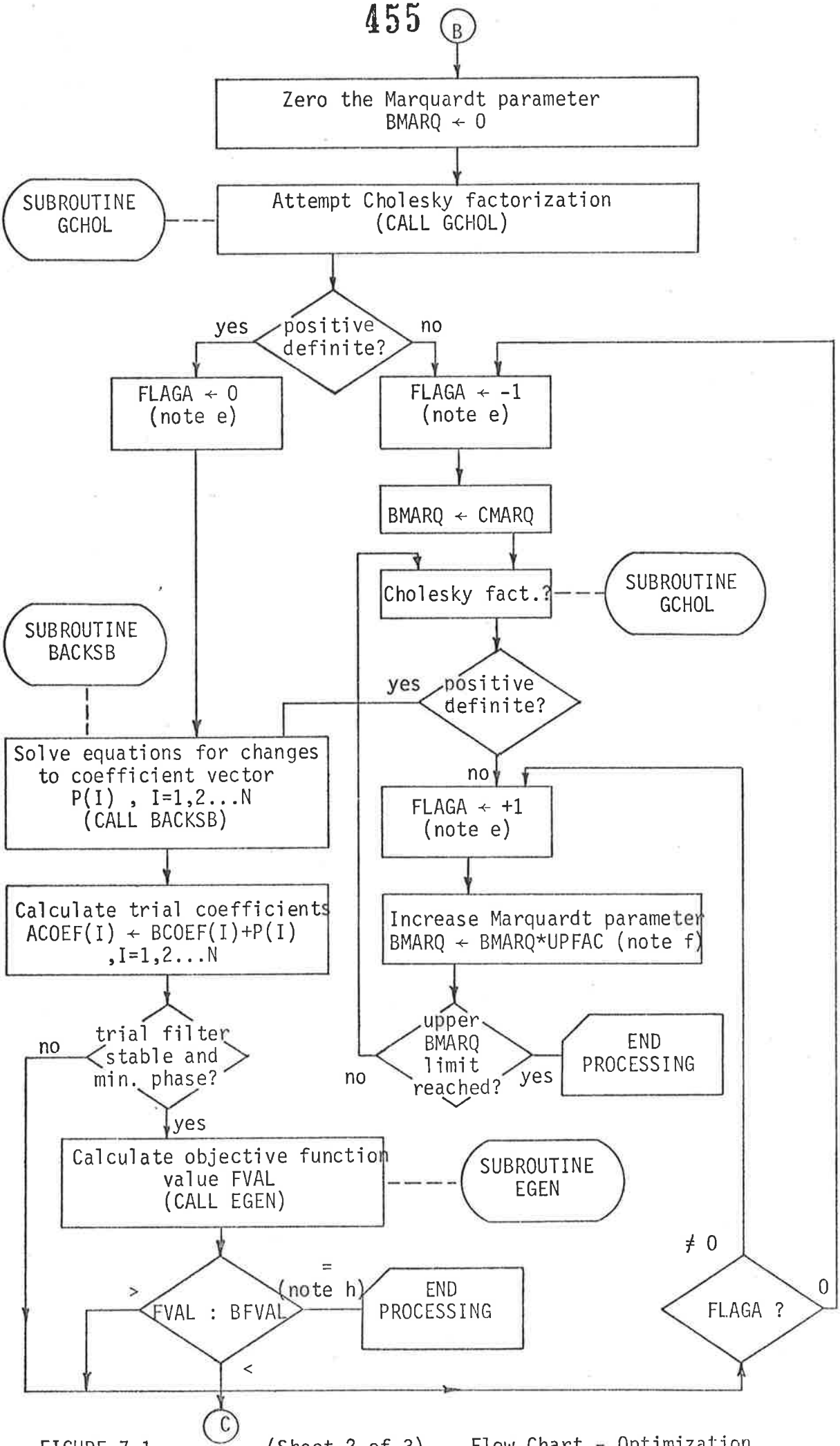


FIGURE 7.1 (Sheet 2 of 3) ..Flow Chart - Optimization..

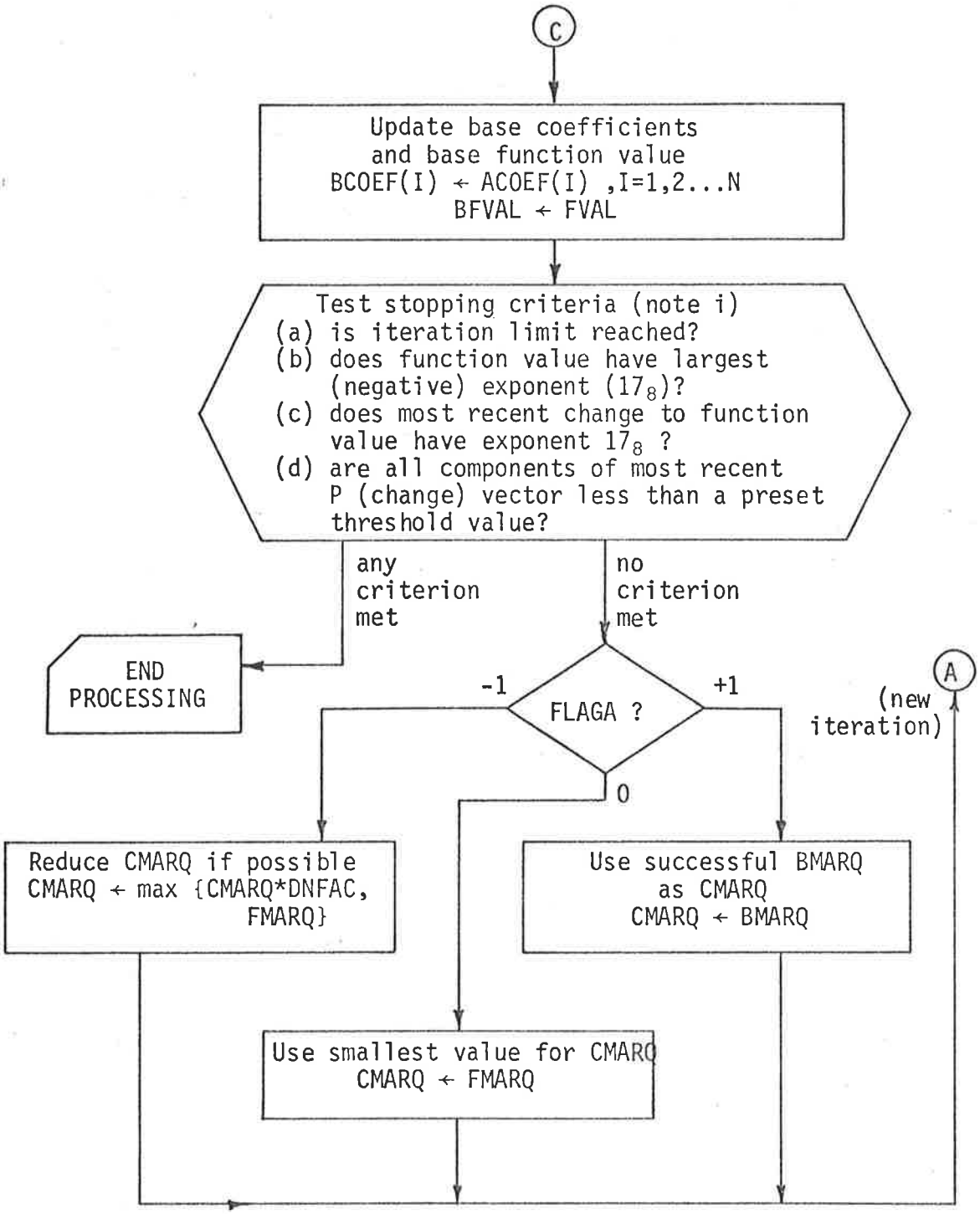


FIGURE 7.1 (Sheet 3 of 3)

Simplified Flow Chart of Main Program - Optimization Phase

Hence, in the tests to be reported, the coefficients from the preceding frame were used. The *first* frame of course requires special treatment, but in the tests it was not necessary to consider this because known correct coefficient values could be introduced to start the procedure (Section 7.7). Coefficients are transferred into an "active" array in GASP data memory (ACOEF) whence they are used by the speed-optimized digital filtering routine in EGEN.

c) The matrices involved are symmetric. Storage and computation time are saved by calculating only the upper triangle (including the main diagonal). The number of elements in each matrix (NMAT) is then $\frac{1}{2}N(N+1)$ where N is the number of variable coefficients.

d) Assuming that the Marquardt parameter is zero, the trial coefficient vector $\mathbf{x}^{(t)}$ is calculated from

$$\mathbf{x}^{(t)} = \mathbf{x}^{(b)} + \mathbf{p} \quad . \tag{7.7}$$

$\mathbf{x}^{(b)}$ is the current "base" vector (BCOEF) and the $\mathbf{p}$ vector is found from

$$\mathbf{A}\,\mathbf{p} = -\mathbf{g} \quad , \tag{7.8}$$

$\mathbf{g}$ being the gradient vector (called GRAD in figure 7.1) and $\mathbf{A}$ the iteration matrix (which is equal to $\mathbf{H}$ or $\mathbf{R}$). From (7.8), $\mathbf{p}$ would be unchanged by scaling all elements of $\mathbf{A}$ and $\mathbf{g}$ by the same amount. At the step labelled (d), the exponents of the $\mathbf{A}$ and $\mathbf{g}$ elements are all moved up or down by the same amount (so that the smallest exponent has some value XREF). This is done to minimize the chance

of failure in the Cholesky factorization routine GCHOL and the back-substitution routine BACKSB due to floating-point overflow.

e) A 3-value variable FLAGA is used to record whether a non-zero value of Marquardt parameter BMARQ was required on a given iteration, and, if so, whether the first value tried (CMARQ) was successful in reducing the objective function (block squared error). If BMARQ = 0 was effective, the first non-zero value to be tried next time, CMARQ, is reset to the smallest possible value, FMARQ. If BMARQ = CMARQ was effective, CMARQ is reduced by some suitable factor DNFAC (say 0.5) for next time. If BMARQ > CMARQ was required, CMARQ is set to BMARQ.

f) Whenever a value of BMARQ (i.e. β) is unsuccessful (due either to the failure of $\mathbf{A} + \beta \mathbf{I}$ to be positive definite or to the trial coefficients' yielding a higher error value), BMARQ is increased by some factor UPFAC (say 2.0). This process is continued if necessary until BMARQ reaches some preset limit, in which case convergence for the frame is assumed.

g) When a set of trial coefficients (ACOEF) have been calculated a test is performed on each pair to ensure that the filter is stable and minimum phase before attempting to calculate the objective function value.

h) If the function value does not change, convergence is assumed. The normal situation in which this exit is taken is that in which the components of $\mathbf{p}$ are too small to make

any difference when added to the base coefficients (which are represented as 16-bit fixed point numbers).

i) Convergence is deemed to have occurred if all the components of the **p** vector were smaller than a certain amount (e.g. that necessary to change the coefficients by ± 4 LSB), or if the change to the function value, or the function value itself, had the most negative possible exponent (since further iterations would probably be pointless because of floating-point underflow). Exit is also made if a preset iteration limit is reached.

7.5.3 Organization of Subroutines

(a) EGEN

The purposes of subroutine EGEN are to calculate the error sequence e_m, m = 1,2 . . . M, which is the term-by-term difference between the target and model impulse response sequences, and the objective function value FVAL, equal to the sum of the squares of the error samples. It is called at the "base" point and at every trial point during each iteration.

The target impulse response Y_m is already stored for the current frame. The first action in EGEN is to calculate the filter impulse response B_m using the currently active set of coefficients which have been placed in array ACOEF. For speed, this calculation is performed in fixed point (using a subroutine CASCA, which assumes the coefficients and the state variables to be placed in data memory in a certain way which takes maximum advantage of the autoincrementing data pointers). Scaling to make the impulse response comparable with Y_m is performed by using the same input impulse height IMPI as was used in the

calculation of Y_m (Section 7.4.4). Arithmetic overflow is possible in the digital filtering routine. If it occurs, an error exit is made. In practice, the value IMPI must be kept small enough that the error exit is never taken; this is determined experimentally.

For maximum accuracy, the objective function value is accumulated using block floating point. Using three autoincrementing data pointers, the corresponding values of B_m and Y_m are fetched, differenced, squared, normalized (for maximum precision) and stored back into data memory. As the squares are stored their exponents are scrutinized and the smallest one recorded (XSMALL). A second pass is then made through the stored values. After each is fetched, it is aligned to have exponent (XSMALL-3) by right shifting in the (32-bit) normalizer/shifter, and then routed to the ALU to be added to the current partial sum. The alignment to (XSMALL-3) ensures six leading guard digits and prevents ALU overflow (guaranteed if $M < 64$).

Such block floating point operations are common in GASP programs as already mentioned in Section 7.4.1. Several passes through the data are needed but the parallelism of data memory, program memory, and arithmetic unit operation renders the scheme time-efficient, especially if arrays are contiguous and data pointers do not require reloading. The provision of a 4-bit exponent with every word allows track to be kept of the true magnitude even though a number is stored in left-shifted (normalized) form to maintain precision. The floating-point provision is thus of considerable value even though there is no hardware for general floating point additions and subtractions.

(b) GSEQ

Subroutine GSEQ calculates a gradient sequence of length M for the "numerator" and the "denominator" of each second order section whose coefficients are variable. This is performed by filtering the stored primary sequence B_m with an additional recursive second-order section, as shown in figures 4.13 and 4.14. Accordingly the filter is required to be minimum phase (as well as stable). Since the gradient sequence with respect to a given b (or d) coefficient is merely a delayed version of the corresponding a (or c) sequence (Section 4.7.2), only one sequence needs to be calculated and stored for each pair of coefficients. GSEQ performs the calculation by an efficient fixed point filtering routine. As with EGEN, arithmetic overflow is possible and leads to an error return. However, this too may be avoided by suitably scaling the original impulse height IMPI.

(b) MGEN

Subroutine MGEN calculates the gradient vector GRAD and the Gauss-Newton matrix $\mathbf{R}$. The formulae for the j th gradient element g_j and the (j,k)th matrix element r_{jk} are

$$g_j = \sum_{m=1}^{M} e_m \frac{\partial Bm}{\partial x_j} \qquad (7.9)$$

and

$$r_{jk} = \sum_{m=1}^{M} \frac{\partial Bm}{\partial x_j} \frac{\partial Bm}{\partial x_k} \qquad (7.10)$$

where x_j and x_k represent the j th and k th variable coefficient respectively. (Strictly speaking, each sum should be multiplied by 2, but this factor cancels out when $\mathbf{p}$ is calculated using (7.8) and so there is no need to introduce it). The error

sequence e_m, m = 1, 2 . . . M (calculated in subroutine EGEN) and gradient sequences $\frac{\partial Bm}{\partial x_j}$, m = 1, 2 . . . M; j = 1, 2 . . . N (calculated in subroutine GSEQ) were developed by fixed-point filtering operations and are stored in data memory arrays with a common exponent. There is thus no need to pre-align individual operands when accumulating the sums in equations (7.9) and (7.10), and full advantage is taken of GASP's efficient design for such sum-of-products operations.

As mentioned in Section 7.5.2, only the upper triangle of the **R** matrix is calculated and stored and so the total number of gradient and matrix elements is $\frac{1}{2}N^2 + \frac{3}{2}N$. However, the gradient sequence $\frac{\partial Bm}{\partial b_k}$ for any section k differs from the sequence $\frac{\partial Bm}{\partial a_k}$ only by being delayed by one sample (with a zero element inserted as the first sample). Thus

$$\sum_{m=1}^{M} \frac{\partial Bm}{\partial b_j} \frac{\partial Bm}{\partial b_k} = \sum_{m=1}^{M-1} \frac{\partial Bm}{\partial a_j} \frac{\partial Bm}{\partial a_k} \qquad (7.11)$$

for any two sections j and k. There is no need to accumulate the sum $\sum_{m=1}^{M} \frac{\partial Bm}{\partial b_j} \frac{\partial Bm}{\partial b_k}$ separately provided that the penultimate partial sum as well as the final result is recorded when accumulating $\sum_{m=1}^{M} \frac{\partial Bm}{\partial a_j} \frac{\partial Bm}{\partial a_k}$. Similar remarks apply if a_j is replaced by c_j and/or b_j is replaced by d_j. The total number of sums-of-products which must be accumulated by MGEN (including those for gradient components) is then only $\frac{3N^2}{8} + \frac{5N}{4}$.

Gradient and matrix elements are normalized before storage to gain maximum accuracy in the solution of linear equations to follow.

(d) SECDR

Subroutine SECDR is called only if the second-derivative version of the algorithm is being used. Its function is to calculate the matrix **S** which must be added to the Gauss-Newton matrix **R** to obtain the Hessian matrix **H** . As derived in Section 4.4.3, the (j,k) th element of **S** is

$$S_{jk} = \sum_{m=1}^{M} e_m \frac{\partial^2 Bm}{\partial x_j \partial x_k} \qquad (7.12)$$

(where as in subroutine MGEN, a scale factor 2 is omitted for simplicity). The second-derivative sequences are developed by further fixed-point recursive filtering of the stored first-derivative sequences, as outlined in Section 4.7.2. The accumulation of the sums (7.12) is interspersed with the filtering operation to avoid having to store the $\frac{N^2}{8} + \frac{N}{4}$ distinct second-derivative sequences, which would (for realistic N and M) require more data memory than that available.

