# MODELLING OF REGIONAL NETWORKS FOR MINIMUM COST

# MODELLING OF REGIONAL NETWORKS
# FOR MINIMUM COST

by

RAY HERBERT TUFGAR, B.SC. ENGR.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Engineering
McMaster University
February 1978

MASTER OF ENGINEERING (1978)
(Civil Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario
Canada

TITLE: MODELLING OF REGIONAL NETWORKS FOR MINIMUM COST

AUTHOR: Ray H. Tufgar, B,Sc. Engr. (McMaster University)

SUPERVISOR: Dr. Alan A. Smith

NUMBER OF PAGES: xii, 401

ABSTRACT:

A computer model is developed for the solution of regional network systems on a minimum cost basis. Different network problem types are identified to define the scope of the model. Generally, the problem involves the determination of the optimum conveyance schedule required to supply a set of consumer nodes with a commodity or public service provided by one or more processing centres (i.e. water supply, solid waste or wastewater collection networks). The system costs include costs incurred due to conveyance of material and the processing of that material; both of these cost components exhibit economies of scale and generally lead to the centralization of processing.

A mathematical statement of the problem is developed which is applicable to all network types, can be utilized easily and efficiently in a digital computer and facilitates the use of a variety of optimization routines. A number of algorithms, ranging from linear approximation to nonlinear gradient search routines, are investigated for possible inclusion in the model with the advantages and disadvantages of each being identified.

A modular package, NETSOL, is developed which facilitates the use of alternate optimization routines and allows for the inclusion of complex design functions in the computation of system costs. An interactive command structure permits the user to modify the network system parameters thereby combining intuitive design with the capacity to select an optimal solution from a large number of alternatives. Thus a model is obtained which is flexible enough to answer many of the questions that arise in network problems and also determine the optimum.

In view of the disadvantages found in the existing optimization techniques (e.g. convergence to local minima), the properties of the network problems are investigated in detail to isolate any special characteristics. For a system involving separable concave cost functions, the minimum cost solution to a network problem lies at one of the vertices formed by the problem constraints. A new solution algorithm, HYVRST, is developed which takes advantage of this important property utilizing a direct search technique. This results in an efficient and stable algorithm with good convergence properties. A number of examples are presented to test the optimization algorithms and demonstrate the usefulness of the NETSOL package in solving regional network problems.

# ACKNOWLEDGMENTS

I would like to express my appreciation to the National Research Council of Canada for the financial support made available to me during the course of this study.

I owe special thanks to the persons who aided me in "getting together" this dissertation. To my wife, Gloria, who spent considerable time in typing drafts and the final version presented here. To both my wife and parents, Mr. and Mrs. C.H. Tufgar for their patience and moral support.

I am especially appreciative to my supervisor, Dr. A.A. Smith, the person who made it all possible. His guidance and help have been invaluable in this study.

## TABLE OF CONTENTS

# LIST OF FIGURES

FIGURE NUMBER

FIGURE
NUMBER

# LIST OF TABLES

# CHAPTER 1

# NETWORK SYSTEMS

## 1.1 Introduction

In recent years, the classical transportation problem has become a general classification for many problems exhibiting certain similar properties. The transportation problem arises when an optimal conveyance schedule is required to convey a commodity from one or more sources (supply nodes), where the commodity is made available, to a number of sinks (demand nodes). The cost is defined by an objective function which comprises the sum of the conveyance costs and costs generated at the supply or processing nodes. The amounts of the commodity available at the supply nodes and required at the demand nodes are generally fixed or constrained within limits. At times, special problems arise in which the conveyance link capacities impose a constraint. The problems encountered are generally economic in nature since the goals or objectives are concerned with monetary costs and benefits, while the network system itself is subject to constraints of a technological or physical form.

The purpose of this study is to develop a package of subroutines which can be used to construct econometric models useful in solving transportation (network) problems. The different network problems which are to be considered for model applications are first covered in detail to identify common characteristics. A general mathematical statement of the problem is then developed and an optimization technique adopted for the development and testing of an initial model. Alternative solution algorithms are investigated for possible adoption in the package. Further testing is carried out to determine the general applicability of the package and its usefulness in problem solving. Consideration is also given to work already carried out in the field of econometric modeling.

The classical transportation problem has a long history. In 1939, L.V. Kantorovich (21) identified a class of problems closely related to transportation problems which applied typically to the allotment of labour in industrial plants. A great amount of research has been devoted to the development of solution algorithms for application to transportation problems, especially those in the linear programming field ( 9 ). At first, most solution methods were designed to facilitate hand calculations but with the advent of computer applications, solution methods are now designed to take advantage of the speed and accuracy of digital computers. Linear graph theory has been very helpful in identifying the special characteristics of transportation problems or equivalent network flow problems which facilitate the development of standardized solution techniques (18). T.C. Koopmans ( 9 ) pioneered the interpretation of the properties of optimal and nonoptimal solutions with respect to the linear graph representing a network.

Linear programming problems are generally solved by using a simplex or revised simplex technique. The network problems considered in this study do not correspond strictly to the class of transportation problems as used in some texts due to the existence of nonlinear instead of linear objective functions. Nonlinear programming problems are generally subdivided into problem types, solvable by different specialized techniques or , if possible, the problem is approximated to a linear one so that the more efficient linear methods can be used. Methods of solution are presented later in this study.

## 1.2 General Problem Types Investigated

Although many different problem types come under the classification of a transportation problem, certain network problems with a common characteristic have been isolated and provide the basis for this study. Frequently, the design engineer encounters the problem of designing a scheme to provide a group of communities with a public utility service or commodity. When considering a number of regionally located communities, it is commonly found that it is more economical to have a small number or only one large service centre which is centrally located. The savings in the system cost obtained through the economies of scale realized with a large central service centre may or may not offset the additional costs incurred in transporting the commodity. It is of interest to find the optimal method of providing the communities with a public utility. In finding the optimal system, it is necessary to determine the size and location of the service centres (or processing nodes) and to find the optimal method (i.e. network of transportation links) of conveying the commodity. Secondary areas of interest to the planner include investigating the economic effect of removing some of the marginal communities or some of the feasible distribution links. Decisions of this type are best made through the use of an econometric model which is capable of choosing the most economical system design from all of the possible alternatives.

There are various problems which fit into the classification of a network flow problem and exhibit the properties of economies of scale with respect to the costs involved. Each major problem type considered is discussed in detail to present their main characteristics and facilitate the identification of the required properties for a useful econometric model.

### 1.2.1 Water Supply System

The growth or planned expansion of a community, resulting in an increased demand of potable water may make it necessary to utilize new sources of water supply. While doing this it may also appear feasible to investigate the practicality of including other communities in a regional supply network. A typical example of a water supply network is illustrated in figure 1.1. The supply nodes consist of any type of water source such as reservoirs, lakes, rivers or wells, any of which may or may not require treatment (processing) depending upon the quality of water required. There is a cost associated with each supply node representing the cost of collecting and treating the water and which varies nonlinearly with the volume rate of supply from that source. The demand nodes are made up of the different communities which can feasibly be included in the supply system scheme. In some cases it may be desirable to subdivide some of the communities into a number of demand nodes to conform to the geographical layout of the region or to meet design requirements. For instance, each community can be subdivided corresponding to different pressure districts. In this case, the required pressure or state at each node is determined by the minimum service pressure in the district. It is worthwhile to note that in the design of a water distribution system within a community, the pressure at the supply point is a variable which can lie between maximum and minimum limits; the maximum limit being determined by considerations of leakage or damage to fittings, and the minimum being the lowest useful service pressure. In the model presented here, the pressure elevation is set and the design variable is the flow between nodes. The introduction of the node pressures as another form of design variable, results in an entirely different type of problem which is discussed in more detail in a later section (see Chapter 6, page193). The links connecting the nodes include all technically and economically feasible pipelines. The supply and demand nodes may be linked either directly in a one-to-one correspondence or indirectly where transshipment of material from one node to another occurs via intermediate nodes. The flows in the links and, indirectly, the quantities of water processed are viewed as the design variables.

The decisions which are normally required in the design of a regional water supply network are:

1) Which of the possible sources should be included in the distribution system and at which nodes should the treatment facilities be located?

2) Which of the communities should be included in the network and which should rely on external or local sources?

3) What is the most economical pipeline layout required to supply the communities and what flow scheme should be used?

System costs, which form the basis for making the required decisions, arise both from processing the water and from conveying it to the demand nodes. The <u>processing costs</u> comprise capital expenditures on intakes, treatment works and ancillary equipment in addition to annual costs for treatment, maintenance and sinking funds for plant replacement. Experience has shown that with controlling factors such as raw water quality, impounding, land costs, construction costs, etc. held constant, the costs involved in constructing and operating water treatment works generally exhibit economies of scale with respect to plant capacity for the quantity of water handled (2). A typical plot demonstrating the economies of scale present is illustrated in figure 1.2. The cost curve is concave and monotonic in shape and generally displays the characteristics of a fixed charge implying a minimum charge limit. The minimum charge limit is a result of the property that there is a lower limit of feasibility on the size of treatment works. Transportation costs arise from capital costs on pipelines and pump stations as well as annual costs for operation, maintenance and sinking funds for the replacement of plant and other items with a relatively short life. The transportation costs are also nonlinear and exhibit economies of scale. The fixed charge component is usually negligible in transportation. A typical transportation cost curve is shown in figure 1.3. There is, therefore, a cost function associated with each variable representing the rate of flow through a link and with the amount of material processed at a node. The resulting family of curves are used in combination with the design variables (flowrates) to make up the objective function of the network problem.

The restriction of continuity being established at the supply and demand nodes results in the generation of a set of feasibility constraints on the solution to the problem. To find the optimal network policy (or the least costly method of supplying the demand nodes), the objective function is minimized subject to maintaining feasibility in the constraints.