The elements of **S** are stored in normalized form. When **R** and **S** are later added (in the main program), individual pairs of elements are added using general floating point operations.

(e) GCHOL

Subroutine GCHOL implements the Cholesky factorization of the matrix $\mathbf{A} + \beta \mathbf{I}$ into the form $\mathbf{L}\ \mathbf{D}\ \mathbf{L}^T$ (where **L** is a unit lower triangular matrix and **D** is a diagonal matrix) as discussed in Section 3.4.7. Since **L** has unit diagonal elements they do not need to be stored; the remainder of its lower triangle ($\frac{1}{2}N(N-1)$ elements) is calculated column-by-column but stored

row-by-row. The row-wise storage allows the use of autoincrementing data pointers when "older" **L** elements must be re-fetched for use in calculating later elements (equations (3.40) and (3.41)). The diagonal elements of **D** are stored in another array. The occurrence of a negative element in **D** indicates that $\mathbf{A}+\beta\mathbf{I}$ is not positive definite and so the factorization cannot proceed. In practice, each element of **D** is tested immediately it is calculated to ensure that it is greater than a certain small positive quantity. If the test fails, GCHOL returns control immediately to the main program (at a different address from the normal return) to allow a jump to the sequence which increases β (BMARQ).

Floating-point operations are used throughout subroutine GCHOL to gain maximum accuracy. If exponent underflow is detected the number is forced to zero. GCHOL itself calls two other subroutines; NEGSUM to accumulate a floating-point sum of products and RECIP to calculate the reciprocals required by equation (3.41). GASP has no hardware divider; the use of the successive approximations register would allow a reciprocal to be found but it would need 17 multiplications and comparisons to obtain a result accurate to the last bit. Subroutine RECIP uses Newton's method and allows the reciprocal to be found (with the last two bits uncertain) in an average of six multiplications. The precise number of multiplications depends on the operand. The initial guess for Newton's method is taken from a 24-word lookup table and based on the 5 most significant bits of the operand.

(f) BACKSB

Subroutine BACKSB completes the solution of the linear equations to find the **p** vector (equation 7.8), using the Cholesky factors **L** and **D** as found by GCHOL. The actions required amount to two operations of back-substitution and a scaling by the elements of **D**. The necessity to *divide* by the diagonal elements of **D** is avoided by storing instead their reciprocals.

As in subroutine GCHOL, floating-point operations of full generality are used throughout for the sake of accuracy.

7.6 Test Results - Exact Target-Model Fit

In order to perform a simple test of the operation of the optimization part of the program, the homomorphic processing section was bypassed and a target impulse response produced using a cascade-form digital filter having four poles and two zeros. The parameters of this generating filter are given in table 7.1, columns A and B. The first 50 samples of the impulse response were used as the target. This is shown in figure 7.2. The model to be fitted had the same complexity as the generator, and starting values for the parameters as given in table 7.1, columns C and D.

Good convergence was expected with both the SD and GN methods because the residuals (and the objective function value) at the solution are zero. This is borne out by figure 7.3, which is a plot of objective function value (on a logarithmic scale) against iteration number. The process was terminated as soon as the function value required the largest possible GASP exponent (17_8) for its representation - equivalent to a log value of 3.01

(A)	(B)	(C)	(D)
Generating Filter		"Initial Try" Filter	
Coefficients	Root Locations	Coefficients	Root Locations
	Complex zeros		Complex zeros
$a_1 = -1.718445$	$r = 0.93$	$a_1 = -1.5$	$r = 0.866$
$b_1 = 0.864929$	$\phi = \pm 22.5°$	$b_1 = 0.75$	$\phi = \pm 30°$
	Complex poles		Complex poles
$c_1 = -1.272766$	$r = 0.90$	$c_1 = -1.0$	$r = 0.707$
$d_1 = 0.809998$	$\phi = \pm 45°$	$d_1 = 0.5$	$\phi = \pm 45°$
	Real poles		Real poles
$c_2 = -0.099976$	$p_1 = 0.8$	$c_2 = 0.0$	$p_1 = 0.707$
$d_2 = -0.559998$	$p_2 = -0.7$	$d_2 = -0.5$	$p_2 = -0.707$

TABLE 7.1 "Generator" and "Initial Try" Coefficients and Root Locations

– Test of Section 7.6

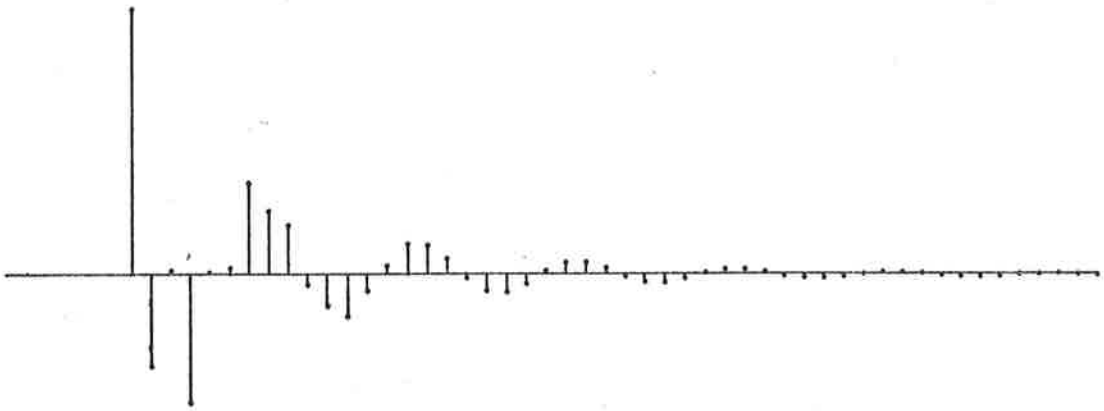


FIGURE 7.2 Target Impulse Response - Test of Section 7.6

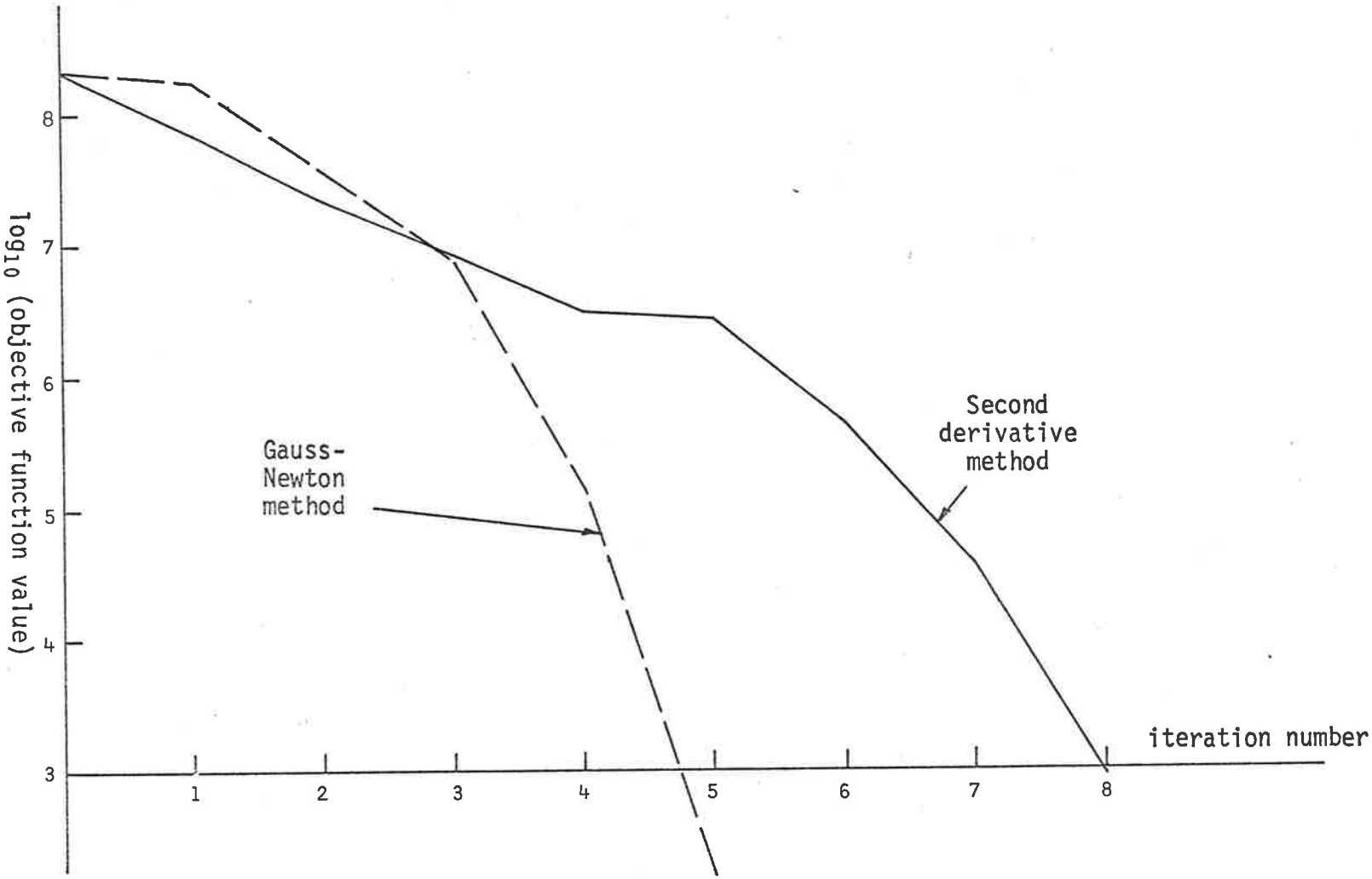


FIGURE 7.3 Objective Function Value - Test of Section 7.6

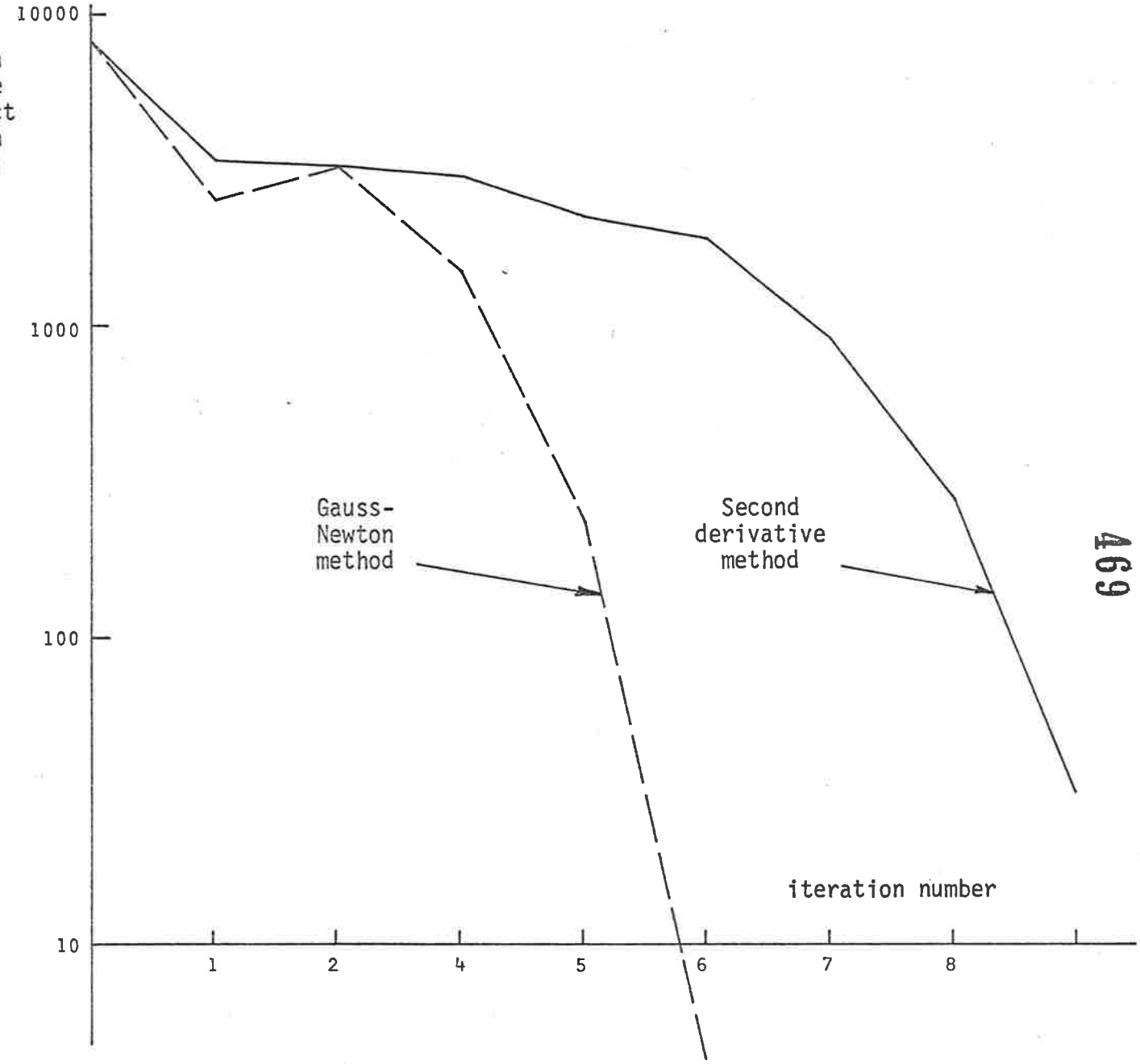


FIGURE 7.4

Progress Towards Solution Vector,

Test of Section 7.6

on the scale of figure 7.3. Five iterations were needed with the GN method and eight with the SD method. "Newton-like" convergence (that is, success of the trial parameter vector obtained using a zero Marquardt parameter) was evident on the last three iterations (with GN) and the last four (with SD).

The Euclidean distance from the correct solution (in the six-dimensional parameter space) was also evaluated after each iteration. This is shown (also on a logarithmic scale) in figure 7.4. The distance is quoted in terms of the least-significant-bit in the fixed-point representation used for the coefficients; for example, an ordinate of 10 in figure 7.4 corresponds to a distance (square root of the sum of squares of component distances) of 10 LSB, or (as a decimal number) 0.000610.

7.7 Test Results - Time-varying Synthetic Signal

This section reports the results of a more complicated program test using a synthetic, quasi-periodic, "speech-like" signal. The situation is still quite artificial because the signal is generated using a digital filter of the same complexity as the model to be fitted, and is considerably simpler than real-world signals (such as speech). Nevertheless, there are some "speech-like" features in this test, which demonstrates the validity of the homomorphic processing phase of the program as well as the ability of the Marquardt algorithm to track changes in signal parameters with time.