### 1.2.2 Sewerage Networks

Increasing water quality restrictions along with community growth result in an ever-increasing requirement for the treatment of sanitary sewage and wastewater. Through economies of scale, it seems logical that in a regional system, savings would arise through the use of a central or series of centralized treatment plants to process the wastes of the communities. Depending upon the characteristics of the region, the economies of scale obtained through centralization may offset additional conveyance costs.

A wastewater collection network is similar to a regional water supply network in that it supplies a public utility (i.e. wastewater disposal service) to a region. The difference lies in the fact that it is a collection network whereas water supply networks involve distribution. In sewerage networks, the "load" imposed on the system by each community takes the form of an input of wastewater described as a flowrate and averaged over some suitable time interval. Directions of flow are therefore reversed and the central processing nodes thus become points of abstraction from the system. The use of the terms source and sink may therefore lead to confusion and are avoided.

In accordance with today's technology, most processing sites are normally close to a river or lake. The treatment capacity, or demand, of the processing nodes is normally regarded as infinite in relation to the amount of wastewater produced. In special cases, the limiting size of the treatment works due to land availability or capacity of the receiving waters affects the system design. A method of handling this special case is discussed in Chapter 7 (page 251).

In a regional wastewater treatment network, some of the wastewater producing communities are also possible sites for treatment, an aspect which is considered in the econometric model design. The links connecting the supply and processing nodes include all technically and economically feasible pipeline locations. Transshipment is possible, hence, wastewater can be routed to the processing centres directly or through interconnected nodes.

The decisions normally required in the design of a regional wastewater treatment network are:

1) Out of all of the possible locations for treatment facilities, which should be included in the optimal system design?

2) Which wastewater producers should be included in the network system?

3) What is the additional cost imposed upon the community (or region), if any, of having the treatment works centralized in a chosen location?

4) What is the most economical sewerage layout required to provide conveyance to the processing centres and what flow scheme should be used?

Centralization of wastewater treatment for a regional system leads to certain environmental considerations. Centralization of treatment leads to a centralized discharge of treated effluent into a receiving water which then confines the degradation of water quality to one area. The degradation of water quality results from the policy of utilizing the assimilative capacity of a receiving water body as much as possible as part of a treatment scheme in order to reduce costs. Lately, however, more effort has been made to improve the quality of the effluent discharged due to the increased awareness of our poor overall water quality. Recent studies ( 1 ) show that the environmental impact of effluent discharge can be reduced by decentralization of treatment so that instead of having a large discharge at one point, there are a number of smaller discharges at different locations. This technique, however, utilizes the assimilative capacity more extensively and a lower level of overall treatment can be used and still meet quality standards. This suggests that the higher efficiency of one large treatment plant may not compensate for the extra cost involved in the higher level of treatment required to meet water quality standards. It is questionable as to whether or not the aspects of environmental impact can be quantified in the cost functions. However, the development of a model which has the capability of generating the costs of numerous network configurations quickly and efficiently facilitates the determination of the cost difference between solutions of differing environmental impact.

As with water supply networks, the costs incurred arise from processing and conveyance which are both nonlinear functions with respect to flow. The processing costs include capital expenditures on collection chambers, treatment works and ancillary equipment in addition to annual costs for treatment, maintenance and sinking funds. Although wastewater treatment differs in many ways from water treatment, the cost functions exhibit the same general property of economies of scale which produce a concave monotonic curve ( 2 , 30 ). The cost functions also have a fixed charge representing a minimum charge, as found for water treatment. The transportation costs originate from capital expenditures on sewers and pump stations as well as annual costs for operation, maintenance and sinking funds. The transportation costs are very similar to those for water supply networks, however, the wastewater collection systems are generally gravity fed instead of being pressurized by booster pumps. Pumping stations may be required, however, depending upon the geographical configuration of the region. An important difference from water supply networks is that the pressure head at each node is either fixed or constrained within quite narrow limits since the flow occurs with a free surface and conduits are located close to the ground surface.

Since both water supply and wastewater collection networks have the same nonlinear network problem properties, the main difference between the two lies in the fact that one is a distribution network and one a collection network which results in a difference in the constraint formulations. The constraint differences and allowances necessary in the mathematical statement are discussed in a later section (see Chapter 2, page 34).

### 1.2.3 Solid Waste Collection Networks

Sanitary landfill is fast becoming a very undesirable method of solid waste disposal due to problems of pollution and aesthetics. Although significantly more costly, there is an increasing use of incinerators and material reclamation centres for solid waste processing. Even in regions where alternate disposal methods are not yet technically or economically feasible, sanitary landfill sites are constantly being filled up and new sites must be found farther away from the waste producing community. The requirement of processing centres or new disposal sites raises the questions of where they should be located and along which routes the solid waste should be transported. The network in this problem is set up in a similar manner to the wastewater collection network; the waste producing communities are connected to all economically or technically feasible processing locations by a series of feasible conveyance links. The manner of processing is quite arbitrary as long as the processing costs can be expressed as a function of the volume rate of flow treated. The two basic ways of treating solid wastes, disposal and processing, both exhibit economies of scale (5). The costs for disposal by sanitary landfill are generally lower than those for incineration or material reclamation. The capital expenditures include the acquisition of suitable landfill sites, preparation and any equipment required. The maintenance of the site makes up the annual costs. The life of a sanitary landfill site is possibly quite short depending upon the volume available for waste and the rate of waste inflow. Therefore, depending upon the design stipulations, a sanitary landfill site has a limiting flow capacity obtained through a relation similar to:

$$\text{flow capacity} = \frac{\text{Available volume for waste}}{\text{Required life of disposal site}}$$

The designer must be aware of this when setting out the design requirements but as with wastewater collection networks, no special attention is given to this aspect in the model development and, theoretically, the processing capacity is assumed to be infinite for both landfill sites and solid waste treatment works.

With a waste processing centre, the capital costs originate from expenditures on waste holding and handling equipment, reclamation facilities and a means to dispose of the material not recycled such as incinerators or disposal sites. Annual costs are incurred through operating costs, maintenance and sinking funds. There are a number of intangible factors such as social, political or aesthetic impact which affect the value of a possible disposal or processing centre. For example, a newly located landfill site may lower the value of any residential property nearby or threaten the water quality of local wells. The designer has the option of building factors such as these into his cost functions, or dealing only with the tangible costs and considering the effect of the intangible factors when he is in the position of finding the costs of alternate solutions.

The links connecting the nodal points include all existing and proposed transportation routes. The concept of flow is slightly different in this problem due to the mode of conveyance. A change of terminology is required to describe the costs and capacities in the network; the basic concept of conveyance remains the same, however, that an input of capital and energy creates a capacity for a specific volume rate of flow. Whereas with pipe flow, the conveyance is simply described in terms of a volume rate of flow, additional parameters such as vehicle capacity, trip cycle time and hours of service per week are introduced when considering the conveyance of solid wastes. With controlling factors held constant, the nonlinear cost functions for solid waste conveyance exhibit economies of scale resulting in concave, monotonic curves with respect to the flowrate ( 5). The specific costs depend upon the mode of transport used and the conditions under which it is used. New transportation routes such as rail lines or roads may be required and the total or partial costs of these included in capital expenditures. The decision of either leasing or purchasing the transporting vehicles affects the allocation of these expenditures to either annual or capital costs. In the instance of using existing transportation means, the problem of exceeding the capacity may arise. If the capacity of an existing link is exceeded, extra costs through the delay times are incurred which may introduce a convex region in the upper part of the cost function. The consequences of this special case are discussed in Chapter 4 (page 133). Transshipment through a node is possible and may involve a cost due to changing from one mode of transportation to another. Methods of handling transshipment are discussed later in this study (see Chapter 2, page 30).

The design decisions which arise in the problem are identical to that of wastewater collection since both problems involve the task of collecting and processing wastes. The question of which processing locations and what type of waste process to use is more prominent in solid waste problems, however, due to the environmental attraction of the more costly solid waste reclamation centres.

## 1.3 Econometric Model Requirements

A similar type of network problem, not covered in the preceding sections, is investigated by Hitchcock (20). The problem he approaches involves the distribution of a product from a number of factories or warehouses to a collection of cities. In his mathematical statement of the problem, Hitchcock allows only for direct conveyance between the factories and cities, not permitting transshipment. The objective function used is the sum of products of constant coefficients and the flow variables. The constant coefficients represent the cost of conveyance per unit of material shipped thus depicting a linear relationship. Hitchcock includes a discussion on the geometric interpretation of the objective function and constraints. The solution method which he proposes foreshadows the simplex method used today. Although the application of properties inherent to transportation systems is limited, he does use them in developing a method of finding a starting solution.

E. Feldman, F.A. Lehrer and T.L. Ray identified a similar type of network problem in the development of a computer model (11). They investigate the problem of determining the number and sizes of warehouses required to supply a set of demand centres. It is assumed that the cost functions associated with transportation costs are linear and the warehouse cost functions are concave with respect to flow, displaying economies of scale. The authors do not look into any special properties of the problem presented but use existing solution methods in developing a computer algorithm to solve the problem. Different possible solution techniques are presented later in this study.