The test signal usedwas 10,000 samples in length, and was generated by driving a cascade-form digital filter having six poles and two zeros with a periodic train of impulses. An

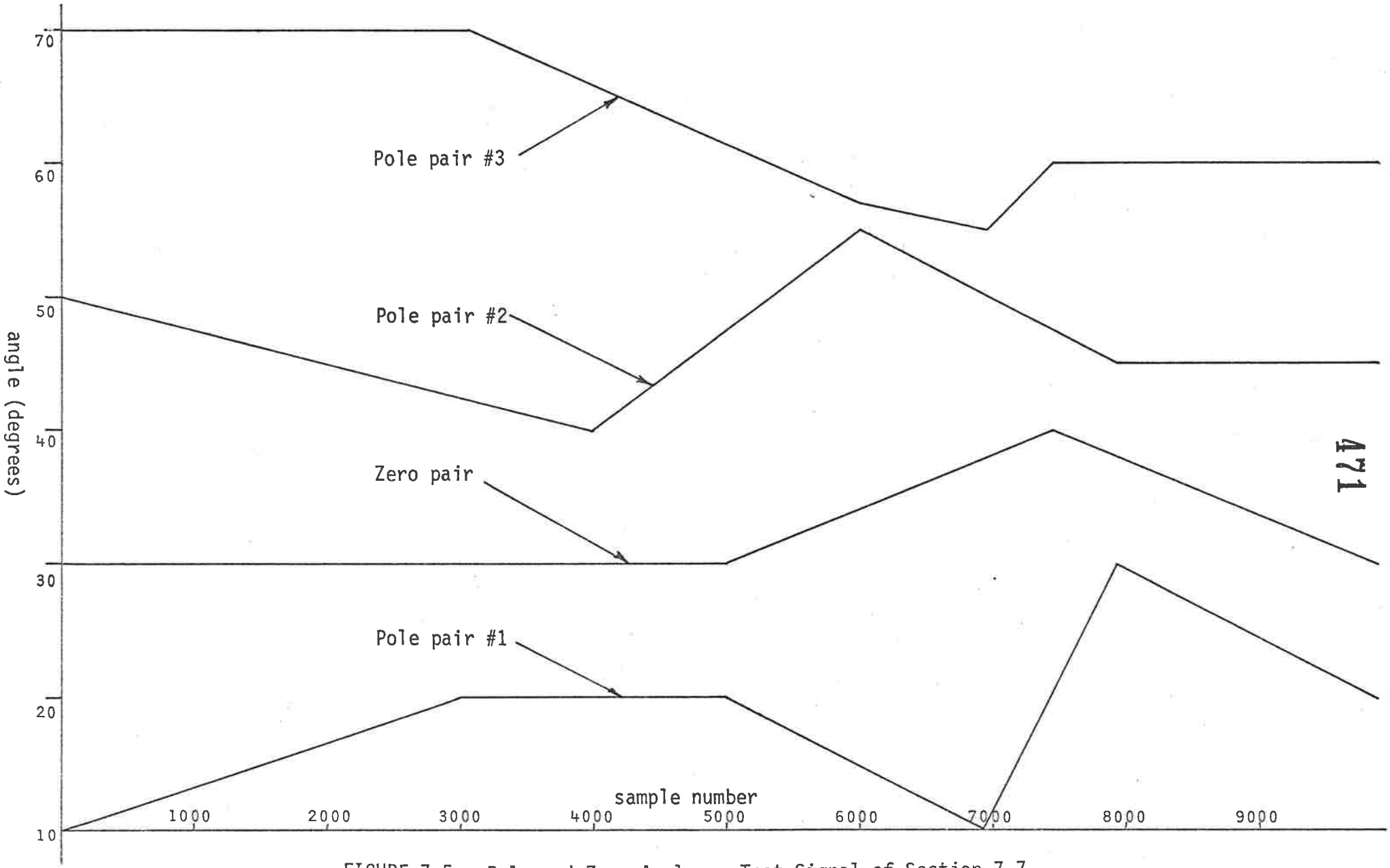


FIGURE 7.5 Pole and Zero Angles - Test Signal of Section 7.7

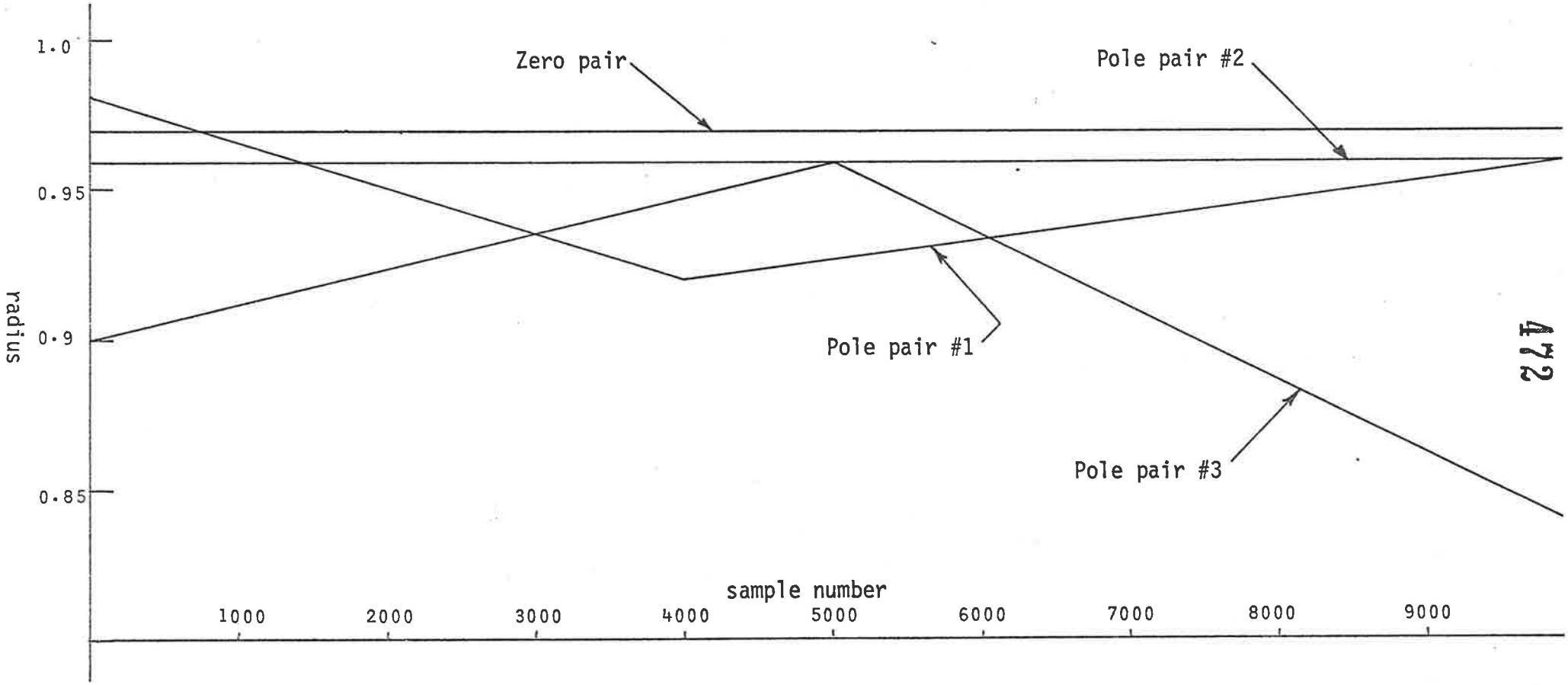


FIGURE 7.6 Pole and Zero Radii - Test Signal of Section 7.7

impulse was injected every 100 samples. All poles and zeros remained as complex conjugate pairs, but their radii and angles were varied in the piecewise-linear fashion shown in figures 7.5 and 7.6. Parameters were updated at every sample.

The signal was analysed in 39 contiguous frames of 256 samples each (the final 16 samples of the signal were ignored). For each frame the 256 signal samples were Hamming windowed and then 256 zero-valued samples were appended, and the cepstrum of the resulting 512-point real-valued sequence was computed. This was truncated to the 48 low-quefrency values, from which a 125-sample target impulse response was computed using the recurrence relation (Section 7.4.4). The true Marquardt optimization procedure was then employed, using the parameters found for the last frame as initial trial values. The values which were correct at the first signal sample were fed in as the initial try for frame 1.

Parameter values found for each frame are shown in figures 7.7 and 7.8, in which the "true" parameter trajectories are repeated for comparison. The resulting spot values are positioned on the time axis at the centre of the relevant frame. In no frame did there appear to be any convergence problem due to large residuals, and the Gauss-Newton method produced results identical to the second-derivative method (to within the plotting accuracy). Generally, the GN method required fewer iterations and less time than the SD method, as is apparent from table 7.2, which lists details for the first ten frames. Average numbers of iterations over all 39 frames were 7.05 (SD) and 6.05 (GN). Times quoted in

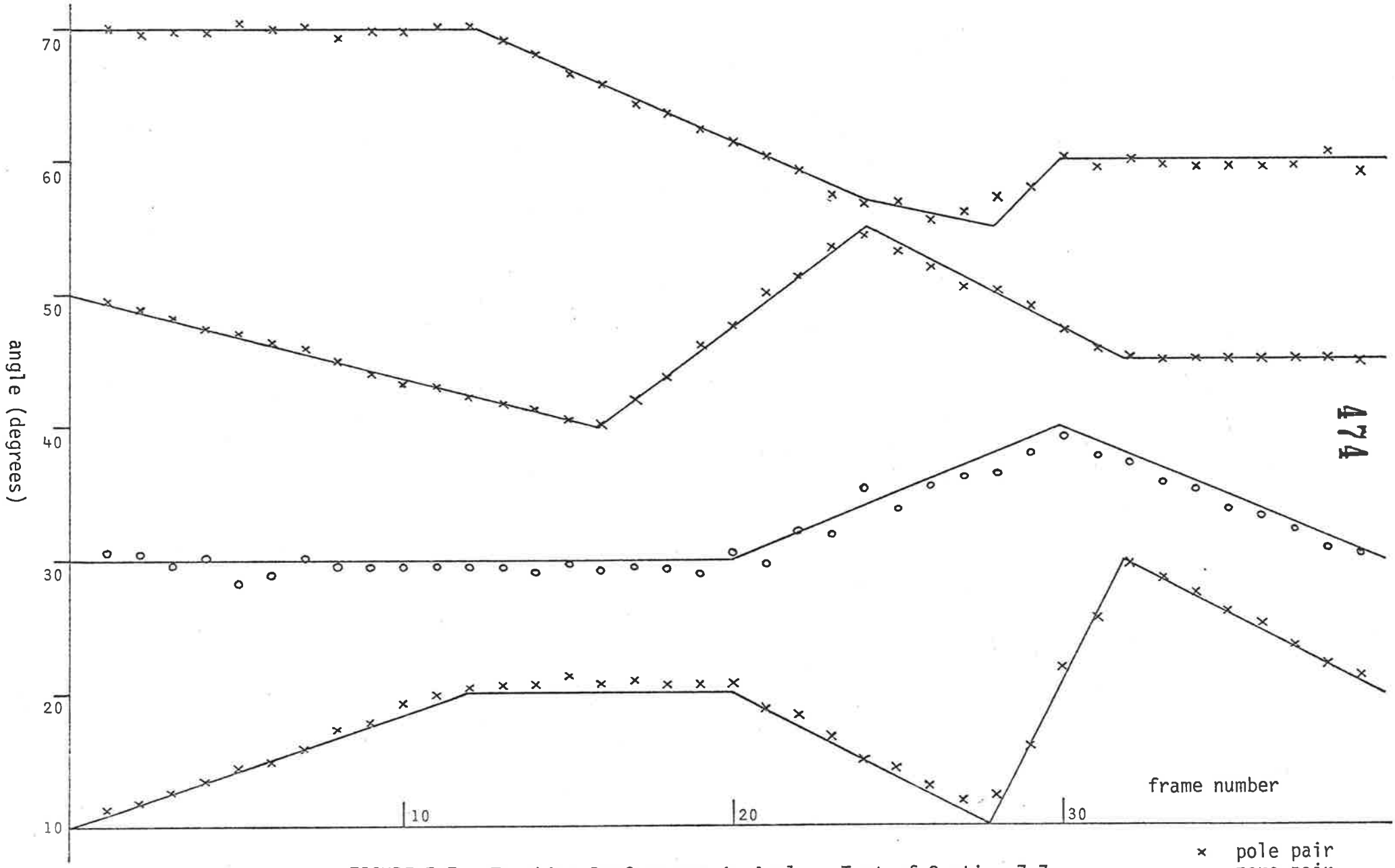
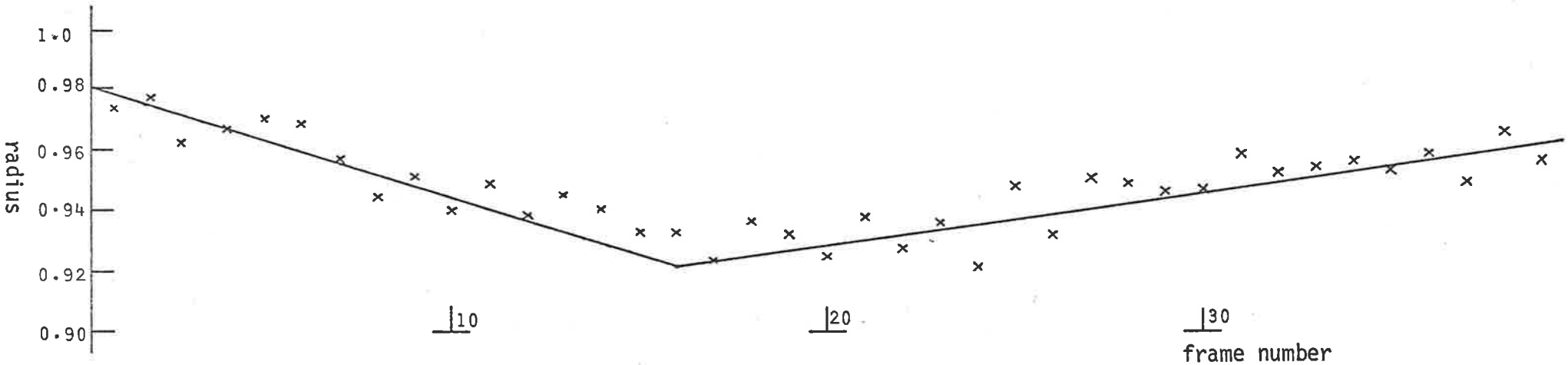
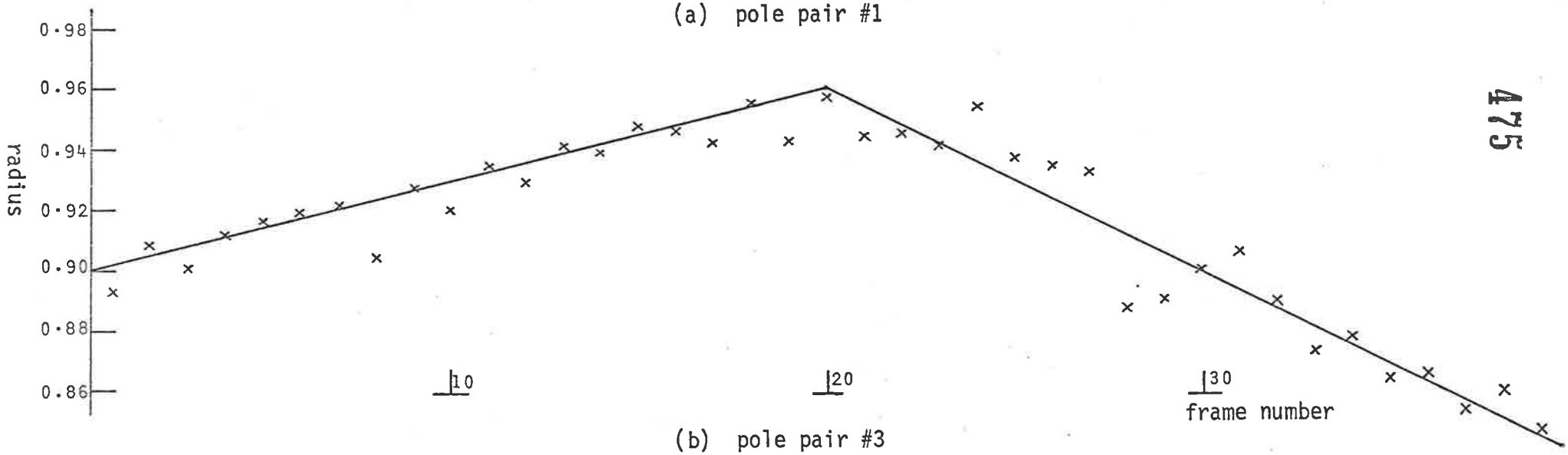


FIGURE 7.7 Tracking Performance in Angle - Test of Section 7.7



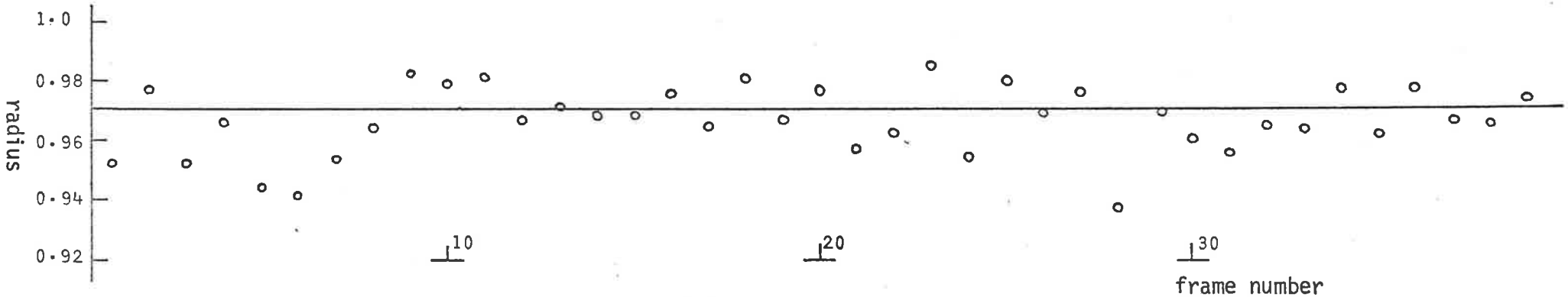
(a) pole pair #1



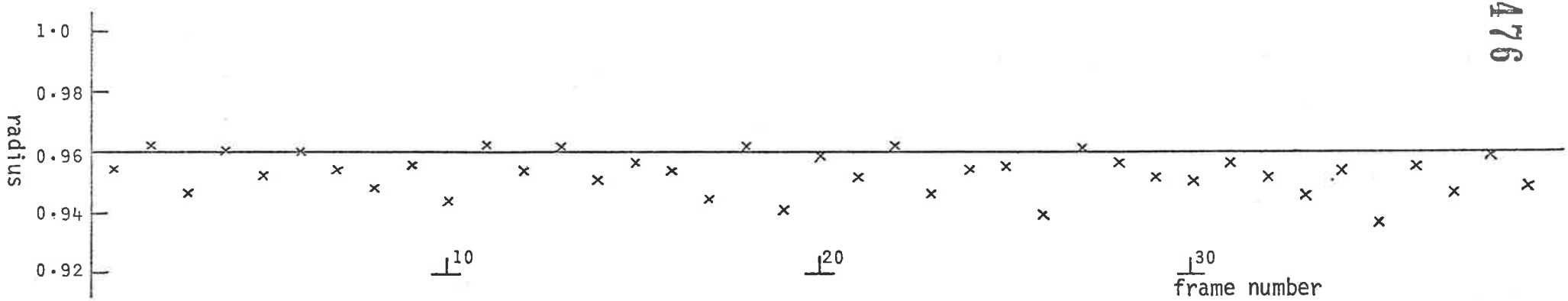
(b) pole pair #3

FIGURE 7.8 Tracking Performance in Radius - Test of Section 7.7

(continued next page)



(c) zero pair



(d) pole pair #2

FIGURE 7.8 (continued) Tracking Performance in Radius - Test of Section 7.7

Frame No.	No of Iterations	Total time (ms)
1	7	158
2	5	166
3	5	134
4	5	156
5	5	137
6	8	169
7	10	223
8	10	245
9	5	145
10	10	256
Averages	7.0	179

(a) Second-derivative method

Frame No.	No of Iterations	Total time (ms)
1	8	172
2	3	120
3	4	116
4	4	130
5	7	151
6	4	142
7	8	142
8	9	178
9	5	123
10	4	119
Averages	5.6	139

(b) Gauss-Newton method

TABLE 7.2 Numbers of Iterations and Total Frame Execution times - Synthetic Signal (First 10 frames only)

table 7.2 include the time taken to transfer the data from NOVA memory as well as to do the homomorphic processing and the optimization. Table 7.3 gives a breakdown of the time taken by various parts of the program. All measured values are averaged over several frames, but in most cases the frame-to-frame variability was only in the third decimal place.

The operation of the homomorphic processing phase of the program is illustrated by example in figure 7.9 which shows waveforms and spectra for a representative frame (frame 2). In (c), the "pitch peak" at sample number 100 (which is due to the periodic excitation) is clearly visible. Features at higher quefrency indices are due to the aliasing which is unavoidable when the DFT is used to compute the cepstrum (Oppenheim and Schafer, 1975, pp. 480-531). Figure 7.9 (d) and (e) show the low-quefrency parts of the complex cepstrum (x_m) and the "associated cepstrum function" γ_m (mx_m) respectively. The zero-quefrency sample (which determines only the overall signal level) is omitted. Figure 7.9 (f) shows the minimum-phase impulse response estimate, derived by recurrence from samples 1-48 of γ_m. Finally, figure 7.9 (g) and (h) show the signal log spectrum (with peaks at the pitch harmonics) and the log spectrum derived by re-transforming the cepstrum of figure 7.9 (d). The smoothing expected from homomorphic processing is well illustrated, and the frequencies of the three poles and the zero are quite obvious.

Program Segment	Time (ms)	Number of frames used in averaging
Get data from NOVA, de-mean, window	1.835	3
Calculate cepstrum	44.90	6
Calculate target impulse response	4.035	5
Calculate error sequence and objective function value	1.721	6
Calculate gradient sequences, gradient vector and Gauss-Newton matrix	3.299	10
Calculate gradient sequences, gradient vector and Hessian matrix	7.454	10
Solution of linear equations (complete Cholesky factorization and back-substitution)	1.368	6 *

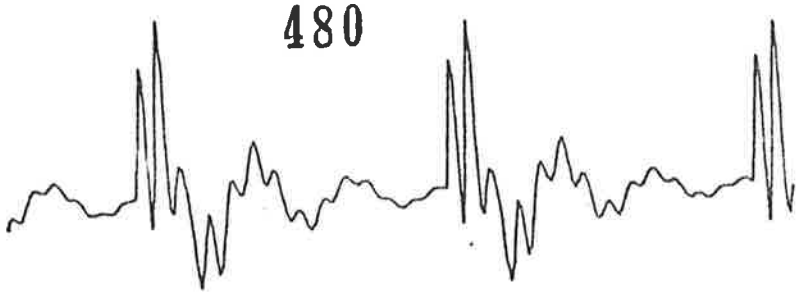
* 6 different values of BMARQ, all in frame #1

TABLE 7.3 Timing Tests - Homomorphic and Optimization Program using synthetic signal

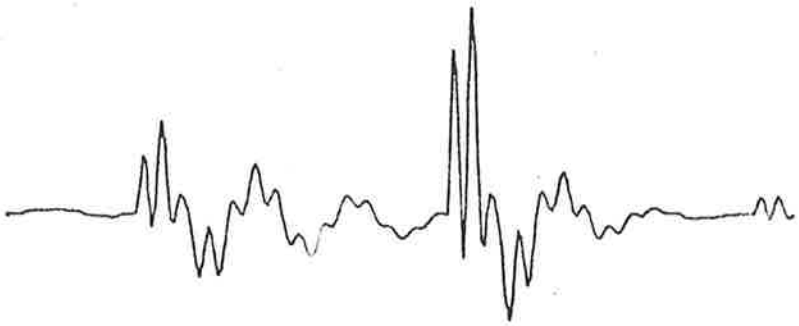
Length of cepstrum window = 48

Impulse response duration considered (M) = 125

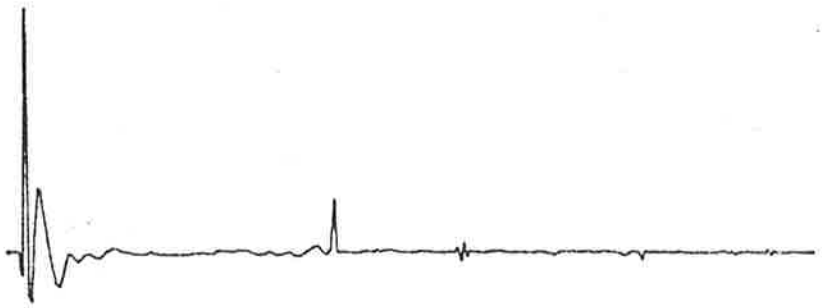
Number of parameters (N) = 8.



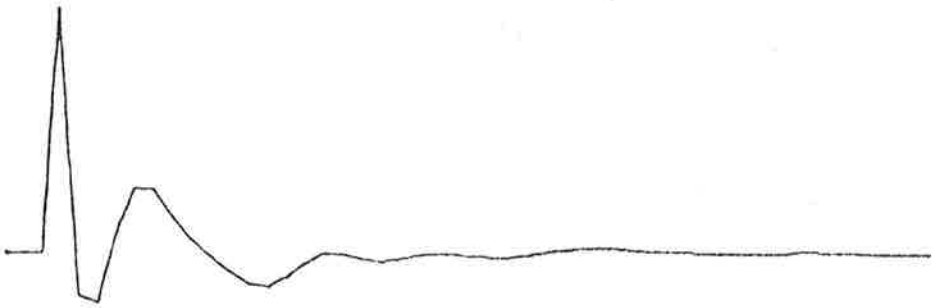
(a) data segment



(b) windowed data segment



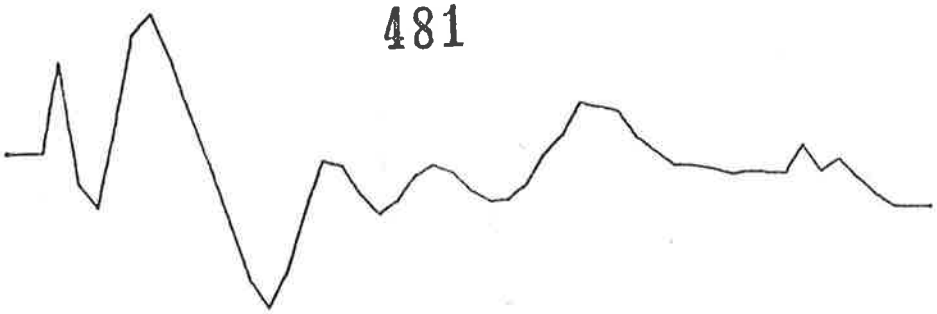
(c) complex cepstrum x_m, m = 0,1, ... 256.



(d) low-quefrency part of complex cepstrum
(samples 1 to 48)

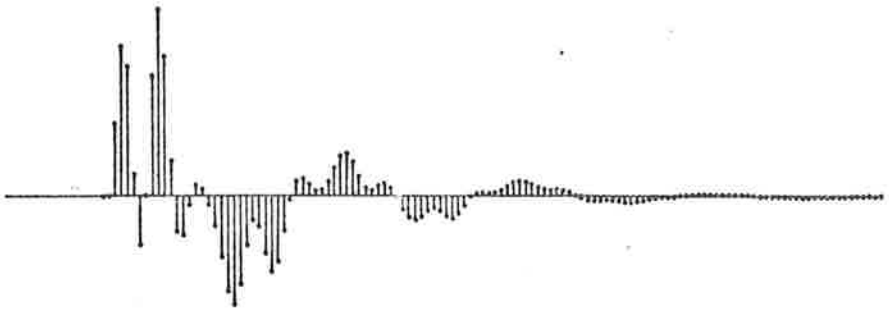
FIGURE 7.9 Representative Analysis for Synthetic Signal

(Frame 2) (continued on next page)



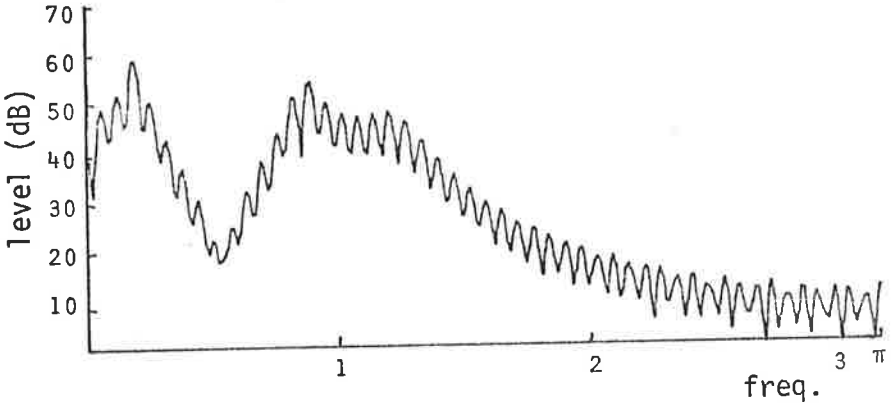
(e) Associated cepstrum function γ_m

(= m × x_m) (samples 1 to 48)

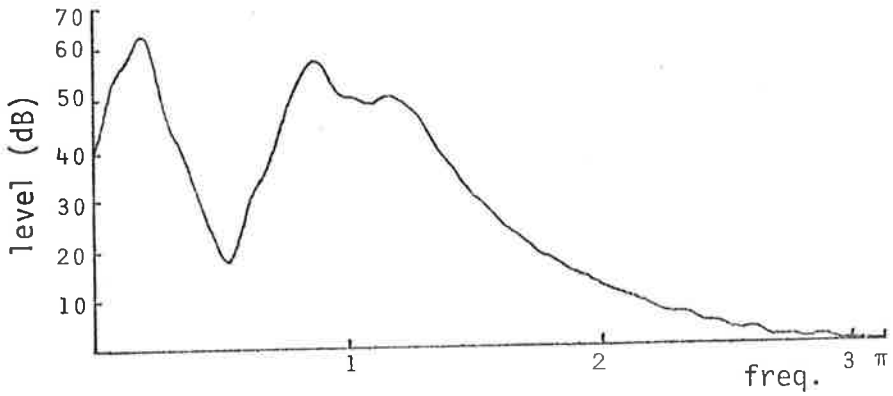


(f) Minimum-phase Impulse Response Estimate

(125 samples)



(g) Log Spectrum of Windowed Date



(h) Log Spectrum, Derived from Cepstrum Samples 1 to 48

FIGURE 7.9 (continued) Representative Analysis for Synthetic Signal (Frame 2)

7.8 Discussion

7.8.1 Resumé of Preceding Sections

A considerable amount of effort has been put into (a) the development of fast signal processing hardware (the GASP processor), and (b) the design of software to utilize the machine for identification of the parameters of speech-like signals. The signal is partitioned into contiguous frames of 256 samples. For each frame the processing is organized into two distinct phases:
a) the effect of the quasi-periodic excitation is removed by homomorphic processing. The output of this phase is the impulse response sequence of a minimum-phase system having a smooth approximation to the magnitude spectrum of the signal. (In a real application the value of the "pitch" would also be of interest; this is derived from the location of the high-quefrency peak in the cepstrum.)

b) a parametric model of the signal is then developed by iteratively adjusting the parameters for a best least-squares match of the impulse response of the model and the "target" response resulting from phase (a). That is, optimization is performed in the time domain, and the process is an example of "formulation B1" of chapter four. Two optimization methods are selectable - true Marquardt SD and GN methods.

The algorithm has been successfully applied to a synthetic test signal with time-varying parameters (Section 7.7). This test has demonstrated the algorithm's ability to track quite rapid changes in signal parameters (e.g. the rapid variation in the frequency of pole pair $\neq 1$ between frames 28 and 32, figure 7.7). It was also able to handle satisfactorily the close

approach of two poles in frequency (poles numbered #2 and #3 in figure 7.7, at frames 24-26).

7.8.2 Limitations and Deficiencies

The foregoing sections of this chapter have demonstrated by example some of the considerations relevant to the application of optimization techniques to system identification using short wordlength digital hardware. It must be emphasized, however, that the test results pertain to a highly artificial situation. Much additional program development would be necessary to obtain satisfactory operation with substantially more general (and realistic) signals.

The hardware, the program and the tests of Sections 7.7 are deficient or insufficiently general in several ways. Some of these are quite minor but others are thought to be far-reaching and may severely limit (or at best, greatly complicate) the application of the system modelling techniques considered. These matters have not been studied in sufficient detail that they may be discussed at length here. The matters of execution speed and the suitability of the GASP hardware for its task are treated in Sections 7.8.3 and 7.9. Apart from this, the following observations are made:

(a) Nature of the Signal

In generating the test signal of Section 7.7, a truly periodic excitation was used, and all poles and zeros remained complex and minimum phase at all times. However, none of these features is thought to be seriously limiting. With regard to

the nature of the excitation, the homomorphic processing technique has been shown to be successful in estimating a system impulse response when the excitation is only approximately periodic, and indeed when it is noise-like, as in unvoiced speech (Rabiner and Schafer, 1978, p. 370).

The method of estimating the impulse response from the cepstrum (Section 7.4.4) guarantees a minimum-phase model regardless of the signal itself. The present optimization algorithm (whose gradient computation scheme fits it only for minimum-phase systems) is thus adequate (theoretically) even when signal zeros occur outside the z-plane unit circle. There is reason to believe, however, that some difficulty may be experienced if zeros occur *close* to the unit circle (see (c) below).

The occurrence of roots on the real axis was avoided in the test signal, for simplicity. However, the implementation of the root pairing idea of Section 4.9.2 should allow real roots to be handled without serious difficulty beyond the intricacies of the programming.

(b) Model Form and Optimization Starting Point

Probably the most serious impediments to successful automatic pole-zero modelling of real-world signals are related to the selection of a form for the model and to the determination of parameter values to start the optimization process.

If it is assumed that a rational z-transform model is adequate the question of form becomes one of how many poles and how many zeros to include. The study of example F in chapter

five illustrates some difficulties which can arise. For example, figure 5.19 (page 332)illustrates the response of a model having a lightly-damped pole on the negative real axis. This pole is obviously not characteristic of the actual signal; it occurs in the least-squares "solution" because the assumed number of poles and zeros was a poor choice for modelling that signal. In addition, models were postulated for which *no* minimizing parameter vector meeting minimum-phase constraints could be found at all, even though the target impulse response was itself minimum phase. Similar difficulties have been encountered when attempting to model real speech using the GASP program.

The above considerations imply that some automatic means for selecting the complexity of the model must be devised. Since this "complexity" may change from frame to frame the technique of beginning the optimization process from the "solution" for the preceding frame may not always be adequate. An automatic means for starting-point selection is also required (as indeed it would be for the *first* frame in any case).

The two matters are likely to be intimately related. It is felt intuitively that frequency-domain analysis offers the most promise - some form of peak and trough picking in the cepstrally-smoothed log spectrum would be involved. A further technique (Christensen *et al.*, 1976) based on peak-picking in the second-difference function of this spectrum may also be considered. Time domain regression methods (those of Shanks (1967), Kalman (1958) and the Padé approximant technique of Brophy and Salazar (1973)) also enable parameter estimation once the form of the model has been decided, but have the serious

complication that a polynomial rooting routine is required to translate the model to the cascade form. This may be especially difficult with a short wordlength implementation.

(c) Generality of Gradient Computations

The "short" method of gradient sequence calculation proposed in Section 4.7.2 has been incorporated into the GASP optimization program. This computational scheme does not permit zeros to wander outside the unit circle during optimization, but it was thought initially to be adequate because the target impulse response is guaranteed to be minimum phase (Section 7.4.4). However, in attempts to model real speech signals failure of the optimization algorithm was often observed because the constraint on zero positions was encountered. Whether this mode of failure can always be avoided by an appropriate choice for the degree of the pole-zero model is not clear.

However, if the "general" method for calculation of gradient sequences were to prove necessary, the evaluation of *second* derivatives would become most unwieldy, and a first-derivative quasi-Newton optimization method would be more appropriate than the Marquardt methods.