The three different problem types considered for solution in this study consist of a network made up of a set of nodal points interconnected by links over which material conveyance takes place with the links connecting the nodal points being directed arcs. The network is considered as being a continuous system in that the supply and demand quantities do not vary with time and hence the flow values are constant. Flow can exist in either direction in a link resulting in the possibility of either positive and negative flowrates or two nonnegative design variables per link and two associated cost functions $C_i(Q)$ and $C_{i+1}(Q)$, one for each direction. In general, $C_i(Q)$ and $C_{i+1}(Q)$ are not equal due to differences in lengths of the directed arcs, nodal states or user defined preferences. Feasibility restrictions require that continuity be established at each node with the general continuity statement appearing as:

Gross Supply = Production - Consumption + Intake - Output

Two basic types of networks exist, collective and distributive. In a distribution network, a public utility service is generated at the processing nodes and consumed at the demand nodes. Generally, there are upper limits on both the demand nodes and processing nodes but the system is not necessarily balanced, that is, the total supply available can exceed the total system demand. Collection networks consist of consumer nodes which generate the material to be processed at the processing nodes. This system is balanced since the amount of material generated equals the amount of material processed. Theoretically, there is no limit to the capacity of the processing nodes. The waste generating nodes, of course, have a finite outflow value associated with them. In this system, it is possible to have a node which acts both as a waste producing and processing node. In this type of situation, a possible location for a processing plant exists in a community which also generates the material to be processed. It can be appreciated, however, that if the costs of the processing plant are prohibitively high, the result is the rejection of the solution or solutions using that plant and the conveyance of the material from that community to another processing node.

Upon inspection of the possible network problems which are solvable by the econometric model developed, it becomes apparent that each problem has the same basic characteristic of conveying a commodity from a set of source nodes to a set of collection nodes. The costs involved arise from transportation of and processing the commodity. Most importantly, the cost functions involved in expressing these costs are concave, monotonic and separable, as shown later, this has a great bearing on the solution algorithms used. With respect to the development of a generalized mathematical statement of the problems, the most significant difference lies in the nodal types, dictated by the network classification (distribution or collection). To avoid any confusion later in the study, any nodal point is classified as either a processing or nonprocessing node. In each network type, the activity involved is:

| Network | Processing Node | Nonprocessing |
|---|---|---|
| Distribution | Nodes which produce the commodity for consumption. | Nodes which consume the commodity. |
| Collection | Nodes which process the commodity (i.e. waste processing node). | Nodes which generate the material to be processed. |

The only special case involved is in a collection network where a processing node can generate a material as well as have the ability to process it. The difference in nodal types must be handled by the algorithm which develops the mathematical statement of the problem.

The next step in the development of an econometric model is to adopt a consistent mathematical statement of the network problem which facilitates the use of common nonlinear optimization techniques. Further refinements on the model objectives may become apparent during the development of the mathematical statement and investigation of possible solution methods which is presented in the following chapters.

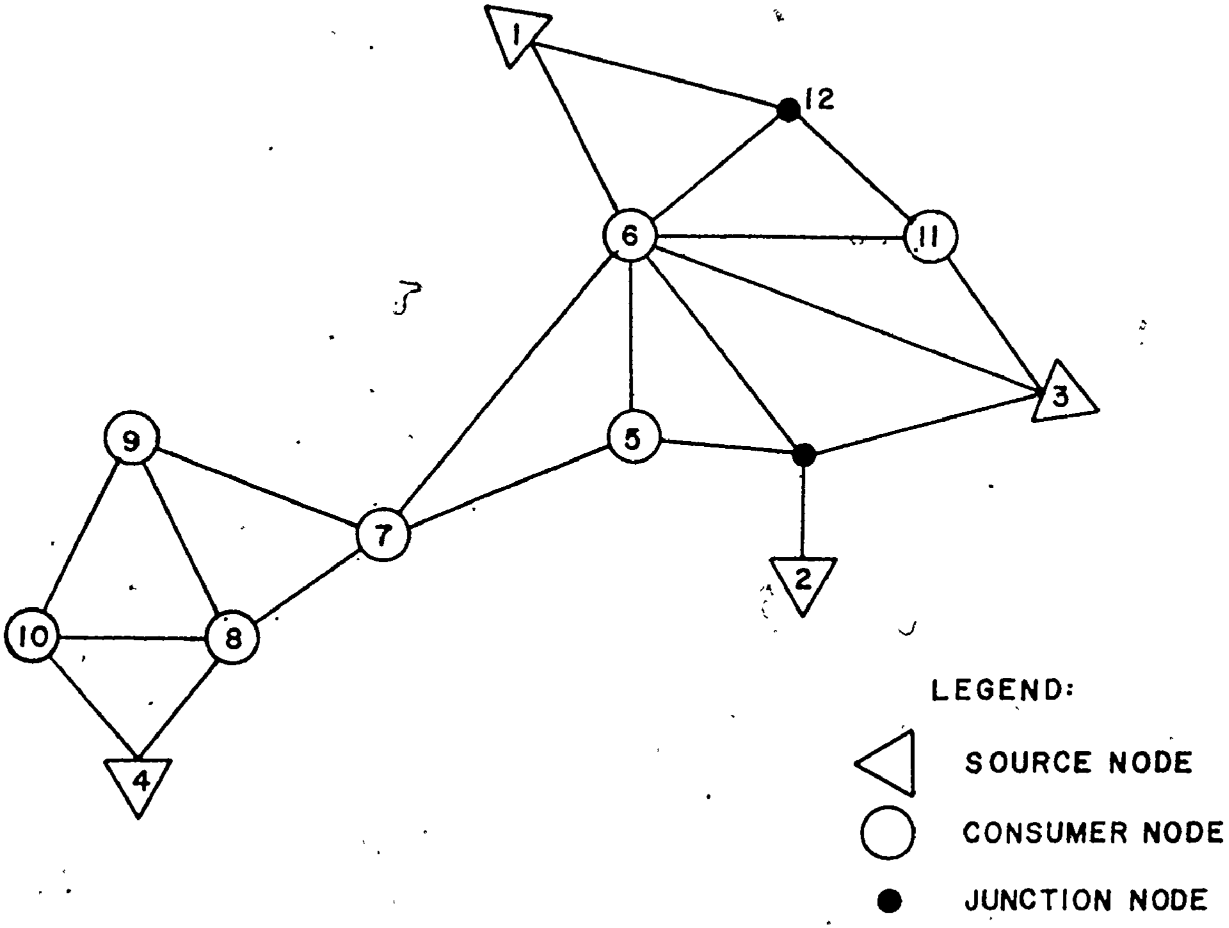


FIGURE 1·1 — TYPICAL REGIONAL WATER SUPPLY NETWORK

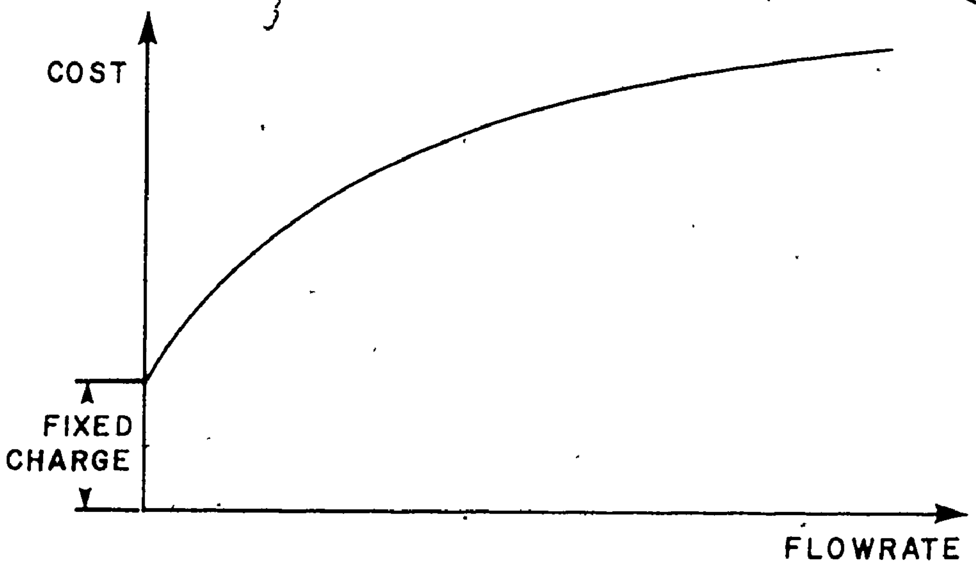


FIGURE 1·2 – TYPICAL PROCESSING COST CURVE EXHIBITING ECONOMIES OF SCALE

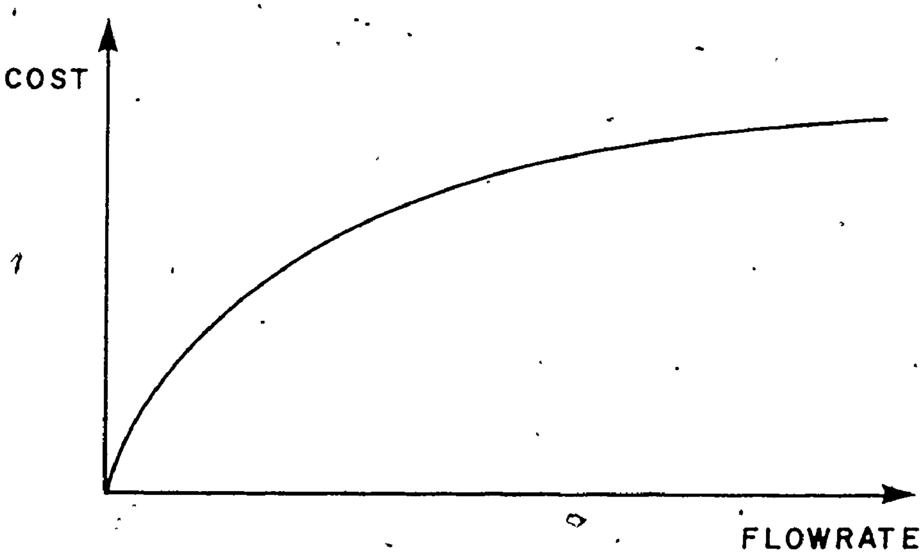


FIGURE 1·3 – TYPICAL TRANSPORTATION COST CURVE (CONCAVE)

# CHAPTER 2

# METHOD OF SOLUTION

## 2.1 Introduction

In constructing an econometric model, the first stage involves developing a mathematical statement of the problem which is applicable to all of the cases considered. Initially, the topological properties of the networks are investigated to identify common graphical properties and to adopt a universal graphical representation. This also aids in the identification of the minimum amount of data required to define the network. A generalized mathematical statement of the problem is then chosen for use in the econometric model. The mathematical statement is developed first for a distribution network to aid in understanding the terminology used. Special treatment is provided to allow for the differences between collection and distribution networks. Other possible forms of mathematical statements are also investigated for possible use. The properties of the problem solution are then investigated and illustrated through the use of a simplified problem.

## 2.2 Graphical Representation of Networks

When considering the development of a generalized model for the analysis of network problems, it is important to investigate the spatial structure, in terms of topologic and geometric components, of the networks involved. Obtaining the topological structure of a network involves reducing the network system to its basic elemental form of links and nodes. When reducing the network to its set of geographical locations intersected by a number of routes, the complex characteristics of the network come into view, for example, the lengths of the links, the degree of tortuosity of the links, the commodity being transported and the nodal state. The nodal state is a measure of nodal characteristics and has a direct effect on the conveyance of material and, hence, the cost of conveyance for any given flow. For instance, in a water distribution network, the pressure elevation of a node affects the pipe, and possibly pump size, required to convey water to another node. All of the network characteristics are important to the solution method used but they make the development of a generalized model very cumbersome. In order to investigate the basic spatial structure of networks, many of the characteristics are initially put aside to be used later in the model. The network is reduced to its spatial structure form through the use of linear graphing. The graph formed does not display the length or shape of a link or the size or characteristics of a node. Directed graphs denote the positive direction of flow and symbols are used to differentiate between supply and demand nodes. An example of reducing a topological network to its linear graph form is illustrated in figure 2.1 (a) and (b). Reducing topological networks to linear graphs in this manner may bring out similarities which are not recognizable at first. Since the characteristics defining the locations of the nodes and links are disregarded, the configuration of the linear graph can be changed to any shape required by the designer.

Linear graph theory provides methods which are useful in establishing standard methods of catagorizing networks. The two classifications which apply to the networks studied here are: (i) branching networks and (ii) circuit networks. A typical branching network is illustrated in figure 2.2 (a). The distinguishing characteristic of a branching network is its tree-like structure. Single path networks with no diverting links, such as a route from one node to another passing through other nodes, also fall under the classification of branching networks but have limited application. The most well known example of a branching network is a stream network which, from a topological viewpoint, has the simplest possible connectivity. In a distribution system comprised of a branching network, each processing node has a smaller branching network associated with it which has no more than one inflow link to any node. Likewise, in a collection system comprised of a branching network, each processing node has a smaller branching network associated with it which has no more than one outflow from any node. In general, optimal solutions take the form of a branching network in which each of the smaller branching networks may be connected at common nodes (see figure 2.2 (b)). Optimal solutions tend towards branching networks due to economies of scale which make it less costly to distribute a commodity, in the case of a distribution network, through one link rather than a number of links.

The initial regional networks as defined (i.e. figure 2.1 (b)) are generally structures with closed loops or circuits. Theoretically, they are nonplanar, that is, two links can cross without forming a junction node. Practically, however, it is not economical to have links crossing without a junction. When a network is reduced to a topological graph, three parameters can be identified:

| | | |
|---|---|---|
| (i) | The number of nodes | (N) |
| (ii) | The number of links | (L) |
| (iii) | The number of subgraphs or non-connected networks | (G) |

In figure 2.3, the parameter values are: N=9, L=8 (only 8 feasible links are shown) and G=2 (two nonconnecting networks are shown). The parameters can be used to form indices useful in establishing common yardsticks for comparing sets of networks. Haggett and Chorley summarize a number of indices useful in graph theory (18). Of these, the following are useful in this study:

1) Gamma Index $= \frac{L}{3(N-2)}$ (Planar graph)

$= \frac{2L}{N(N-1)}$ (Nonplanar graph)

2) Cyclomatic Number $= L - N + G$

3) Alpha Index $= \frac{L - N + G}{2(N-5)}$ (Planar graph)

$= \frac{L - N + G}{\frac{N(N-1)}{2} - (N-1)}$ (Nonplanar graph)

The Gamma index gives a measure of the degree of connectivity of the network. A value of zero denotes that no nodes are connected and a value of 1 denotes that all possible nodal pairs have a connecting link. An increasing Gamma index, in the range 0 to 1, therefore gives some insight as to the degree of connectivity and also some description of the increasing complexity. The Cyclomatic number yields the number of circuital paths in a network (i.e. in figure 2.3, there is 1 circuital path). The number of circuital paths is identical to the number of redundant links in the network, that is, the number of links above the minimum required to connect a set of nodes in a network. For instance, in figure 2.3, any link in the circuital path can be deleted and not disrupt the connectivity of the overall network. The cyclomatic number for a branching network is, therefore, equal to zero regardless of the number of subgraphs.

A more useful index of the connectivity of networks is the Alpha index, or, redundancy index. The Alpha index is the ratio of the number of the existing circuital paths in a network to the maximum number of possible circuital paths. Values of zero indicate a branching network and values of one indicate a fully connected network in which no further links may be added without duplicating existing links. An index such as this provides a measure of the degree of redundancy of an initial network. The index also serves to identify a branching network, and gives a measure of the relative complexity of a network. For a given set of nodes, an increase in the Alpha index represents an increase in the number of feasible solutions to be considered when searching for an optimum, and thus an increase in the computer time required.

## 2.3 Mathematical Statement

Although Chapter 1 describes both distribution and collection networks, the mathematical statement is presented in terms of a distribution network in order to simplify the understanding of the terms and quantities. Essentially the same procedure may be applied to collection networks, and minor differences are discussed towards the end of this chapter. Figure 2.4 illustrates a linear graph for a typical distribution network. The processing centres are, therefore, sources of supply and nonprocessing nodes are demand points. All possible links are included with two flow variables assigned to each link. The flow variables, $Q_i$, are marked on the diagram and the demand parameters, $D_i$, and supply parameters, $S_i$, are included alongside the nodes with which they are associated. The possibility of flow existing in either direction in a link is ensured by assigning two flow variables to each link, thus allowing a nonnegativity restriction on the flow. By using double flow variables, the number of flow variables are increased, but this method aids in the flexibility in choosing solution algorithms. The use of two flow values also facilitates the application of cost functions if the costs are not equal in each direction. Transshipment is allowed for through the inclusion of links connecting like nodes (links 1-2, 3-4, 4-5, 3-5). For example, the total demand at nodes 3, 4 and 5 might be routed through node 4. It may seem impractical to have a link between two supply nodes, however, if processing is carried out at a supply node, such as in the case of water treatment in a water distribution system, a saving in cost may be realized by linking the two sources and treating at the downstream supply node, due to economies of scale in the cost curves.

The objective function is designed to provide a measure of the system cost arising from the values assigned to the design variables. Actual system cost may include overheads and other cost components which are unaffected by the values adopted and these constant elements may be dropped from the objective function (e.g. the cost of balancing reservoirs may depend only on the total system demand, irrespective of how the demand is met). The objective function costs fall into two general catagories:

(1) Production costs at the processing centres.

(2) Transportation costs associated with the link over which conveyance of the commodity takes place.

Total costs for the network are formed from the sum of the cost functions evaluated in terms of the design (flow) variables, i.e.:

$$\text{Total Cost} = Z = \sum_{i=1}^{20} C_i(Q) \qquad \text{where:} \tag{2.1}$$

$C_i(Q)$ = cost function for the $i^{th}$ flow variable

It should be noted at this point that the functions C(Q) are assumed to include both transportation and processing costs. In equation 2.1, no provision is made to express the processing costs separately from the transportation costs and they are assumed to be computed jointly (i.e. processing costs distributed among the nonzero flow variables leaving these nodes). An expanded form of the objective function is presented later in this section where the processing costs are expressed separately. To obtain the optimal solution, this objective function must be minimized and to maintain feasibility, constraint equations must be satisfied which ensure mass balance at the nodes. Specific values of supply and demand are defined by a set of nodal stipulations. For convenience, the sign convention adopted for the constraints assumes that flow out of a node is positive and flow into a node is negative. The sign of the stipulation may be either positive or negative implying an addition to or abstraction from the system.

At the supply (processing) nodes, the total outflow of material from the node must not exceed the processing capacity, S. The constraint for node 1 in figure 2.4 is:

$$Q_1 - Q_2 + Q_3 - Q_4 + Q_5 - Q_6 + Q_{19} - Q_{20} \leq S_1 \tag{2.2}$$

The demand node constraints stipulate that the algebraic sum of the inflows to a demand node must equal the stipulation requirements. The constraint for node 3 in figure 2.4 is therefore:

$$-Q_1 + Q_2 - Q_7 + Q_8 - Q_{14} + Q_{13} - Q_{18} + Q_{17} = -D_3 \tag{2.3}$$

Slack variables are used to change any inequality constraints into an equality form. The inequality (2.2) is changed to an equation by adding the slack variable $Q_{21}$ to the left hand side:

$$Q_1 - Q_2 + Q_3 - Q_4 + Q_5 - Q_6 + Q_{19} - Q_{20} + Q_{21} = S_1 \tag{2.4}$$

Physically, the nonnegative quantity $Q_{21}$, represents the surplus of supply, $S_1$, over the net outflow from node 1. This quantity is used in the evaluation of the quantity of material processed at a supply node. The total quantity of material processed is:

$$\text{Net Outflow} = S_1 - Q_{21} \tag{2.5}$$

In special cases, the designer may wish to redefine the quantity of material processed at a supply node. For instance, if, as mentioned at the beginning of this section, transshipment takes place between two processing nodes, the user may wish to investigate any economies of scale available in processing at the downstream node. Therefore in figure 2.4, if the supplied commodity is conveyed from node 1 to node 2 where all of the processing is carried out for subsequent distribution to the nonprocessing nodes, the amount of material processed at node 1 is given by:

$$\text{Quantity Processed} = S_1 - Q_{21} - Q_{19} \qquad (2.6)$$

It is easily shown, however, that the relation (2.6) does not hold for some cases. If, for example, a water supply system is being analyzed, even though the water is transported from one supply node to another for treatment, there are still costs associated with the upstream node for amassing the water for conveyance. The apportioning of treatment among processing nodes is not identical for all problems, hence no attempt is made to accommodate this in a general sense. In the mathematical statement, it is therefore assumed that the material leaving the supply node is fully processed. Special methods are presented later in this study for handling cases such as this (see Chapter 4, page 135).