7.8.3 Speed

As apparent from table 7.2, frame processing times of 100-200 ms were observed with 6 poles, 2 zeros, and a target impulse response duration of 125 samples. It is obvious that real-time processing power for speech has not been attained, especially as (a) a 6-pole 2-zero model is certainly too simple for real speech, and (b) the resolution of the difficulties

discussed in Section 7.8.2, if possible at all, will result in a further time penalty. For a reasonably good parametric representation of speech, it is generally accepted that a new parameter set is required every 10-20 ms (Rabiner and Schafer, 1978), so that the present GASP implementation falls short of real time by a factor of at least 10. However, because advantage is taken of parallelism in the operation of program memory, data memory, and the various elements of the arithmetic section, because of the incorporation of a hardware sum-of-products unit and a hardware normalizer/shifter, and because of the use of a judicious mix of fixed-point, block floating-point and pure floating-point computation, the implementation is still considerably more efficient than the equivalent operation on a conventional minicomputer. Some real-time system identification problems which are less time-critical than speech analysis may thus be facilitated by such a system.

The GASP program is probably close to being as efficient as possible given the basic algorithm and the hardware available (except that if attention were given to the scaling problem, a fixed point FFT could be programmed which would be satisfactory, and faster). The most obvious way of improving speed would be to incorporate more hardware parallelism. In this regard, several additions could be made to a future version of GASP, without significantly changing the concept. This is discussed further in Section 7.9. Apart from this, several other more algorithm-specific observations may be made:

a) The "deconvolution phase" and the "optimization phase" are independent of each other, and could run in parallel on two

different processors (the optimization being one frame behind). Alternatively, the FFT's could be greatly speeded up by using a pipelined FFT processor (Gold and Bially, 1973).

b) The "optimization phase" could usefully exploit a bank of hardware digital filters which would run in parallel to compute the gradient sequences.

c) The calculation of the error sequence and objective function value may be performed several times within each iteration. A very fast hardware digital filter having a multiplier dedicated to each coefficient would be of significant value in speeding up this step, which constitutes a large proportion of the total computation.

d) If a second-derivative method is retained, the necessity to solve linear equations should be reduced to once per iteration, by using a line search method in place of true Marquardt.

7.9 Suggested Improvements to GASP

The main architectural features of the GASP processor have been outlined in the paper reproduced in Section 7.2.1. The processor is a conventional stored-program digital computer in the sense that a single stream of instructions from memory is executed in a serial manner. Program looping and branching is allowed by the incorporation of fairly conventional conditional jump instructions. However, GASP differs from conventional

computers in the complete absence of any *arithmetic* instructions as such. In their place is the MOV instruction, a flexible data routing instruction which causes the simultaneous transfer of two 20-bit words from any of a variety of sources to any of a variety of destinations. "Arithmetic" is performed by virtue of the fact that some of the "destinations" are latches which feed operands to arithmetic units, while some of the "sources" are the outputs of those units. The arithmetic units themselves are either combinational logic interconnections requiring no sequencing for their internal operation (e.g. the ALU), or are clock-controlled sub-systems whose operation is initiated automatically by operand latching (e.g. the normalizer/shifter). In either case, some parallelism of data processing is achieved because the *program* may proceed immediately to the next instruction - no waiting is necessary unless a MOV instruction is calling for a result from a unit whose operation is not finished. The normal rate for processing of successive MOV instructions (with no waiting) is one per 90 ns.

The data memory, too, has its own internal controller, and its operations run in parallel both with program activity and with each of the arithmetic units. Data memory input and output ports are dedicated hardware registers; simply ordinary "sources" and "destinations" which may be addressed normally by the MOV instruction. For example, a word may be MOV'd at any time to an input port provided that the data memory controller has handled the storage of the last such word.

Orderly operation of the whole system is ensured by providing a "busy" flag for each source and each data memory

input port; if the relevant flag is set a MOV instruction waits. By exercising care in the sequencing of instructions, the programmer can usually keep such dead time to a minimum. *Two* data busses (and the simultaneous routing capability of the MOV instruction) are incorporated for several reasons:

a) Both 16-bit operands for a multiplication may be routed (usually from data memory) at the same time. This is important in maximizing throughput in the very important sum-of-products operation. (The multiplication rate achieved is slightly better than 4 per μs.)

b) 32-bit (double precision) numbers may have both halves MOV'd simultaneously, making double precision arithmetic virtually as fast as single precision.

c) With care in program design, two independent 16 or 20 bit numbers may often be routed simultaneously, saving instructions (and so, time).

The optimization study reported in this chapter has confirmed the usefulness of all the features mentioned above, as well as of the fast sum-of-products unit and the limited hardware floating-point capability. However, in several ways the arithmetic section of the machine is not as powerful or as flexible as it could be. Much improvement would result from providing more "arithmetic" hardware (together with more "sources" and "destinations" for the MOV instruction). Such modifications would involve no change to the concept of the machine, although it would be necessary to increase the wordlength from 20 to 24 bits to allow the MOV instruction to address 16, rather than 8,

sources and destinations for each bus, and so the suggestions are not relevant to the presently built version of the machine. The extra bits could also be used to some advantage in some of the non-MOV instructions. For data representation the exponent could perhaps be increased to 8 bits and the radix dropped to 2. No speed penalty would be incurred by incorporating additional arithmetic hardware; all units would continue to operate asynchronously and in parallel. Some examples of additions thought to be useful are given below.

a) General Purpose Registers

At the core of many signal processing operations is a short loop during whose execution some small number of intermediate results must be stored. To use data memory for this purpose may be inefficient because data pointer load and select instructions are required. In many practical cases the normalizer/shifter (N/S) has been used as a temporary register; it may be set in a "shift" mode but with a zero shift count. Alternatively, the ALU may be arranged to propagate its input I latch contents (say) direct to its output by setting ADD mode and clearing the other (J) operand. However, such ploys are artificial, they increase program complexity, and may not be possible at all if the ALU and N/S are in use as such. The inclusion of a few (perhaps 2 or 4) general registers would be simple and very useful. However, if made necessary by addressing constraints, these registers could conceivably double as the input latches for other additional units.

b) Second ALU

The existing 32-bit ALU has four modes (ADD, SUBTRACT, OR and COMPARE). In some routines, repeated switching between modes has been necessary, an inefficiency which could be avoided by having two ALU's, with one set to each mode. In addition, two ALU's operating in parallel would greatly simplify certain routines which previously required temporary storage of results just to free the ALU. Loops requiring both data memory address calculations and data arithmetic provide an obvious example.

c) Complex Multiplier

Provision of a unit for the fast evaluation of the product of two complex numbers would greatly improve GASP's performance in the Fast Fourier Transform. This would involve four 16-bit by 16-bit multipliers operating in parallel. When GASP was built such a unit would have accounted for a large proportion of the total cost and complexity (the existing GASP multiplier uses 32 4-bit by 2-bit multiplier devices), but 16-bit by 16-bit devices have since appeared, at reasonable prices.

d) Second Normalizer/Shifter (N/S)

The use of the N/S for block floating-point operations has been mentioned in Sections 7.4.1 and 7.5.3. In particular, alignment of all elements of a data memory array to a common exponent is often necessary. The inverse operation is the normalization of 32-bit words prior to storing the more-significant half in data memory.

These operations are efficient in the sense that program and data memory operations run in parallel with the shifting. However, if program loops are simple and shifts of more than a few places are required, the N/S performance limits the achieved speed. Alternate routing to *two* normalizer/shifters could significantly speed this operation.

A second N/S would also accelerate (and simplify) program loops which require the use of (say) both alignment *and* normalization.

Another deficiency of GASP is lack of flexibility in data memory access. The autoincrementing data pointers are extremely valuable, because arrays can often be stored in the order in which they are needed. However, this is not always the case; a bit-reversed pointer would be useful for the FFT, and autodecrementing pointers could be useful in a great many cases. For some purposes it would be advantageous to access alternate words, or perhaps every fourth word of an array. Considerable generality would be gained by using fast adders rather than counters for the data pointers. The increment at each access could then be any arbitrary (register-held) amount.

Finally, although the data memory is fully exercised during many "tight" program loops, there is much time during execution of the average program when it is idle. This would provide an opportunity for the host computer to send or receive data on a "direct memory access" (DMA) basis, instead of such transfers having to be (GASP) program controlled. Because GASP program and data memories are entirely separate, no slowing

down of GASP by "cycle stealing" is necessary; interprocessor communications would run in a low-priority background mode and the completion of a block transfer would be signalled by a flag accessible to GASP's conditional jump logic.

APPENDIX A

GENERAL MATHEMATICAL TERMINOLOGY

1. The concepts of *vector* and *matrix* are employed in this thesis in the usual sense (of linear algebra). A *vector* $\mathbf{x}$ is an ordered N-tuple of real numbers $(x_1, x_2, \ldots x_N)$ where the x_i $(i = 1, 2, \ldots N)$ are referred to as the *components*. The components are considered to be arranged as a *column* so that the usual rules of matrix multiplication may be applied. A square matrix $\mathbf{A}$ is a two-dimensional (say N by N) array of real numbers, and its general component (jth row, kth column) is denoted a_{jk}.

2. A *linear combination* of a set of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_M$ is any vector $\mathbf{y}$ which may be expressed as

$$\mathbf{y} = c_1 \mathbf{x}_1 + c_2 \mathbf{x}_2 + \ldots + c_M \mathbf{x}_M$$

where the $c_1, c_2, \ldots c_M$ are real numbers (*scalars*). (The operations of multiplication by the scalars and addition are performed on *each component*). Such a set of vectors is *linearly independent* if the only linear combination of them which is equal to the zero vector is that for which all the scalars $c_1, c_2, \ldots c_M$ are zero. A linear combination of two vectors

$$\mathbf{y} = c_1 \mathbf{x}_1 + c_2 \mathbf{x}_2$$

is a *convex combination* if $0 \leqslant c_1 \leqslant 1$ and $c_1 + c_2 = 1$.

3. The *Euclidean norm* of a vector $\mathbf{y}$ is the real, non-negative number (denoted $\| \mathbf{y} \|$) computed from the formula

$$\| \mathbf{y} \| = \sqrt{y_1^2 + y_2^2 + \ldots + y_N^2}$$

The *Euclidean distance* between two vectors $\mathbf{x}$ and $\mathbf{y}$ is the norm of the difference vector, i.e. $\| \mathbf{x} - \mathbf{y} \|$. A *neighbourhood* of a vector $\mathbf{y}$ is the set of all vectors $\mathbf{x}$ such that $\| \mathbf{x} - \mathbf{y} \| < \varepsilon$ for some positive number ε.

4. The term *subspace* is used here in the sense of the set of all vectors $\mathbf{y}$ such that $\mathbf{y}$ is the sum of a particular vector $\mathbf{q}$ and an arbitrary linear combination of M linearly independent vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_M$. The subspace is a *hyperplane* if M = N-1 (the analog of a *plane* in "ordinary" space). A *polyhedron* in N-dimensional space is a closed region bounded by hyperplanes.

5. A vector $\mathbf{u}$ and a (generally, complex) number λ satisfying the vector equation

$$\mathbf{A}\,\mathbf{u} = \lambda\,\mathbf{u}$$

are respectively called an *eigenvector* and the corresponding *eigenvalue* of the (N by N) square matrix $\mathbf{A}$. The maximum possible number of distinct eigenvalues of a given matrix is N. The matrices considered in this thesis are *symmetric* (that is, $a_{jk} = a_{kj}$ for all j and k) and for such matrices all eigenvalues are *real*. A symmetric matrix $\mathbf{A}$ is *positive definite* if the scalar quantity $\mathbf{x}^T\mathbf{A}\,\mathbf{x}$ is greater than zero for all nonzero vectors $\mathbf{x}$. Equivalently, all eigenvalues are greater than zero. If both positive and negative eigenvalues exist the matrix $\mathbf{A}$ is called *indefinite* and $\mathbf{x}^T\mathbf{A}\,\mathbf{x}$ can take both positive and negative values, depending on $\mathbf{x}$. *Zero* eigenvalues appear if and only if the N columns of $\mathbf{A}$ are linearly dependent. The number of nonzero eigen-

values is equal to the maximum number of linearly independent columns, also called the *rank* of the matrix. A matrix with rank less than N is called *singular*.

6. The process of determining eigenvalues and eigenvectors is referred to in this thesis as *eigensystem analysis*. Numerical difficulties associated with matrix manipulations are often significant if there is a large spread in the magnitudes of the eigenvalues. A measure frequently used is the *condition number*, defined for a positive definite symmetric matrix as the ratio of the largest to the smallest eigenvalue.