With slack variables providing a means by which the processing quantities can be quantified, the processing costs can be defined with the use of a processing cost function $CP_1(Q)$. The processing cost for node 1 in figure 2.4 is:

$$CP_1(S_1 - Q_{21}) \qquad (2.7)$$

The use of slack variables, therefore, provide a means of expressing the processing quantities and processing costs independently of the conveyance quantities. If this method was not followed, the processing costs would have to be proportioned among the conveyance costs of the flows leaving the supply nodes which is not as straightforward.

As an extension, artificial slack variables are introduced into the nonprocessing node constraint equations. For the example in figure 2.4, the constraint equation for demand node 3 now becomes:

$$-Q_1 + Q_2 - Q_7 + Q_8 - Q_{14} + Q_{13} - Q_{18} + Q_{17} - Q_{22} = -D_3 \qquad (2.8)$$

The nonnegative quantity $Q_{22}$ represents an additional inflow into node 3. Being artificial slack variables, they do not alter the mathematical statement or set of feasible solutions for the network problem, except in adding extra variables. Since these new variables provide an additional means of supplying the demand nodes, however, they must have a cost assigned to them. The artificial slack flow does not originate from any physical source in the network, hence, feasible solutions containing any artificial slack flows are undesirable. The costs assigned with the artificial slack variables are defined in the model to be prohibitively high, that is, higher than any conveyance costs for a design variable. These costs are assigned through the use of a penalty cost function. The artificial slack variables are not directly needed in the development of the mathematical statement but are required for use in a solution algorithm introduced later in the study to establish initial feasible solutions. Their importance is covered in detail when the solution algorithm is developed.

The set of simultaneous equations obtained from the equality constraints may be converted to the structural matrix:

$$\bar{A} \cdot \bar{Q} = \bar{B} \qquad (2.9)$$

where: $\bar{A}$ = a matrix of structural coefficients which take the value 0, 1 or −1 (N x NQ elements)

$\bar{Q}$ = a column vector of design variables (NQ elements)

$\bar{B}$ = a column vector of constraint stipulations (N elements)

N = the number of nodes in the network for which constraints are written

NQ = the total number of design (flow) plus slack plus artificial slack variables

A formal statement of the optimization problem takes the following form:

$$\underset{Q^*}{\text{Minimize}} \quad Z = \sum_{i=1}^{NQ-N} CT_i(Q) + \sum_{i=NQ-N+1}^{NQ-N+NCEN} CP_{i-NQ+N}(S_{i-NQ+N} - Q_i) \qquad (2.10)$$

$$+ \sum_{i=NQ-N+NCEN+1}^{NQ}$$

Subject to $\bar{A} \cdot \bar{Q} = \bar{B}$

and $Q_i \geq 0 \qquad i = 1, NQ$

$CT_i(Q)$ = Transportation cost function for the $i^{th}$ flow variable

$CP_i(\ )$ = Processing cost function for the $i^{th}$ node

NCEN = The number of processing nodes

$CPEN_i(Q)$ = Penalty cost function for the $i^{th}$ flow variable (normally a constant)

Table 2.1 illustrates the matrices formed when the problem of figure 2.4 is formulated as outlined above. For convenience, the processing node constraints appear first in the structural matrix. The constraints are contained in the matrices of structural coefficients and stipulations. Two auxiliary matrices, NQ elements in length, are included to denote the nodal origin and destination numbers for each flow variable.

## 2.4 Modification for Collection Networks

The mathematical statement developed in the previous section is for a distribution network, such as regional water supply. Both distribution and collection networks have the same basic spatial structures and only minor restrictions have to be imposed on the mathematical statement to allow for the differing network characteristics. It is important that the mathematical statement be applicable to both network types so that the same econometric model can be used in their analysis. A collection network is comprised of a set of community or input nodes and processing nodes interconnected by feasible links. The input nodes generate the commodity to be processed at the processing nodes which may or may not be coincident with the input nodes.

Two examples are provided to help in identifying the differences between distribution and collection networks. Figure 2.5 (a) depicts a 3 node distribution network with nodes 1 and 2 representing processing nodes and node 3 a nonprocessing node. The sign convention used in these examples is the same as presented in the last section, that is, the outflow from a node is assumed positive and, hence, an exogenous input, or material inflow to the network is positive and an exogenous output is negative in accordance with the form of the constraint equations. The constraining relations for the network of figure 2.5 (a) are:

$$\begin{aligned} Q_1 \quad + Q_3 \qquad\qquad &= S_1 \\ Q_2 \quad + Q_4 \qquad &= S_2 \\ -Q_1 - Q_2 \qquad\qquad - Q_5 &= -D_3 \end{aligned} \tag{2.11}$$

The structural matrix of the network is also shown in figure 2.5 (a) and the slack variables $Q_3$, $Q_4$ and $Q_5$ are schematized on the linear graph diagram.

Figure 2.5 (b) depicts a collection network with node 1 representing both a processing centre and a node which generates a volume of material. Node 2 is a processing node only and node 3 generates material. The nodes are numbered in accordance with the convention that processing nodes are numbered before nonprocessing nodes. The constraining relations for this example are:

$$\begin{array}{rrrrrl} Q_1 & & +Q_3 & & & = S_1 \\ -Q_1 & -Q_2 & & +Q_4 & & = 0 \\ & Q_2 & & & +Q_5 & = S_3 \end{array} \tag{2.12}$$

The structural matrix of the collection network is included in figure 2.5 (b). The stipulations $S_1$ and $S_2$ represent the material generated at node 1 and 2 respectively. The zero stipulation assigned to node 2 denotes that there is no material generated at that node. Node 2 is, however, capable of processing material which implies that a variable is required to represent the quantity of material processed. Both of the first two constraints in the equation group 2.12 are, before the addition of slack variables, of the following form:

$$\text{Net outflow} \leq \text{Gross supply of material} \tag{2.13}$$

Upon introducing slack variables, the relation becomes:

$$\text{Net outflow} + \text{Slack} = \text{Gross supply} \tag{2.14}$$

The slack variables introduced are $Q_3$ and $Q_4$. This slack represents the amount of material processed at the node which it is associated with. Since node 3 does not have the facilities for processing the material, the constraint relation is an equality before the addition of $Q_5$. This slack variable is, therefore, an artificial slack variable similar to those added to the nonprocessing node constraints in a distribution network. Similarly, the artificial slack variable is required only for the formulation of an initial feasible solution and, in general, will not appear in the final solution. In order to avoid the assignment of a nonzero value to the artificial slack variable when the model is searching for the optimal solution, the artificial slack variable must have a prohibitively high cost associated with it.

Another type of node, not mentioned yet, which can exist in either distribution or collection networks is a null, or junction, node. These nodes represent junctions in a pipeline or other type of conveyance route where a flow of material can be diverted to alternate links. There is, of course, no stipulation associated with a null node since it does not generate, process or consume any commodity. The general form of the continuity equation for a null node is:

$$\text{Net outflow} - \text{Net inflow} = 0.0 \qquad (2.15)$$

For convenience, the null nodes are classified as nonprocessing nodes in distribution networks although, they could theoretically be included in either category. To maintain consistency with the other demand node constraints, an artificial slack variable is incorporated in the null node constraints by subtracting it from the left hand side of the equation. This slack variable, as with the other artificial slack variables, represents an additional nonexistent source. It is important here, also, that the costs associated with nonzero values of the new artificial slack variables be prohibitively high.

In collection networks, the null nodes are again classified as nonprocessing nodes. The null node constraints have a nonnegative artificial slack variable added to the left hand side of the equations to maintain consistency with the other constraints. Since the null nodes have zero stipulations, their constraint equations appear the same as the constraint equations for processing nodes with zero stipulations. As with other artificial slack variables in the problem definition, the artificial slack variables in the null node constraints must have a prohibitively high cost assigned to them in the objective function.

The nodal number designations are similar for both collection and distribution type networks. The first nodal numbers are assigned to processing nodes and the remaining are nonprocessing nodes in both network types. Furthermore, since the nonprocessing node slack variables have high penalty costs associated with them so as to keep them out of the solution for both types of networks, it is important that the number of processing nodes as well as the total number of nodes be specified in the network definition data. In this manner, the model can distinguish between the node types and assign penalty costs where required. .

The most significant distinction between the mathematical statements for distribution and collection networks is the existence of negative stipulations and negative artificial slack variables in the processing node constraints of distribution networks. An important aspect of this difference is that it is useful in providing an easy means of allowing the econometric model to distinguish between the type of network to which it is being applied and in using this facility, the coding is made general enough to allow for any other differences between the network types. One such difference lies in the definition of the processing cost terms of the objective function. In the distribution network problem, the quantity of material processed is given by the processing node stipulation minus the value of the slack variable. Recall that the distribution network objective function appears as:

$$Z = \sum_{i=1}^{NQ-N} CT_i(Q) + \sum_{i=NQ-N+1}^{NQ-N+NCEN} CP_{i-NQ+N}(S_{i-NQ+N} - Q_i) \qquad (2.16)$$

$$+ \sum_{i=NQ-N+NCEN+1}^{NQ} CPEN_i(Q)$$

The quantity of material processed in a collection network is equal to the value of the slack variable for the processing node under consideration. Using the same notation as in equation 2.16, the objective function for a collection network is as follows:

$$Z = \sum_{i=1}^{NQ-N} CT_i(Q) + \sum_{i=NQ-N+1}^{NQ-N+NCEN} CP_{i-NQ+N}(Q) + \sum_{i=NQ-N+NCEN+1}^{NQ} CPEN_i(Q) \qquad (2.17)$$

Since both objective functions have the same format and the only difference lies in the definition of the processing quantity, the model easily compensates for this once the network type is identified.

## 2.5 Alternate Methods of Problem Definition

The preceding two sections present a method of stating the network problems in general mathematical terms which can be used in an econometric model. The objective function is formed as a function of the flow design variables and the network topology is defined by the constraint equations generated from mass balance restrictions at the nodes. Nodal origin-destination pairs are used to define the flow variables. The remaining network characteristics, such as link length and nodal state, can then be stored in separate arrays which are accessed as required by cost function routines.

It is important to realize that the form of the mathematical statement adopted is not necessarily the most simple or straightforward, but it is developed to be compatible with possible solution algorithms. It is worthwhile, however, to examine other methods of defining the networks mathematically and discuss any advantages or disadvantages. Through the development of solution techniques for transportation problems, formulations are developed which take advantage of the special structure possible for the matrices (9, 3). In a transportation model without transshipment, the matrix defining the connectivity of the nodes in a network can be stored very efficiently in a rectangular array displaying the flow variables in an origin-destination manner. When applied to a distribution network, each row of the array represents a set of flow variables with the property of originating from a common node. Each array column represents a set of flow variables with the property of terminating at a common node. Therefore, the sum of the elements in any row is equal to the supply available at the corresponding node and the sum of the elements in any column is equal to the demand at the corresponding node. This formulation is further illustrated using the example in figure 2.6 (a). The 5 node distribution network has 2 supply and 3 demand nodes. The resulting matrix, shown in table 2.2 (a), is a 6 element (2x3) nodal connectivity matrix with all positive flow values. The variable designation is slightly different from that used before with $Q_{ij}$ representing the flow originating at supply node i and terminating at demand node j. All possible direct transportation links are included in the example resulting in a full matrix. If any links are not included, the corresponding flow variable is zero valued. The connectivity matrix may, therefore, have a number of "empty" locations, the number of which depends upon the degree of connectivity in the network problem. The nodal supply, $S_i$, and nodal demand, $D_j$, values appear as edge vectors on the right side and bottom of the connectivity matrix respectively.

The same type of matrix formulation can be extended for use in transportation networks with transshipment. Figure 2.6 (b) displays the same basic network as 2.6 (a) with the difference that all possible transshipment links are included. The nodes have been renumbered since flows can exist between like nodes as well as unlike nodes. The connectivity matrix defining the network, given in table 2.2 (b), still uses an origin-destination formulation but it is now square (i.e. NxN elements). A new variable, T, is introduced to represent the maximum amount of material which can be transshipped through any node (i.e. the sum of the supply or demand quantities, which are equal in a balanced system). The maximum transshipment quantity is added to the supply and demand quantities along the right side and bottom of the matrix. The variables $Q_{ij}$, where i=j, represent a slack variable to the transshipment in the system, thus the amount of material transshipped through node i is T - $Q_{11}$. Through the inclusion of transshipment, the matrix size has increased significantly over that required for a network with no transshipment.

The application of the nodal connectivity matrix to a typical network is illustrated in figure 2.7. A five node distribution network is shown with a specified solution. The flow variables in the corresponding connectivity matrix have values consistent with the solution given. The network solution given in this example is of the branching type as defined earlier in this chapter. Branching networks result in sparse matrices (matrices with many zero elements) as seen in figure 2.7. This is common in network analysis problems since the optimal policy is a branching network.

The matrices shown above are for balanced networks, where the total network supply equals the total network demand. If the supply and demand quantities do not balance an extra row or column, depending upon whether total supply or demand is greater, can be added and the corresponding stipulation equal to the excess amount. This procedure is analogous to the use of slack variables in constraint relations to change inequality constraints to equalities.

To complete the mathematical statement, the objective function must be included. In problems where the cost functions are separable, a cost function matrix can be formed which is compatible with the flow connectivity matrix. Table 2.2 (c) illustrates the cost function matrix for the example in figure 2.6 (b). The new matrix is the same size (NxN) as the connectivity matrix and each cost function, $C_{ij}(Q)$, is located in the same position as its corresponding flow variable, $Q_{ij}$. The cost functions, $C_i(-Q)$, which lie along the diagonal of the cost matrix represent the transshipment costs. The negative sign preceding the flow variable is a result of transposition since the transshipment quantity is costed with supply and demand quantities. The network cost is comprised of the sum of the costs calculated through introducing each flow variable into its corresponding cost function.

When comparing the use of connectivity matrices to the constraint equation form, it appears, upon first inspection, that the use of the connectivity matrix would be advantageous due to the smaller matrices involved. In general, the following matrix sizes are required for the mathematical statement.

1. Constraint equation formulation:

| | | |
|---|---|---|
| Objective function (cost): | | 1 x NQ |
| Structural Matrix | : | N x (NQ + N) |

2. Transportation (connectivity) formulation:

| | | |
|---|---|---|
| Cost Matrix | : | N x N |
| Structural Matrix | : | N x N |

This type of comparison is deceiving, however, since further matrices are required in the solution algorithms with the number and size depending upon the solution algorithm applied. Additional array storage is required for storing network information such as link lengths and nodal states. Further investigation of the actual array storage requirements for the solution algorithms is made in Chapter 4 where different solution algorithms are investigated. In general, it is found that solution algorithms require substantially more computer storage than the mathematical statement.

It is decided that the constraint equation formulation is the best to use in the mathematical statement for the model presented here. The main reason for adopting this method is that most standardized algorithms are structured to solve problems expressed in the form of an objective function subject to a set of constraint equalities or inequalities. Restricting the econometric model to the use of optimization routines designed for connectivity constraints would remove the generality desired for the model.

It should be noted here that if, as a result of problem size, the matrices involved are excessively large, it is possible to use special computer techniques to decrease the amount of computer storage required. For example, since the elements of the structural coefficient matrix are only one of 0, 1 or -1, it is possible to use a space saving translation technique. The elements could be transformed into "equivalent" numbers, say 0, 1 and 2 represented in binary form as 00, 01 and 10 respectively. Then several elements could be stored in a single word, the actual number being dependent on the word space available (e.g. 29 in CDC 6400). The main drawback to this procedure is that the need for the translation of matrix elements from zero-one form to a 2-bit binary form and back again would naturally increase the execution time of the program. This procedure may be required, however, if a large regional network is being analyzed and the matrices exceed the storage capabilities of the computer facilities. The actual development of storage saving procedures is not carried out in this study but is left as an opportunity for further work.

## 2.6 Properties of the Solution

The nature of the network problems presented results in certain solution properties which are important when considering the type of solution method to use. The two important characteristics of network problems are:

1) The objective function is separable and concave (in practice, the function is also monotonic).

2) The solution is subject to a set of linear constraints.

As a result of these characteristics, the following statements may be made which significantly affect the method of solution and choice of algorithm.

Theorem 1: The linear problem constraints form a convex feasible solution space.

Theorem 2: A function which comprises the sum of a series of separable, strictly concave functions is in itself strictly concave.

Theorem 3: The minimum cost solution of a concave objective function subject to a set of linear constraints must lie on a vertex of the feasible solution space, called a basic solution.

Theorem 1 is reproduced in detail in Appendix A and is identical to the condition generally used for Linear Programming. Appendix B contains a fuller development of the statement in Theorems 2 and 3 and is fundamental to the choice of the solution strategy for problems of this type.

To demonstrate the location of the optimal solution to network problems, a simple 3-node distribution network is solved, using a graphical method. This example also facilitates the presentation of any further properties of the solution. The network considered is illustrated in figure 2.8. A single demand node with the stipulation $D_3$ is connected, through two links, to two supply nodes with the stipulations $S_1$ and $S_2$. It is possible to have either supply node 1, 2 or both supplying the demand node with the required commodity. Only two design variables $Q_1$ and $Q_2$ need be considered, and the corresponding conveyance costs for each are represented by the cost functions $C_1(Q)$ and $C_2(Q)$. The mathematical statement of the problem becomes:

$$\min_{Q_i^*} z = C_1(Q) + C_2(Q) \qquad (2.18)$$

$$\text{Subject to:} \quad Q_1 \leq S_1$$

$$Q_2 \leq S_2$$

$$Q_1 + Q_2 = D_3$$

$$Q_1, Q_2 \geq 0.0$$

In this case $S_1, S_2 > D_3$

In this mathematical statement, slack and artificial slack variables are not introduced since a manual solution technique is used. Although network costs normally include both processing and transportation costs, for simplicity, only transportation costs are considered in this illustrative problem. The transportation cost functions are typically concave and monotonic, exhibiting economies of scale. In this particular problem, the cost functions appear in figure 2.9. The optimal solution is obvious through inspection of the cost curves. Due to the concavity of the cost function, any solution with both $Q_1$ and $Q_2$ nonzero is more costly than having only one nonzero variable. The optimal solution is, therefore, obtained through the lowest intercept of the vertical ordinate through the point $Q=D_3$ on the cost curves. In this case, the optimal solution is $Q_2=D_3$ and $Q_1=0.0$. This method is only valid, however, when both $S_1$ and $S_2$ exceed $D_3$. If either one or both are less than $D_3$, a trial and error method is required in determining the optimal solution. The effort required in finding the solution is reduced substantially by the fact that one variable will have full value due to economies of scale (i.e. $Q_1$ will equal $S_1$ or $D_3$, or $Q_2$ will equal $S_2$ or $D_3$).