APPENDIX B

FORTRAN SUBROUTINE TO EVALUATE THE OBJECTIVE FUNCTION AND ITS DERIVATIVES FOR TIME DOMAIN (IMPULSE RESPONSE) OPTIMIZATION

```
      SUBROUTINE FNC(PARVEC,ESQ,GAIN,GRAD,HESS,RESP)
C
C     CALCULATES THE IMPULSE RESPONSE OF A CASCADE-FORM DIGITAL
C     FILTER AND THE FOLLOWING QUANTITIES WITH RESPECT TO A GIVEN
C     'TARGET' RESPONSE, OVER ANY SUBSET OF TIME INDICES:
C       (1)THE TOTAL (WEIGHTED) SQUARED ERROR
C       (2)THE ERROR GRADIENT VECTOR (W.R.T. VARIABLE PARAMETERS)
C       (3)THE GAUSS-NEWTON APPROXIMATION TO THE ERROR HESSIAN
C          MATRIX.
C       (4)THE FULL HESSIAN MATRIX INVOLVING SECOND DERIVATIVES
C          OF THE FILTER OUTPUT TIME SAMPLES
C
C     OPTIONS (2),(3),(4) AND THE OUTPUTTING OF THE RESPONSE
C     ARE SWITCHED ON AND OFF WITH LOGICAL VARIABLES
C
C     A AND B OR C AND D COEFFICIENTS FOR A SECOND-ORDER
C     SECTION ARE HELD IN THE ARRAYS CA(K) AND CB(K), K=1,NS
C     THE KTH SECTION IS INDICATED AS BEING A NUMERATOR OR
C     A DENOMINATOR BY SETTING IT(K) POSITIVE OR NEGATIVE
C     RESPECTIVELY.
C     SOME COEFFICIENTS MAY BE FIXED, IN WHICH CASE THEIR
C     VALUES MUST ALREADY BE IN THESE ARRAYS. VARIABLE
C     COEFFICIENTS HAVE THEIR CURRENT VALUES TRANSFERRED FROM
C     THE PARAMETER VECTOR 'PARVEC' AS THE FIRST ACTION OF
C     THIS SUBROUTINE. THE ARRAY IV(K),K=1,NS CONTAINS
C     INFORMATION ABOUT WHICH COEFFICIENTS OF THE INDIVIDUAL
C     SECTIONS ARE VARIABLE, I.E.
C       IV=1  NEITHER COEFF. VARIABLE
C       IV=2  A OR C COEFF. VARIABLE
C       IV=3  B OR D COEFF. VARIABLE
C       IV=4  BOTH COEFFS. VARIABLE.
C
C     THE ARRAY MF(M),M=1,NF CONTAINS THE TIME INDICES OF
C     THE IMPULSE RESPONSE SAMPLES TO BE MATCHED, WITH Y(M)
C     AND WT(M) CONTAINING CORRESPONDING TARGET VALUES
C     AND WEIGHT FACTORS. IRL CONTAINS THE HIGHEST IMPULSE
C     RESPONSE INDEX NEEDED (MAX OF THE MF(M) VALUES).
C
      LOGICAL KB,SWGRAD,SWHESS,SWRESP,SWNEWT
      DIMENSION PARVEC(1),GRAD(1),HESS(1),RESP(1)
      DIMENSION RSPL(202),G2(202)
      EQUIVALENCE (RSPL,G2)
      COMMON/CONT/NF,NS,NP,NMAT,W,SWRESP,SWGRAD,SWHESS,SWNEWT,IRL
      COMMON/FREQ/MF(50),Y(50),WT(50)
      COMMON/SECT/CA(20),CB(20),IT(20),IV(20)
      COMMON TPAR(40),ST1(20),ST2(20),FM(50),GIR(1)
      DATA MIRL,MGIR/200,1110/
```

```
C
C      PUT CURRENT VALUES OF PARAMETERS INTO SECTION TABLES
C      AND ZERO STATE VARIABLES IN FILTER SIMULATION.
C
       IF(IRL.GT.MIRL)GO TO 101
       IRLP=IRL+1
       GAIN=1.0
       N=0
       DO 10 K=1,NS
       GO TO (11,12,13,14),IV(K)
   12  N=N+1
       CA(K)=PARVEC(N)
       GO TO 11
   13  N=N+1
       CB(K)=PARVEC(N)
       GO TO 11
   14  N=N+2
       CA(K)=PARVEC(N-1)
       CB(K)=PARVEC(N)
   11  ST1(K)=0.0
       ST2(K)=0.0
C
C      TEST FOR STABILITY
C      IF UNSTABLE, RETURN GAIN=-1.0 AND ESQ=1.E9
C      DISALLOW NONMINIMUM PHASE FILTERS ALSO BECAUSE
C      GRADIENT FILTERS WOULD BE UNSTABLE
C
       A1=CB(K)
       A2=1.+CA(K)+A1
       A3=1.-CA(K)+A1
       IF(A1.LT.0.999999.AND.A2.GT.1.E-6.AND.A3.GT.1.E-6)
      *   GO TO 10
       IF(.NOT.SWGRAD)GO TO 40
       IF(A1.LT.1.0.AND.A2.GT.0.0.AND.A3.GT.0.0)GO TO 10
   40  GAIN=-1.0
       ESQ=1.0E9
       IF(SWGRAD)GO TO 100
       WRITE(2,41)
   41  FORMAT(*  REQUEST FOR GRADIENT FROM UNSTABLE FILTER*)
       DO 106 I=1,NP
  106  GRAD(I)=0.0
       ESQ=-ESQ
       GO TO 100
   10  CONTINUE
C
C      CALCULATE IMPULSE RESPONSE OF MODEL FILTER
C
       SI=1.0
       DO 71 I=1,IRL
       DO 72 K=1,NS
       STN=SI
       IF(IT(K).LT.0)GO TO 73
C
C      NUMERATOR SECTION (ZERO PAIR)
C
       SI=SI+CA(K)*ST1(K)+CB(K)*ST2(K)
       GO TO 74
```

```
C
C       DENOMINATOR SECTION (POLE PAIR)
C
   73 STN=STN-CA(K)*ST1(K)-CB(K)*ST2(K)
      SI=STN
   74 ST2(K)=ST1(K)
      ST1(K)=STN
   72 CONTINUE
      RSPL(I)=SI
      IF(.NOT.SWRESP)RESP(I)=SI
      SI=0.0
   71 CONTINUE
C
C       CALCULATE TOTAL SQUARED ERROR
C
      ESQ=0.0
      DO 21 I=1,NF
      J=MF(I)
      FV=RSPL(J)-Y(I)
      FM(I)=FV*WT(I)
      ESQ=ESQ+FV*FM(I)
   21 CONTINUE
      IF(SWGRAD)GO TO 100
C
C****************** GRADIENT REQUIRED **********************
C
C       CALCULATE IMPULSE RESPONSES OF GRADIENT FILTERS
C       FOR ALL VARIABLE SECTIONS
C
      NST=0
      DO 76 K=1,NS
      IF(IV(K).EQ.1)GO TO 76
      S1=0.0
      S2=0.0
C
C       PREFIX SEQUENCE WITH A ZERO FOR GRADIENT WRT B OR D
C
      NST=NST+1
      IF(NST.GT.MGIR)GO TO 102
      GIR(NST)=0.0
      IF(IT(K).LT.0)GO TO 77
C
C       NUMERATOR SECTION
C
      DO 75 I=1,IRL
      STN=RSPL(I)-CA(K)*S1-CB(K)*S2
      NST=NST+1
      IF(NST.GT.MGIR)GO TO 102
      GIR(NST)=S1
      S2=S1
      S1=STN
   75 CONTINUE
      GO TO 76
C
C       DENOMINATOR SECTION
C
   77 DO 78 I=1,IRL
      STN=-(RSPL(I)+CA(K)*S1+CB(K)*S2)
      NST=NST+1
      IF(NST.GT.MGIR)GO TO 102
      GIR(NST)=S1
      S2=S1
      S1=STN
   78 CONTINUE
   76 CONTINUE
```

```
C
C      CALCULATE GRADIENT COMPONENTS
C
       NPR=-1
       N=0
       DO 22 K=1,NS
       IF(IV(K).EQ.1)GO TO 22
       NPR=NPR+1
       GO TO (22,23,24,25),IV(K)
C
C      A OR C ONLY VARIABLE
C
   23  NGR=NPR*IRLP+2
       N=N+1
       CALL XCOR1(FM,GIR(NGR),GRAD(N))
       GO TO 22
C
C      B OR D ONLY VARIABLE
C
   24  NGR=NPR*IRLP+1
       N=N+1
       CALL XCOR1(FM,GIR(NGR),GRAD(N))
       GO TO 22
C
C      BOTH PARAMETERS VARIABLE
C
   25  NGR=NPR*IRLP+2
       N=N+2
       CALL XCOR1(FM,GIR(NGR),GRAD(N-1))
       CALL XCOR1(FM,GIR(NGR-1),GRAD(N))
   22  CONTINUE
       IF(SWHESS)GO TO 100
C
C****************** PARTIAL HESSIAN REQUIRED ********************
C
C      CALCULATE COMPONENTS OF APPROXIMATE HESSIAN
C      (J TRANSPOSE J) MATRIX
C
       NPR=-1
       IM=0
       DO 31 K=1,NS
       IF(IV(K).EQ.1)GO TO 31
       NPR=NPR+1
       KB=.FALSE.
       GO TO (31,32,33,34),IV(K)
C
C      A OR C ONLY VARIABLE
C
   32  NGR1=NPR*IRLP+2
       ASSIGN 31 TO KSW
       GO TO 50
C
C      B OR D ONLY VARIABLE
C
   33  NGR1=NPR*IRLP+1
       ASSIGN 31 TO KSW
       GO TO 50
C
```

```
C      BOTH PARAMETERS VARIABLE
C
   34 NGR1=NPR*IRLP+2
      ASSIGN 35 TO KSW
      GO TO 50
   35 KB=.TRUE.
      NGR1=NPR*IRLP+1
      ASSIGN 31 TO KSW
      GO TO 50
   31 CONTINUE
      IF(SWNEWT)GO TO 100
C
C****************** FULL HESSIAN REQUIRED **********************
C
C      CALCULATE SECOND DERIVATIVE TERMS TO PRODUCE FULL HESSIAN
C
      G2(1)=0.0
      G2(2)=0.0
      IM=1
      INC=N
      NG2=IRL+2
      NPR=-1
C
C      RUN THROUGH THE FILTER SECTIONS (INDEX K)
C
      DO 51 K=1,NS
      GO TO (51,85,85,87),IV(K)
   85 N1=IM+1
      GO TO 53
   87 N1=IM+2
      N2=IM+INC+1
   53 NPR=NPR+1
      KGST=NPR*IRLP+1
C
C      RUN THROUGH SECTIONS K ONWARD (INDEX L)
C      CALCULATE THE SECOND DERIVATIVE TIME SEQUENCE (G2) BY
C      FILTERING THE KTH STORED GRADIENT SEQUENCE WITH THE
C      LTH SUBFILTER
C
      DO 52 L=K,NS
      IF(IV(L).EQ.1)GO TO 52
      IF(L.EQ.K.AND.IT(K).EQ.1)GO TO 52
      S1=0.0
      S2=0.0
      IF(IT(L).LT.0)GO TO 54
C
C      LTH SECTION IS NUMERATOR (ZERO PAIR)
C
      DO 55 I=1,IRL
      STN=GIR(KGST+I)-CA(L)*S1-CB(L)*S2
      G2(I+2)=S1
      S2=S1
   55 S1=STN
      GO TO 56
C
C      LTH SECTION IS DENOMINATOR (POLE PAIR)
C
   54 DO 57 I=1,IRL
      STN=-(GIR(KGST+I)+CA(L)*S1+CB(L)*S2)
      G2(I+2)=S1
      S2=S1
   57 S1=STN
      IF(L.NE.K)GO TO 56
```

```
C
C       DOUBLE TERMS FOR K=L WHEN DENOMINATOR
C
        DO 58 I=3,NG2
   58   G2(I)=G2(I)+G2(I)
C
C       VECTOR CONTROL OUT TO SEPARATE CODE SEQUENCES
C       FOR THE IV(K),IV(L) COMBINATIONS TO SIMPLIFY
C       HESSIAN MATRIX INDEX CALCULATIONS
C
   56   IVB=3*IV(K)+IV(L)-7
        GO TO (61,62,63,64,65,66,67,68,69),IVB
C
C       BOTH IN K, BOTH IN L
C
   69   CALL XCOR1(FM,G2(3),VAL1)
        CALL XCOR1(FM,G2(2),VAL2)
        CALL XCOR1(FM,G2,VAL3)
        IF(L.NE.K)GO TO 81
        HESS(IM)=HESS(IM)+VAL1
        HESS(IM+1)=HESS(IM+1)+VAL2
        HESS(IM+INC)=HESS(IM+INC)+VAL3
        GO TO 52
   81   HESS(N1)=HESS(N1)+VAL1
        HESS(N1+1)=HESS(N1+1)+VAL2
        HESS(N2)=HESS(N2)+VAL2
        HESS(N2+1)=HESS(N2+1)+VAL3
        N1=N1+2
        N2=N2+2
        GO TO 52
C
C       BOTH IN K, B IN L
C
   68   CALL XCOR1(FM,G2(2),VAL1)
        CALL XCOR1(FM,G2,VAL2)
   82   HESS(N1)=HESS(N1)+VAL1
        HESS(N2)=HESS(N2)+VAL2
        N1=N1+1
        N2=N2+1
        GO TO 52
C
C       BOTH IN K, A IN L
C
   67   CALL XCOR1(FM,G2(3),VAL1)
        CALL XCOR1(FM,G2(2),VAL2)
        GO TO 82
C
C       B IN K, BOTH IN L
C
   66   CALL XCOR1(FM,G2(2),VAL1)
        CALL XCOR1(FM,G2,VAL2)
   83   HESS(N1)=HESS(N1)+VAL1
        HESS(N1+1)=HESS(N1+1)+VAL2
        N1=N1+2
        GO TO 52
C
C       B IN K, B IN L
C
   65   CALL XCOR1(FM,G2,VAL1)
        IF(L.NE.K)GO TO 84
   86   HESS(IM)=HESS(IM)+VAL1
        GO TO 52
   84   HESS(N1)=HESS(N1)+VAL1
        N1=N1+1
        GO TO 52
```

```
C
C      B IN K, A IN L
C
   64 CALL XCOR1(FM,G2(2),VAL1)
      GO TO 84
C
C      A IN K, BOTH IN L
C
   63 CALL XCOR1(FM,G2(3),VAL1)
      CALL XCOR1(FM,G2(2),VAL2)
      GO TO 83
C
C      A IN K, B IN L
C
   62 CALL XCOR1(FM,G2(2),VAL1)
      GO TO 84
C
C      A IN K, A IN L
C
   61 CALL XCOR1(FM,G2(3),VAL1)
      IF(L.NE.K)GO TO 84
      GO TO 86
   52 CONTINUE
C
C
C      UPDATE POINTER TO DIAGONAL ELEMENT OF HESS (IM)
C      AND ITS INCREMENT FOR NEXT TIME (INC)
C
      GO TO (51,91,91,92),IV(K)
   91 IM=IM+INC
      INC=INC-1
      GO TO 51
   92 IM=IM+INC+INC-1
      INC=INC-2
   51 CONTINUE
  100 RETURN
C
C*******************************************************************
C
C      EXTENDED RANGE OF DO LOOP ...DO 31... FOR CALCULATION
C      OF J TRANSPOSE J MATRIX
C      RUN THROUGH SECTIONS FROM K TO NS
C
   50 LPR=NPR-1
      DO 36 L=K,NS
      IF(IV(L).EQ.1)GO TO 36
      LPR=LPR+1
      IF(KB)GO TO 38
      GO TO (36,37,38,39),IV(L)
C
C      A OR C ONLY VARIABLE
C
   37 NGR2=LPR*IRLP+2
      IM=IM+1
      CALL XCOR2(WT,GIR(NGR1),GIR(NGR2),HESS(IM))
      GO TO 36
C
C      B OR D ONLY VARIABLE
C
   38 NGR2=LPR*IRLP+1
      IM=IM+1
      CALL XCOR2(WT,GIR(NGR1),GIR(NGR2),HESS(IM))
      KB=.FALSE.
      GO TO 36
```

```
C
C       BOTH PARAMETERS VARIABLE
C
   39   IM=IM+2
        NGR2=LPR*IRLP+2
        CALL XCOR2(WT,GIR(NGR1),GIR(NGR2),HESS(IM-1))
        CALL XCOR2(WT,GIR(NGR1),GIR(NGR2-1),HESS(IM))
   36   CONTINUE
        GO TO KSW,(31,35)
C
C       ERROR MESSAGES
C
  101   WRITE(2,103)MIRL
  104   STOP
  102   WRITE(2,105)
        GO TO 104
  103   FORMAT(*  MAX IMP RESP LENGTH PERMITTED IS *,I5)
  105   FORMAT(*  GRADIENT IMPULSE RESPONSE BLOCK FULL*)
        END
```

```
        SUBROUTINE XCOR1(AI,AJ,VAL)
        DIMENSION AI(1),AJ(1)
        COMMON/CONT/NF
        COMMON/FREQ/MF(1)
        VAL=0.0
        DO 21 I=1,NF
        J=MF(I)
        VAL=VAL+AI(I)*AJ(J)
   21   CONTINUE
        VAL=VAL+VAL
        RETURN
        END
```

```
        SUBROUTINE XCOR2(AI,AJ1,AJ2,VAL)
        DIMENSION AI(1),AJ1(1),AJ2(1)
        COMMON/CONT/NF
        COMMON/FREQ/MF(1)
        VAL=0.0
        DO 21 I=1,NF
        J=MF(I)
        VAL=VAL+AI(I)*AJ1(J)*AJ2(J)
   21   CONTINUE
        VAL=VAL+VAL
        RETURN
        END
```

NOTATION

a_k coefficient of z^{-1} in numerator, kth filter section.

A_0 gain factor in cascade-form IIR digital filter.

A_1, A_2 sums-of-products, formulation B3.

A_{BB}, A_{YB}, A_{YY} sums-of-products, formulation B2.

$\mathbf{A}$ general N by N matrix; approximation to Hessian, quasi-Newton method.

b_k coefficient of z^{-2} in numerator, kth filter section.

$B(\mathbf{x}, \theta)$ general response function, type unspecified.

$B_m(\mathbf{x})$ achieved value of mth component function.

c_k coefficient of z^{-1} in denominator, kth filter section.

c_m mth sample of cepstrum.

$\mathbf{C}^{(k)}$ update matrix for $\mathbf{A}^{(k)}$, quasi-Newton method.

d_0, d_1, d_2 coefficients in special-purpose second-order filter sections.

d_k coefficient of z^{-2} in denominator, kth filter section.

d_{max} maximum radial distance for discrete solution search.

$D(\theta)$, $D(z)$ function derived from denominator of transfer function.

$D_k(z)$ $(= 1 + c_k z^{-1} + d_k z^{-2})$

D_m $(= D(\theta_m))$

D diagonal matrix, Cholesky factor; block diagonal matrix, Bunch-Parlett factor.

e_0, e_1 coefficients in special-purpose second-order filter sections.

$\mathbf{e}_i$ ith unit vector (dimension N)

E diagonal matrix, Gill-Murray method.

$f_m(\mathbf{x})$ mth component of SS objective function.

$F(\mathbf{x})$ objective function.

$F^*(\mathbf{x},q)$ objective function with explicit dependence on gain factor q.

F_s sampling frequency (Hz).

g safety factor for d_{max}.

g_i ith component of **g** (gradient).

g gradient vector.

h random step size, Suk and Mitra method.

h_i ith sample of filter impulse response.

h_{ij} component of **H** (Hessian) in ith row, jth column.

H(z) z-transform of h_i (filter transfer function).

H_0, H_1 cube edge sizes, Suk and Mitra method.

H Hessian matrix of objective function.

i index for general component of N-dimensional vector.

I N by N identity matrix.

j index for column of matrix; general index.

J total number of real poles or zeros.

$\mathbf{J}$ Jacobian matrix (M by N) of least-squares objective function.

k a constant; index for general second-order section.

(k) (superscript) values used at, or prevailing before, the kth iteration.

K total number of sections in cascade-form IIR digital filter.

K_D number of denominator sections.

K_N number of numerator sections.

l_1 index of lowest discrete frequency in transition band.

l_2 index of highest discrete frequency in transition band.

l_{ij} component of $\hat{\mathbf{L}}$, ith row, jth column.

L distance from "continuous optimum" to its rounded equivalent.

$-L_m$ lower limit of error at mth frequency.

$\mathcal{L}$ Lagrangian function.

$\mathbf{L}$ unit lower triangular matrix, Cholesky factor or Bunch-Parlett factor.

$\hat{\mathbf{L}}$ lower triangular matrix, Cholesky "square-root" factor.

m Index for general component function with SS objective function; index for general discrete frequency.

M number of component functions; number of discrete frequencies; number of variable transition-band samples.

$M(\theta)$ magnitude-squared response of digital filter.

M_m ($= M(\theta_m)$)

N dimension of parameter vector; duration of impulse response, FIR digital filter.

$N(\theta)$, $N(z)$ function derived from numerator of transfer function.

N_c upper quefrency limit of cepstral window.

$N_k(z)$ ($= 1 + a_k z^{-1} + b_k z^{-2}$)

N_m ($= N(\theta_m)$)

N_p number of iterations between root reflections and pairing changes.

$2p$ exponent in least-pth objective function.

$p_1, p_2 \ldots$ real poles

p_i ith component of $\mathbf{p}$ (search direction)

P_{Bi}, P_{Yi} sums-of-products, formulation B2.

P_{ij} sum -of-products, formulations B2 and B3.

P_{1i}, P_{2i} sums-of-products, formulation B3.

$\mathbf{p}$ search direction vector.

q optimal gain term or factor; auxiliary variable for linear program; random search bias exponent.

$q_1, q_2 \ldots$ distances between real poles

$Q_i(\mathbf{x}) \geq 0$ general inequality constraint.

$Q_i(\mathbf{x}) = 0$ general equality constraint.

Q_{Bij}, Q_{Yij} sums-of-products, formulation B2.

Q_{1ij}, Q_{2ij} sums-of-products, formulation B3.

Q_{Cmk} ($= 2 (1 - d_k) \cos \theta_m$)

Q_{Dmk} ($= - 4d_k - 2c_k \cos \theta_m$)

$\mathbf{Q}$ symmetric orthogonal projection matrix.

r penalty factor; radius of complex pole in z-plane

r_k radius of complex pole, kth filter section.

r_{ij} component of $\mathbf{R}$ (Gauss-Newton matrix) in ith row, jth column.

R^1 tolerance on pole position in radial direction.

R_{Amk} ($= 2a_k + (1 + 3b_k) \cos \theta_m$)

R_{Bmk} ($= 4b_k + 3a_k \cos \theta_m + 2 \cos 2\theta_m$)

R_{Cmk} ($= 2c_k + (1 + 3d_k) \cos \theta_m$)

R_{Dmk} ($= 4d_k + 3c_k \cos \theta_m + 2 \cos 2\theta_m$)

$\mathbf{r}$ parameter vector which minimizes quadratic form.

$\mathbf{R}$ Gauss-Newton matrix ($= 2\mathbf{J}^T\mathbf{J}$)

s steplength in radial random search.

$S(\theta)$ magnitude response of digital filter.

S_1, S_2 subspaces generated by points $\mathbf{x}_1, \mathbf{x}_2$ respectively respectively and a linearly independent set of vectors.

S_{Amk} ($= a_k + (1 + b_k) \cos \theta_m$)

S_{Bmk} ($= b_k + a_k \cos \theta_m + 2 \cos 2\theta_m$)

S_{Cmk} ($= - (c_k + (1 + d_k) \cos \theta_m)$)

S_{Dmk} ($= - (d_k + c_k \cos \theta_m + \cos 2\theta_m)$)

$\mathbf{s}^{(k)}$ parameter change, kth iteration ($\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$).

$\mathbf{S}$ Hessian correction matrix ($\mathbf{R} + \mathbf{S} = \mathbf{H}$)

t auxiliary variable, Botsaris & Jacobson method.

t_i ($= P_{Yi} - 2q P_{Bi}$, formulation B2)

T sampling interval.

T (superscript) denotes matrix transpose.

T_{mk} ($= 1 + a_k^2 + b_k^2 + 2a_k (1 + b_k) \cos \theta_m + 2b_k \cos 2\theta_m$)

$\mathbf{t}$ direction of negative curvature of $F(\mathbf{x})$.

$\mathbf{T}$ approximation to inverse Hessian, quasi-Newton method.

u_i ith sample of filter input sequence.

$U(z)$ z-transform of u_i.

U_m upper limit of error at mth frequency.

U_{mk} ($= 1 + c_k^2 + d_k^2 + 2c_k\,(1 + d_k)\cos\theta_m + 2d_k \cos 2\theta_m$)

$\mathbf{u}_i$ ith normalized eigenvector of $\mathbf{H}$.

$\mathbf{U}$ upper triangular matrix.

v_i ith sample of filter output sequence.

$V(z)$ z-transform of v_i.

V_{mk} ($= a_k^2 + 2b_k^2 + a_k\,(1 + 3b_k)\cos\theta_m + 2b_k \cos 2\theta_m$)

$\mathbf{v}$ auxiliary vector, quasi-Newton method.

w_m weight factor for mth component function.

W sum-of-products, formulation B3.

W_{mk} ($= c_k^2 + 2d_k^2 + c_k\,(1 + 3d_k)\cos\theta_m + 2d_k \cos 2\theta_m$)

x_i ith component of $\mathbf{X}$ (parameter vector).

x_m mth sample of complex cepstrum.

X_{mk} ($= 2\,(1 - d_k^2 + c_k\,(1 - d_k)\cos\theta_m$))

$\|X\|_p$ 1_p norm of signal x

$\mathbf{X}$ vector of parameters.

$\mathbf{X}^*$ augmented vector of parameters.

$Y(\theta)$ target function for general response.

$Y_L(\theta)$ lower limit function for general response.

$Y_U(\theta)$ upper limit function for general response.

Y_m target value for mth component function;
target value at mth discrete frequency.

Y_{Lm} lower response limit at mth discrete frequency.

Y_{Um} upper response limit at mth discrete frequency.

$\mathbf{y}$ arbitrary vector (dimension N).

$\mathbf{y}^{(k)}$ gradient change, kth interation ($\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$).

z complex variable for sampled-data transform.

$\mathbf{z}_1$, $\mathbf{z}_2$ parameter vectors minimizing $F(\mathbf{x})$ in subspaces S_1 and S_2 respectively.

α steplength.

$\alpha_0, \alpha_1, \ldots \alpha_m$ feedforward coefficients in direct-form digital filter.

β Marquardt (Levenberg) parameter; bound in Gill-Murray matrix factorization.

$\beta_0, \beta_1, \ldots \beta_n$ feedback coefficients in direct-form digital filter.

γ_0 gain term in parallel-form IIR digital filter.

γ_k first numerator coefficient in kth section of parallel-form IIR digital filter.

γ_m mth sample of "associated cepstrum function" (= $2m\, c_m$)

δ_1 passband ripple tolerance; small negative quantity.

δ_2 stopband ripple tolerance; small positive quantity.

δ_k second numerator coefficient in kth section of parallel-form IIR digital filter.

δ_m response tolerance at mth discrete frequency.

$\Delta\mathbf{x}$ small change to $\mathbf{x}$.

$\varepsilon(\theta)$ arbitrary function of θ defined on $0 \leqslant \theta \leqslant \pi$.

ε arbitrary small positive number.

θ digital frequency variable (polar angle).

θ_m mth discrete frequency.

θ_p passband edge (digital) frequency.

θ_s stopband edge (digital) frequency.

λ	auxiliary complex variable.
λ_i	ith eigenvalue of **H**.
λ_x, λ_y	cartesian coordinates of λ plane.
λ_x^*	coordinate ($\lambda_x = -(\lambda_x^*)^2$)
μ	smallest eigenvalue of **R**.
μ_i	ith Lagrange multiplier.
ρ	parameter in quasi-Newton rank-2 family.
σ_w^2	mean-square convergence criterion in frequency domain.
τ	absolutely largest eigenvalue of **S**.
$\tau(\theta)$	group delay response of digital filter.
ϕ	argument (angle) of complex pole.
$\phi(\theta)$	phase response of digital filter.
ϕ_k	angle of complex pole in 1st or 2nd quadrant, kth filter section.
Φ^1	tolerance on pole position in ϕ direction.
$\chi(z)$	($= N(z^{-1})\, D(z)$) auxiliary function for group delay calculation.

LIST OF ABBREVIATIONS

Although they are introduced as they occur in the text, the abbreviations used in this thesis are for convenience summarized below. Omitted from the list are the special mnemonic codes for search direction and line search algorithms, which appear in Tables 5.1 (page 215) and 5.2 (page 221) respectively.

A/D	analog-to-digital (conversion)
ALU	arithmetic and logic unit
BFGS	Broyden-Fletcher-Goldfarb-Shanno (optimization method)
CG	conjugate gradient (optimization method)
CP	central processor
D/A	digital-to-analog (conversion)
DFP	Davidon-Fletcher-Powell (optimization method)
DFT	discrete Fourier transform
FF	Fletcher-Freeman (optimization method)
FFT	fast Fourier transform
FIR	finite impulse response (digital filter)
GASP	"General Arithmetic Signal Processor"
GM	Gill-Murray (optimization method)
GN	Gauss-Newton (optimization method)
GR	Greenstadt (optimization method)
IDFT	inverse discrete Fourier transform
IIR	infinite impulse response (digital filter)
I/O	input/output
LC	inductance-capacitance
LSB	least significant bit

LSI	large scale integration
MQ	Marquardt (optimization method)
MSI	medium scale integration
N/S	normalizer/shifter
QN	quasi-Newton (optimization method)
SD	second derivative (optimization method)
SR1	symmetric rank one
SS	sum of squares
SSI	small scale integration
TM	true Marquardt (optimization method)
VLSI	very large scale integration

BIBLIOGRAPHY

I.A. Athanassopoulos and J.F. Kaiser (1970), "Constrained optimization problems in digital filter design," presented at *IEEE Arden House Workshop on Digital Filtering*, Harriman, N.Y., January 1970.

I.A. Athanassopoulos, J.D. Schoeffler and A.D. Waren (1966), "Time domain synthesis by nonlinear programming," in *Proc. 4th Annual Allerton Conference on Circuit and System Theory*, Monticello, Il., October 1966, pp. 766-775.

E. Avenhaus (1971), "An optimization procedure to minimize the wordlength of digital filter coefficients," presented at *1971 Imperial College Symposium on Digital Filtering*, Imperial College, London, August-September 1971.

E. Avenhaus (1972a), "A proposal to find suitable canonic structures for the implementation of digital filters with small coefficient wordlength," *Nachrichtentechnische Zeitschrift*, vol. 25, pp. 377-382.

E. Avenhaus (1972b), "On the design of digital filters with coefficients of limited wordlength," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, No. 3, pp. 206-212.

P. Balakrishnan and K.P. Rajappan (1974), "Synthesis of recursive digital filters," *Computer Aided Design*, vol. 6, No. 3, pp. 148-152.

J.W. Bandler and B.L. Bardakjian (1973), "Least *pth* optimization of recursive digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-21, no. 5, pp. 460-470.

J.W. Bandler, B.L. Bardakjian and J.H.K. Chen (1975), "Design of recursive digital filters with optimized wordlength coefficients," *Computer Aided Design*, vol. 7, no. 3, pp. 151-156.

J.W. Bandler and C. Charalambous (1974), "Nonlinear programming using minimax techniques," *Journal of Optimization Theory and Applications*, vol. 13, pp. 607-619.

Y.Bard (1970), "Comparison of gradient methods for the solution of nonlinear parameter estimation problems," *Society for Industrial and Applied Mathematics; Journal of Numerical Analysis*, vol. 7, pp. 157-186.

J.G.P. Barnes (1965) "An algorithm for solving nonlinear equations based on the secant method," *Computer Journal*, vol. 8, pp. 66-72.

J.T. Betts (1976), "Solving the nonlinear least square problem: application of a general method," *Journal of Optimization Theory and Applications*, vol. 18, no. 4, pp. 469-483.

M.C. Biggs (1977), "The estimation of the Hessian matrix in nonlinear least squares problems with non-zero residuals," *Mathematical Programming*, vol. 12, pp. 67-80.

R.E. Bogner (1977), "Random operations in signal processing," in *Convention Digest, Institution of Radio and Electronics Engineers, Australia; 16th International Convention and Exhibition,* Melbourne, 8-12 August 1977, pp. 337-340.

R.E. Bogner and A.G. Constantinides (1975), *"Introduction to Digital Filtering",* London, John Wiley and Sons, Ltd., 1975.

R. Boite, H. Dubois and H. Leich (1974), "Optimization of digital filters in the discrete space of coefficients," *Electronics Letters,* vol. 10, no. 10, pp. 179-180.

C.A. Botsaris and D.H. Jacobson (1976), "A Newton-type curvilinear search method for optimization," *Journal of Mathematical Analysis and Applications,* vol. 54, pp. 217-229.

M.J. Box (1966), "A comparison of several current optimization methods, and the use of transformations in constrained problems," *Computer Journal,* vol. 9, pp. 67-77.

F. Brglez (1978), "Digital filter design with short wordlength coefficients," *IEEE Transactions on Circuits and Systems,* vol. CAS-25, no. 12, pp. 1044-1050.

F. Brophy and A.C. Salazar (1973), "Considerations of the Pade approximant technique in the synthesis of recursive digital filters," *IEEE Transactions on Audio and Electroacoustics,* vol. AU-21, no. 6, pp. 500-505.

F. Brophy and A.C. Salazar (1974), "Recursive digital filter synthesis in the time domain," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-22, no. 1, pp. 45-55.

C.G. Broyden (1965), "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation*, vol. 19, pp. 577-593.

C.G. Broyden (1967), "Quasi-Newton methods and their application to function minimization," *Mathematics of Computation*, vol. 21, pp. 368-381.

C.G. Broyden (1970), "The convergence of a class of double-rank minimization algorithms," *Journal of the Institute of Mathematics and its Applications*, vol. 6, pp. 76-90 (part 1), and pp. 222-231 (part 2).

C.G. Broyden (1972), "Quasi-Newton methods," in W. Murray (ed.) *"Numerical Methods for Nonlinear Optimization,"* London and New York, Academic Press, 1972.

J.R. Bunch and B.N. Parlett (1971), "Direct methods for solving symmetric indefinite systems of linear equations," *Society for Industrial and Applied Mathematics; Journal of Numerical Analysis*, vol. 8, no. 4, pp. 639-655.

C.S. Burrus and T.W. Parks (1970), "Time domain design of recursive digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-18, no. 2, pp. 137-141.

J.A. Cadzow (1976), "Recursive digital filter synthesis via gradient-based algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, no. 5, pp. 349-355.

R.Cann and K. Steiglitz (1978), "Implementation of a pole-zero analysis-synthesis system for speech," presented at *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tulsa, Ok., April 10-12, 1978.

R.K. Cavin, C.H. Ray and V.T. Rhyne (1969), "The design of optimal convolutional filters via linear programming," *IEEE Transactions on Geoscience Electronics*, vol. GE-7, pp. 142-145.

C. Charalambous and M.J. Best (1974), "Optimization of recursive digital filters with finite wordlength," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-22, pp. 424-431.

R.L. Christensen, W.J. Strong and E.P. Palmer (1976), "A comparison of three methods of extracting resonance information from predictor-coefficient coded speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, pp. 8-14.

R. Coates (1975), "Fourier transform methods," in R.E. Bogner and A.G. Constantinides (eds.) *Introduction to Digital Filtering*, London, John Wiley and Sons, Ltd., 1975.

R.E. Crochiere (1972), "Digital ladder structures and coefficient sensitivity," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, no. 4, pp. 240-246.

R.E. Crochiere (1975), "A new statistical approach to the coefficient wordlength problem for digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-22, no. 3, pp. 190-196.

R.J. Dakin (1966), "A tree-search algorithm for mixed-integer programming problems," *Computer Journal*, vol. 8, pp. 250-255.

W.C. Davidon (1959), "Variable metric method for minimization," *U.K. Atomic Energy Commission; Research and Development Report ANL-5990*.

W.C. Davidon (1968), "Variance algorithms for minimization," *Computer Journal*, vol. 10, pp. 406-410.

W.C. Davidon (1975), "Optimally-conditioned optimization algorithms without line searches," *Mathematical Programming*, vol. 9, pp. 1-30.

W.C. Davidon (1976), "New least-square algorithms," *Journal of Optimization Theory and Applications*, vol. 18, no. 2, pp. 187-197.

M. Davies and I.J. Whitting (1971), "A modified form of Levenberg's correction," in F.A. Lootsma (ed.) *Numerical Methods for Nonlinear Optimization*, London, Academic Press, 1972, pp. 191-201.

A.G. Deczky (1969), "General expression for the group delay of digital filters," *Electronics Letters*, vol. 5, no. 25, pp. 663-665.

A.G. Deczky (1972), "Synthesis of recursive digital filters using the minimum p-error criterion," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, no. 4, pp. 257-263.

A.G. Deczky (1973), "Computer-aided synthesis of digital filters in the frequency domain," *Swiss Federal Institute of Technology, Dissertation no. 4980*, Juris Druck & Verlag, Zurich, 1973.

J.E. Dennis (1973), "Some computational techniques for the nonlinear least squares problem," in G.D. Byrne and C.A. Hall (eds.) *Numerical Solution of Systems of Nonlinear Algebraic Equations*, New York, Academic Press, 1973, pp. 157-183.

S.W. Director and R.A. Rohrer (1969a), "Automated network design - the frequency domain case," *IEEE Transactions on Circuit Theory*, vol. CT-16, pp. 330-337.

S.W. Director and R.A. Rohrer (1969b), "The generalized adjoint network and network sensitivities," *IEEE Transactions on Circuit Theory*, vol. CT-16, pp. 318-323.

L.C.W. Dixon and M.C. Biggs (1970), "Meander, a Newton based procedure for N-dimensional function minimization," *Hatfield Polytechnic (U.K.) Numerical Optimization Centre; Technical Report no. 9.*

A.G. Evans and R. Fischl (1973), "Optimal least squares time domain synthesis of recursive digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-21, pp. 61-65.

D.S. Fensom, N.I. Smith and B.D. Ackland (1979), "GASP: a fast general purpose signal processor," *The Institution of Engineers, Australia; Electrical Engineering Transactions*, vol. EE-15, no. 1, pp. 26-30.

A. Fettweis (1971a), "Digital filter structures related to classical filter networks," *Archiv fur Elektronik und Ubértragungstechnik*, vol. 25, pp. 79-89.

A. Fettweis (1971b), "Some principles of designing digital filters imitating classical filter structures," *IEEE Transactions on Circuit Theory*, vol. CT-18, pp. 314-316.

A. Fettweis (1971c), "A general theorem for signal-flow networks, with applications," *Archiv fur Elektronik und Ubertragungstechnik*, vol. 25, pp. 557-561.