Further insight into the solution properties of the network problem presented here is obtained by a different graphical method. If the constraint equations and objective function are plotted on a graph of $Q_1$ versus $Q_2$, figure 2.10 is obtained. In the graph, it is assumed that $S_1$ is greater than $D_3$ and $S_2$ is less than $D_3$. The problem constraints form a set of lines which bound a convex region. If the constraint $Q_1 + Q_2 = D_3$ had the form $Q_1 + Q_2 \geq D_3$ instead, the convex region bounded by the constraints would represent the feasible solution set. Due to the equality constraint, however, all solutions lie along the line A - B. The cost isograms (i.e. iso-cost lines), which appear in the figure, comprise the sum of the two cost functions for the corresponding flowrates. The iso-cost curves of figure 2.10 are typical of concave monotonic cost functions. The values of the isograms increase in the upper right hand direction.

In linear programming, the objective function forms a set of linear isograms of constant slope and, therefore, parallel to each other. To obtain the optimal solution, successive isograms are considered, moving in the direction of decreasing cost, until an extreme point of the convex solution set is reached. In this case, the extreme point is the optimal solution. With concave cost functions, however, the isograms are neither linear nor parallel but have different degrees of curvature. The method of considering successive parallel isograms does not ensure an optimal solution and therefore cannot be used. In this simplified problem, the optimal solution can be found by inspection providing sufficient isograms are plotted. It is found, from figure 2.10, that the optimal solution lies at point A in this case where $Q_1=D_3$ and $Q_2=0.0$.

It is important to notice in figure 2.10 that certain isograms cross over the constraint $Q_1 + Q_2 = D_3$ which implies that along the line segment A - B, there are points which have higher costs than at point A or B. In more mathematical terms, this is in effect saying that the sum of concave functions is in itself a concave function and thus the cost of any point along the line A - B must be greater than the equivalent weighted average of the costs at the extremities A and B, i.e.:

$$C\left(\sum_{i=1}^{2} \alpha_i X_i\right) > \sum_{i=1}^{2} \alpha_i C(X_i) \qquad (2.19)$$

$$\text{where} \quad \sum_{i=1}^{2} \alpha_i = 1$$

and $X_i$ is a transformed variable.

$$X_i = Q_i - S_i + D_3$$

Therefore, if applying a solution algorithm and using the solution at point B as a starting position, the solution mechanism must effectively cross over a "hill" of higher cost along the line A - B to reach the optimal solution at point A. Point B can therefore be identified as a local optimum. This illustrates the main difficulty encountered in devising a solution algorithm which will converge to a global optimum.

The most important conclusion which can be derived from this example is that the optimum lies on an exterior point or vertex of the convex solution set. It is proven generally in Appendices A and B that for problems with linear constraints and a nonlinear concave objective function, the optimal solution lies on a vertex of the solution set. Another important aspect lies in the fact that adjacent vertices can be separated by segments representing a boundary of the solution set with points having higher costs than that of either vertex. This may result in convergence to a local optimum when using solution algorithms which employ a strategy based solely on information available at the current position within the feasible space. Hill climbing techniques which use local gradients and simplex type algorithms which depend on local artificial cost coefficients are typical examples. This aspect is demonstrated further in Chapter 3. It appears as though optimization routines which consider only the vertices in their search for the optimal solution are most likely to converge to the global optimum and will generally be more efficient in execution times than those which search through the entire convex set.

## 2.7 Conclusions

The importance of graphical theory is stressed in the beginning of this chapter. Transforming a regional network into an equivalent linear graph simplifies the problem, allowing the designer to consider initially the topological characteristics of the system, temporarily setting aside the physical properties for later use in the econometric model.

The two types of network topology involved are branching and cyclic networks. In general, initially specified regional networks are cyclic and the optimal solution determined eliminates redundant links and reduces the network to a branching form. Indices, such as the Alpha index, provide a measure of the connectivity of a network, which is helpful in determining the relative complexity of a problem.

The mathematical statement adopted for use in the econometric model comprises an objective function subject to a set of equality constraints. The objective function represents the total network costs and is formed by the sum of separable, concave monotonic cost functions of the design variables. The equality constraints are nodal mass relations and, hence, are linear in the design variables. Minor differences occur in the development of the mathematical statement for distribution and collection networks, but the adoption of certain rules allows the use of the same formulation:

1) Outflow from a node is assumed positive.

2) Processing and nonprocessing nodes must be distinguished by the node numbering convention adopted (see section 2.4).

3) The null or junction nodes are grouped with nonprocessing nodes.

4) For a distribution network, the processing node constraints represent the upper limit on supply from that node. The nonprocessing node stipulation, which ~~is~~ negative, represents the nodal demand.

5) For a collection network, both the processing and nonprocessing stipulations represent the material generated at a node.

The mathematical statement has certain features built in to facilitate its use with standardized solution algorithms. Each constraint has a slack variable associated with it and appropriate costs are included in the cost functions. The mathematical statement is not structured to allow for the transshipment of untreated material between processing nodes.

Other possible formulations for the mathematical statement are investigated. These involve mainly defining the network through the use of a connectivity matrix. Using this method, the structural matrix is smaller than that required for the constraint equation formulation. Unfortunately, fewer algorithms exist which can be used in conjunction with connectivity matrices. Therefore, to facilitate access to a greater number of optimization algorithms, the equality constraint formulation is preferred. Computer storage saving techniques are possible in case the array storage requirements exceed the computer facility capabilities in larger problems.

When investigating the properties of the optimal solution to network problems, it is found that the existence of linear constraints and separable concave cost functions are important in that they dictate that the optimal solution lies on a vertex of the convex solution set. Furthermore, the line segments joining adjacent vertices can have points of higher cost than either vertex. This should be considered when selecting a solution method to be used since it implies that vertex searching algorithms will have better convergence properties and have the potential for a more economical optimization strategy.