A. Fettweis, H. Levin and A. Sedlmeyer (1974), "Wave digital lattice filters," *International Journal of Circuit Theory and Applications*, vol. 2, pp. 203-211.

A.V. Fiacco and G.P. McCormick (1968), *"Nonlinear Programming: Sequential Unconstrained Minimization Techniques,"* New York, John Wiley & Sons., 1968.

J.L. Flanagan (1972), *"Speech Analysis, Synthesis and Perception,"* 2nd edition, New York, Springer-Verlag, 1972.

R. Fletcher (1970), "A new approach to variable metric algorithms," *Computer Journal*, vol. 13, pp. 317-322.

R. Fletcher (1972a), "Conjugate-direction methods," in W. Murray (ed.) *Numerical Methods for Unconstrained Optimization*, London and New York, Academic Press, 1972.

R. Fletcher (1972b), "An algorithm for solving linearly constrained optimization problems," *Mathematical Programming*, vol. 2, pp. 133-165.

R. Fletcher (1976), "Factorizing symmetric indefinite matrices," *Linear Algebra and its Applications*, vol. 14, pp. 257-272.

R. Fletcher and T.L. Freeman (1977), "A modified Newton method for minimization," *Journal of Optimization Theory and Applications*, vol. 23, no. 3, pp. 357-372.

R. Fletcher and M.J.D. Powell (1963), "A rapidly convergent descent method for minimization," *Computer Journal*, vol. 6, pp. 163-168.

R. Fletcher and C.M. Reeves (1964), "Function minimization by conjugate gradients," *Computer Journal*, vol. 7, pp. 149-154.

P.E. Gill and W. Murray (1974), "Newton-type methods for unconstrained and linearly constrained optimization," *Mathematical Programming*, vol. 7, pp. 311-350.

P.E. Gill and W. Murray (1978), "Design and implementation of software for unconstrained optimization," in H.J. Greenberg (ed.) *Design and Implementation of Optimization Software*, Alphen aan den Rijn, Sijthoff & Noordhoff, 1978.

B. Gold and T. Bially (1973), "Parallelism in fast Fourier transform hardware," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-21, no. 1, pp. 5-16.

B. Gold and K.L. Jordan (1969), "A direct search procedure for designing finite duration impulse response filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-17, pp. 33-36.

B. Gold, I.L. Lebow, P.G. McHugh and C.M. Rader (1971), "The FDP, a fast programmable signal processor," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 33-38.

D. Goldfarb (1970), "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, pp. 23-26.

S.M. Goldfeld, R.E. Quandt and H.F. Trotter (1966), "Maximization by quadratic hill-climbing," *Econometrica*, vol. 34, no. 3, pp. 541-551.

A.A. Goldstein and J.F. Price (1967), "An effective algorithm for minimization," *Numerische Mathematik*, vol. 10, pp. 184-189.

R.E. Gomory (1958), "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol 64, pp 275-278.

B.S. Gottfried and J. Weisman (1973) *Introduction to Optimization Theory*, Englewood Cliffs, N.J., Prentice-Hall Inc., 1973.

J. Greenstadt (1967), "On the relative efficiencies of gradient methods," *Mathematics of Computation*, vol. 21, pp. 360-367.

A. Hadjifotiou and D.G. Appleby (1976), "Design of digital filters with severely quantized coefficients," *The Radio and Electronic Engineer*, vol. 46, no. 1, pp. 23-28.

R. Hastings-James and S.K. Mehra (1977), "Extensions of the Pade-approximant technique for the design of recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 6, pp. 501-509.

O. Herrmann (1970), "Design of nonrecursive digital filters with linear phase," *Electronics Letters*, vol. 6, pp. 328-329.

O. Herrmann and W. Schuessler (1970), "On the accuracy problem in the design of nonrecursive digital filters," *Archiv fur Elektronik und Ubertragungstechnik*, vol. 24, pp. 525-526.

D.M. Himmelblau (1972), "A uniform evaluation of unconstrained optimization techniques," in F.A. Lootsma (ed.) *Numerical Methods for Nonlinear Optimization*, London, Academic Press, 1972.

E. Hofstetter, A.J. Oppenheim and J. Siegel (1971), "A new technique for the design of nonrecursive digital filters," in *Proc. 5th Annual Princeton Conference on Information Sciences and Systems*, Princeton, NJ., March 1971, pp. 64-72.

R. Hooke and T.A. Jeeves (1961), "Direct-search" solution of numerical and statistical problems," *Journal of the Association for Computing Machinery*, vol. 8, pp. 212-229.

J.V. Hu (1971), "Frequency domain design of two-dimensional finite impulse response digital filters," in *Proc. of Two-dimensional Digital Signal Processing Conf.*, University of Missouri, Columbia, October 1971.

S.Y. Hwang (1974), "On optimization of cascade fixed-point digital filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-21, pp. 163-166.

L.B. Jackson (1970a), "On the interaction of roundoff noise and dynamic range in digital filters," *Bell System Technical Journal*, vol. 49, pp. 159-184.

L.B. Jackson (1970b), "Roundoff noise analysis for fixed-point digital filters realized in cascade or parallel form," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-18, pp. 107-122.

L.B. Jackson and S.L. Wood (1978), "Linear prediction in cascade form," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, no. 6, pp. 518-528.

A. Jones (1970), "SPIRAL - anew algorithm for nonlinear parameter estimation using least squares," *Computer Journal*, vol. 13, no. 3, pp. 301-308.

G.A. Jullien and M.A. Sid-Ahmed (1974), "Stability constraints used in computer-aided design of recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-22, no. 2, pp. 153-158.

J.F. Kaiser (1965), "Some practical considerations in the realization of linear digital filters," in *Proc. 3rd Annual Allerton Conference on Circuit and System Theory*, Monticello, Il., pp. 621-633.

R.E. Kalman (1958), "Design of a self-optimizing control system," *Transactions of the American Society of Mechanical Engineers*, vol. 80, pp. 468-478.

R.E. King and G.W. Condon (1973), "Frequency domain synthesis of a class of optimum recursive digital filters," *International Journal of Control*, vol. 17, no. 3, pp. 497-509.

J.B. Knowles and E.M. Olcayto (1968), "Coefficient accuracy and digital filter response," *IEEE Transactions on Circuit Theory*, vol. CT-15, no. 1, pp. 31-41.

D.M. Kodek (1980), "Design of optimal finite-wordlength FIR digital filters using integer programming techniques," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-28, pp. 304-308.

G.E. Kopec, A.V. Oppenheim and J.M. Tribolet (1977), "Speech analysis by homomorphic prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 1, pp. 40-49.

H-K. Kwan (1979), "On the problem of designing IIR digital filters with short coefficient wordlengths," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, no. 6, pp. 620-624.

A.H. Land and A. Doig (1960), "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497-520.

K. Levenberg (1944), "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164-168.

B. Liu and A. Peled (1975), "Heuristic optimization of the cascade realization of fixed-point digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-23, no. 5, pp. 464-473.

G.B. Lockhart (1975), "Filters with finite duration impulse responses," in R.E. Bogner and A.G. Constantinides (eds.) *Introduction to Digital Filtering*, London, John Wiley and Sons, Ltd., 1975.

J. Makhoul (1975), "Linear prediction - a tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561-580.

G.A. Maria and M.M. Fahmy (1974), "l_p approximation of the group delay response of one- and two-dimensional filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-21, no. 3, pp. 431-436.

J.D. Markel and A.H. Gray, jr. (1976), *"Linear Prediction of Speech,"* Berlin, Heidelberg and New York, Springer-verlag, 1976.

D.W. Marquardt (1963), "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431-441.

A. Matthews and D. Davies (1971), "A comparison of modified Newton methods for unconstrained optimization," *Computer Journal*, vol. 14, no. 3, pp. 293-294.

G.A. Matthews, F. Davis, J.C. McKee, R.K. Cavin and G.M. Sibbles (1963), "Approximating transfer functions by linear programming," in *Proc. 1st Annual Allerton Conference on Circuit and System Theory*, Monticello, Il., pp. 191-204.

J.J. McKeown (1975), "Specialised versus general-purpose algorithms for minimizing functions that are sums of squared terms," *Mathematical Programming*, vol. 9, pp. 57-68.

R.R. Meyer (1970), "Theoretical and computational aspects of nonlinear regression," in J. Rosen, O. Mangasarian and K. Ritter (eds.) *Nonlinear Programming*, New York, Academic Press, 1970.

G. Miller (1973), "Least-squares rational z-transform approximation," *Journal of the Franklin Institute*, vol. 295, no. 1, pp. 1-7.

S.K. Mitra and R.J. Sherwood (1974), "Estimation of pole-zero displacements of a digital filter due to coefficient quantization," *IEEE Transactions on Circuits and Systems*, vol. CAS-21, no. 1, pp. 116-124.

W. Murray (1972a), "Second Derivative methods," in W. Murray (ed.) *Numerical Methods for Unconstrained Optimization*, London and New York, Academic Press, 1972.

B.A. Murtagh and R.W.H. Sargent (1970a), "A constrained minimization method with quadratic convergence," in R. Fletcher (ed.) *Optimization*, London and New York, Academic Press, 1970.

B.A. Murtagh and R.W.H. Sargent (1970b), "Computational experience with quadratically convergent minimization methods," *Computer Journal*, vol. 13, pp. 185-194.

National Physical Laboratory (U.K.) (1961) *"Modern Computing Methods" (Notes on Applied Science, no. 16)* London, HMSO, 2nd edition, 1961.

J.A. Nelder and R. Mead (1965), " A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308-313.

J.P.Olive (1971), "Automatic formant tracking by a Newton-Raphson technique," *Journal of the Acoustical Society of America*, vol. 50, no. 2 (part 2), pp.661-670.

A.V. Oppenheim (1969), "Speech analysis-synthesis system based on homomorphic filtering," *Journal of the Acoustical Society of America*, vol. 45, no. 2, pp. 458-465.

A.V. Oppenheim, R.W. Schafer and T.G. Stockham (1968), "Nonlinear filtering of multiplied and convolved signals," *Proceedings of the IEEE*, vol. 56, no. 8, pp. 1264-1291.

A.V. Oppenheim and C.J. Weinstein (1972), "Effects of finite register length in digital filtering and the fast Fourier transform," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 957-976.

T.W. Parks and J.H. McClellan (1972), "Chebyshev approximation for nonrecursive digital filters with linear phase," *IEEE Transactions on Circuit Theory*, vol. CT-19, pp. 189-194.

J.D. Pearson (1969), "Variable metric methods of minimization," *Computer Journal*, vol. 12, pp. 171-178.

G. Peckham (1970), "A new method for minimizing a sum of squares without calculating gradients," *Computer Journal*, vol. 13, pp. 418-420.

M.J.D. Powell (1964), "An efficient method of finding the minimum of a function of several variables without calculating derivatives," *Computer Journal*, vol. 7, pp. 155-162.

M.J.D. Powell (1965), "A method for minimizing a sum of squares without calculating derivatives," *Computer Journal*, vol. 7, pp. 303-307.

M.J.D. Powell (1970), "A FORTRAN subroutine for solving systems of nonlinear algebraic equations," in P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, London, Gordon & Breach, 1970. pp. 115-161.

M.J.D. Powell (1972), "Problems related to unconstrained optimization," in W. Murray (ed.) *Numerical Methods for Unconstrained Optimization*, London and New York, Academic Press, 1972.

L.R. Rabiner (1972), "Linear program design of finite impulse response (FIR) digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, pp. 280-288.

L.R. Rabiner, B. Gold and C.A. McGonegal (1970), "An approach to the approximation problem for nonrecursive digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-18, pp. 83-106.

L.R. Rabiner, N.Y. Graham and H.D. Helms (1974), "Linear programming design of IIR digital filters with arbitrary magnitude function," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-22, pp. 117-123.

L.R. Rabiner and R.W. Schafer (1971), "Recursive and non-recursive realizations of digital filters designed by frequency sampling techniques," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-19, pp. 200-207.

L.R. Rabiner and R.W. Schafer (1978), *Digital Processing of Speech Signals*, Englewood Cliffs, NJ, Prentice-Hall, 1978.

C.M. Rader and B. Gold (1967), "Effect of parameter quantization on the poles of a digital filter," *Proceedings of the IEEE*, vol. 55, pp. 688-689.

M.H. Rahman and M.M. Fahmy (1975), "On pole distribution in digital filters," *International Journal of Circuit Theory and Applications*, vol. 3, pp. 95-100.

J.B. Rosen (1960), "The gradient projection method for non-linear programming; part I - linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 181-217.

H.H. Rosenbrock (1960), "An automatic method for finding the greatest or least value of a function," *Computer Journal*, vol 3, pp. 175-184.

R.W.H. Sargent and D.J. Sebastian (1972), "Numerical experience with algorithms for unconstrained minimization," in F.A. Lootsma (ed.) *"Numerical Methods for Nonlinear Optimization,"* London and New York, Academic Press, 1972.

E.R. Schulz (1968), "Estimation of pulse transfer function parameters by quasilinearization," *IEEE Transactions on Automatic Control,* vol. AC-13, pp. 424-426.

R.E. Seviora and M. Sablatash (1971), "A Tellegen's theorem for digital filters," *IEEE Transactions on Circuit Theory,* vol. CT-18, no. 1, pp. 201-203.

J.L. Shanks (1967), "Recursion filters for digital processing," *Geophysics,* vol. 32, pp. 33-51.

D.F. Shanno (1970), "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation,* vol. 24, pp. 647-656.

N.I. Smith (1977), "Optimum quantized coefficients for digital filters," in *Convention Digest, Institution of Radio and Electronics Engineers, Australia; 16th International Convention and Exhibition,* Melbourne, 8-12 August 1977, pp. 289-291.

N.I. Smith (1979), "A random-search method for designing finite-wordlength recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. ASSP-27, no. 1, pp. 40-46.

W. Spendley, G.R. Hext and F.R. Himsworth (1962), "Sequential application of simplex designs in optimization and evolutionary operation," *Technometrics*, vol. 4, pp. 441-461.

K. Steiglitz (1970), "Computer-aided design of recursive digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-18, pp. 123-129.

K. Steiglitz (1971), "Designing short-word recursive digital filters," in *Proc. 9th Annual Allerton Conference on Circuit and System Theory*, Monticello, Il., pp. 778-788.

K. Steiglitz (1977), "On the simultaneous estimation of poles and zeros in speech analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 3, pp. 229-234.

K. Steiglitz and B. Dickinson (1977), "The use of time domain selection for improved linear prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 1, pp. 34-39.

K. Steiglitz and L.E. McBride (1965), "A technique for the identification of linear systems," *IEEE Transactions on Automatic Control*, vol. AC-10, pp. 461-464.

G.W. Stewart (1967), "A modification of Davidon's minimization method to accept difference approximations of derivatives," *Journal of the Association for Computing Machinery*, vol. 14, pp. 72-83.

M. Suk and S.K. Mitra (1972), "Computer-aided design of digital filters with finite wordlengths," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, pp. 356-363.

B.O.H. Tellegen (1952), "A general network theorem, with applications," *Philips Research Report*, vol. 7, pp. 259-269.

P. Thajchayapong and P.J.W. Rayner (1973), "Recursive digital filter design by linear programming," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-21, no. 2, pp. 107-112.

P. Thajchayapong and P.J.W. Rayner (1974), "Design of recursive digital filters approximating to arbitrary prescribed magnitude responses," *Electronics Letters*, vol. 10, no. 9, pp. 159-161.

W.Wegener (1978), "On the design of wave digital filters with short coefficient wordlengths and optimal dynamic range," *IEEE Transactions on Circuits and Systems*, vol. CAS-25, no. 12, pp. 1091-1098.

J.H. Wilkinson (1960), "Householder's method for the solution of the algebraic eigenproblem," *Computer Journal*, vol. 3, pp. 23-27.

J.H. Wilkinson (1965), *"The Algebraic Eigenvalue Problem,"* London, Oxford University Press, 1965.

W.I. Zangwill (1967), "Minimizing a function without calculating derivatives," *Computer Journal*, vol. 10, pp. 293-296.

PUBLICATIONS AND CONFERENCE PRESENTATIONS

relating to the subject-matter of this thesis, by N.I. Smith and co-authors.

1. N.I. Smith (1977), "Optimum quantized coefficients for digital filters," in *Convention Digest, Institution of Radio and Electronics Engineers, Australia; 16th International Convention and Exhibition,* Melbourne, 8-12 August 1977, pp. 289-291.

2. D.S. Fensom, N.I. Smith and B.D. Ackland (1978), "GASP: a fast general purpose signal processor," in *Preprints of papers, The Institution of Engineers, Australia; Conference on Computers in Engineering,* Canberra, 23-25 August, 1978, pp. 81-85.

3. N.I. Smith (1979), "A random-search method for designing finite-wordlength recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. ASSP-27, no. 1, (February 1979), pp. 40-46.

4. D.S. Fensom, N.I. Smith and B.D. Ackland (1979), "GASP: a fast general purpose signal processor," *The Institution of Engineers, Australia; Electrical Engineering Transactions,* vol. EE15, no. 1, (1979), pp. 26-30.

Items 1 and 3 are reproduced as part of chapter six of this thesis (pp. 397-406) and item 4 as part of chapter seven (pp. 427-431).