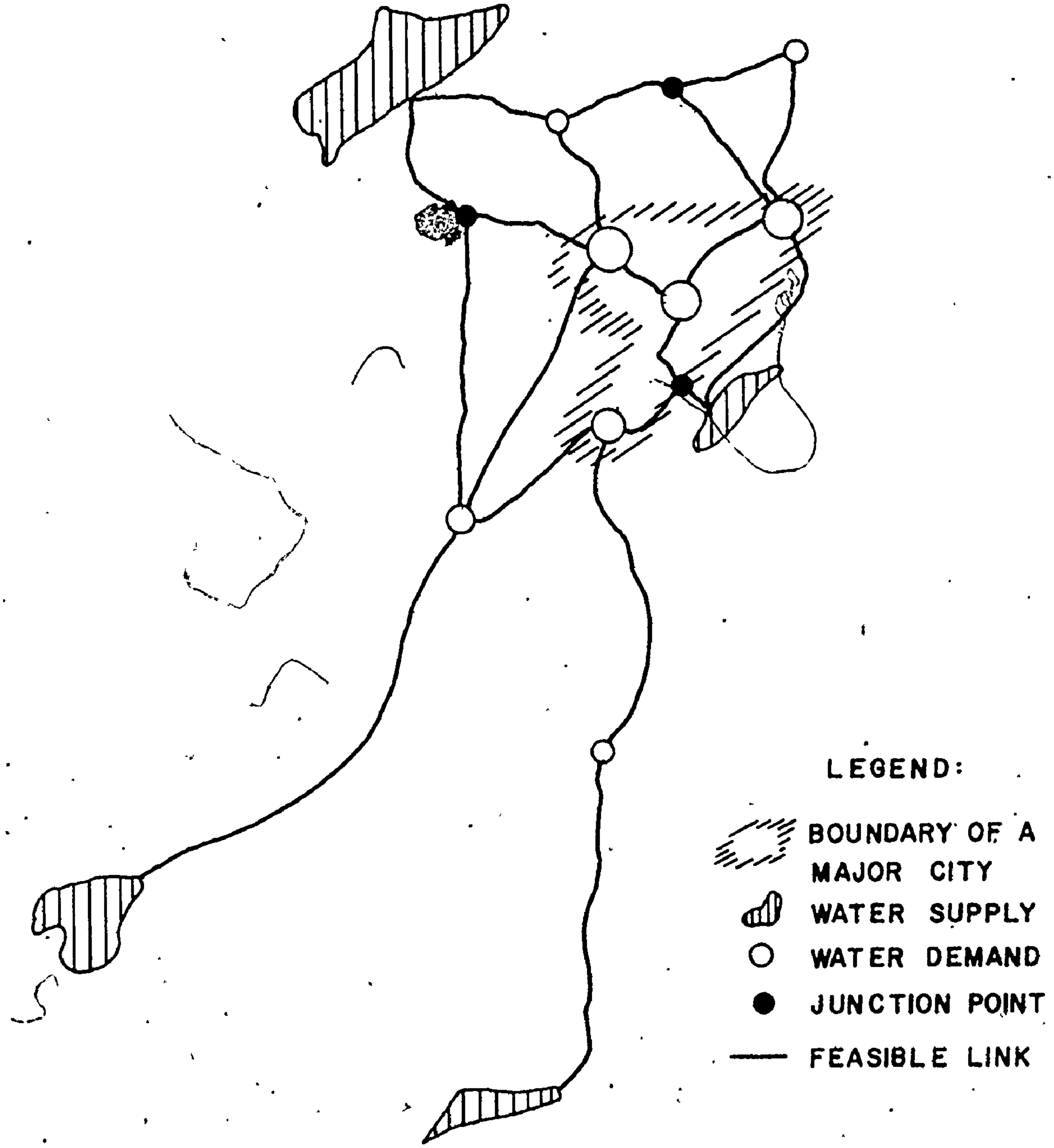


FIGURE 2·1(a) – WATER SUPPLY NETWORK

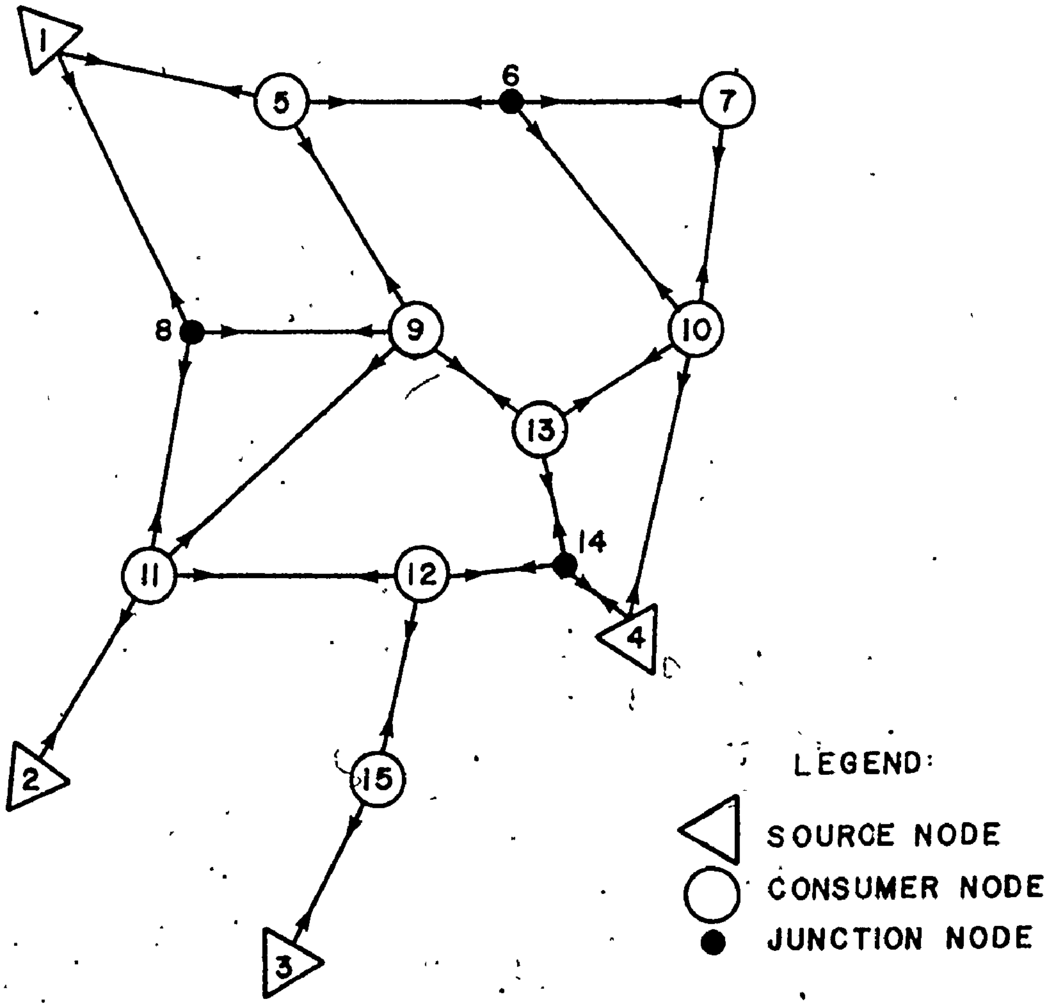


FIGURE 2·1(b) – GRAPHICAL REPRESENTATION OF THE WATER SUPPLY NETWORK

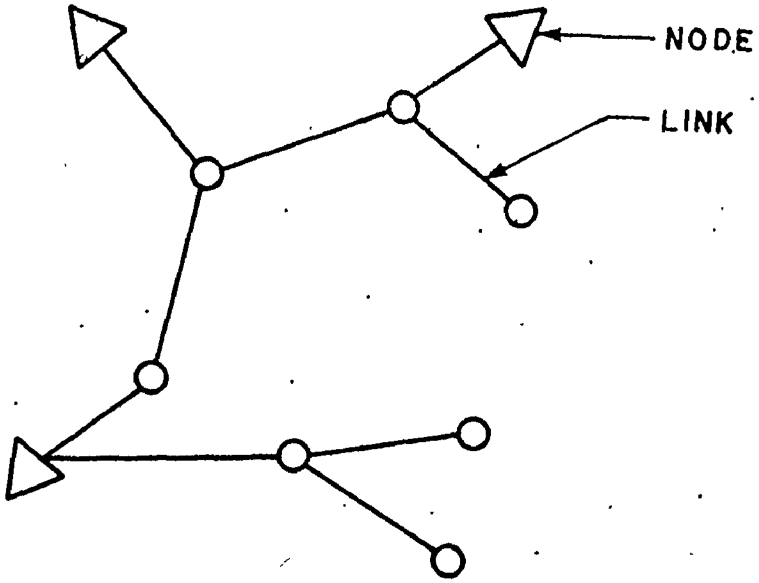


FIGURE 2·2(a) - TYPICAL BRANCHING NETWORK

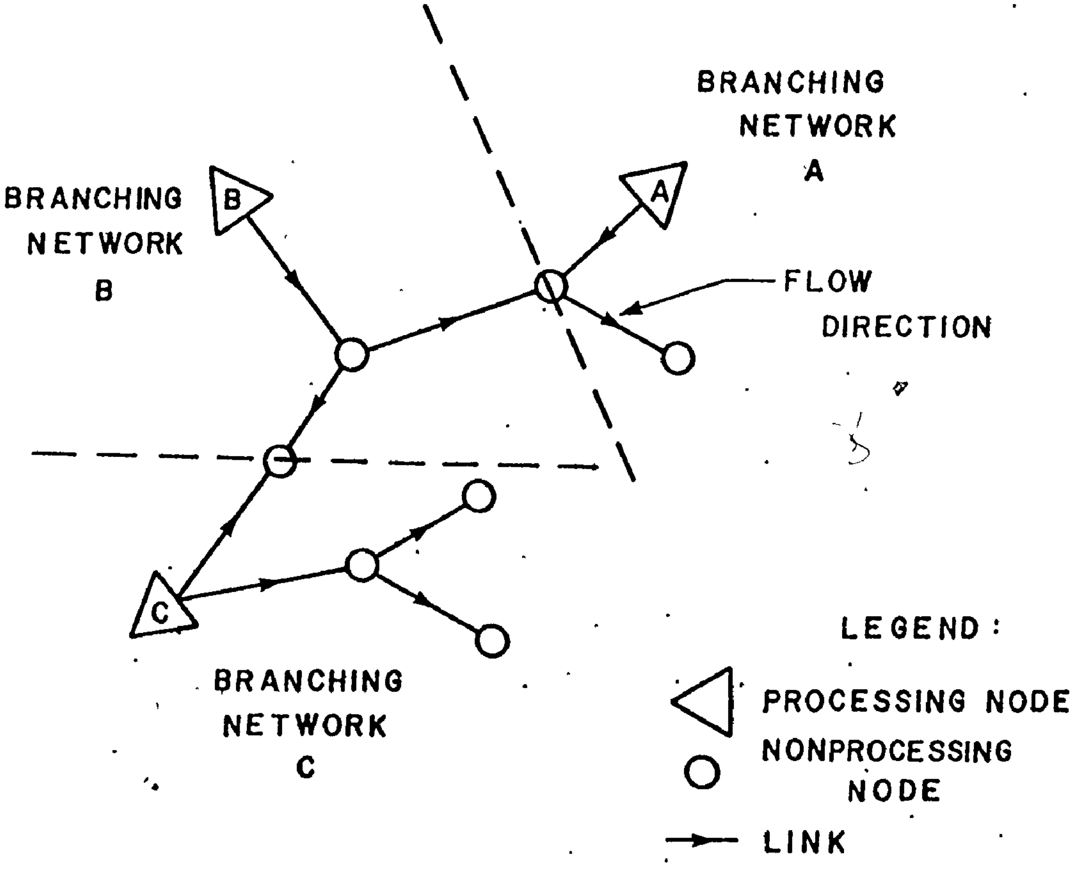


FIGURE 2·2(b) — DIVISION OF A BRANCHING NETWORK FOR A DISTRIBUTION SYSTEM

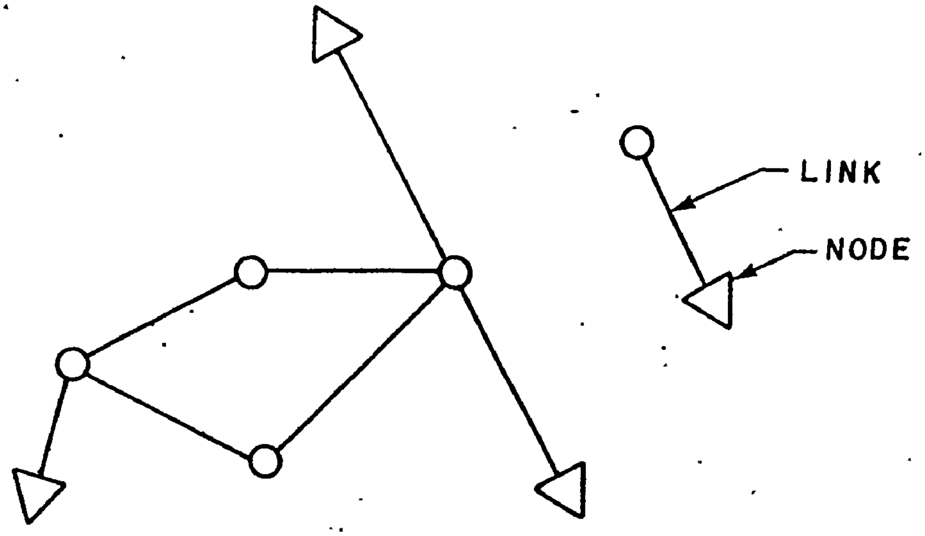


FIGURE 2·3 – NETWORK WITH A SINGLE CIRCUITAL PATH

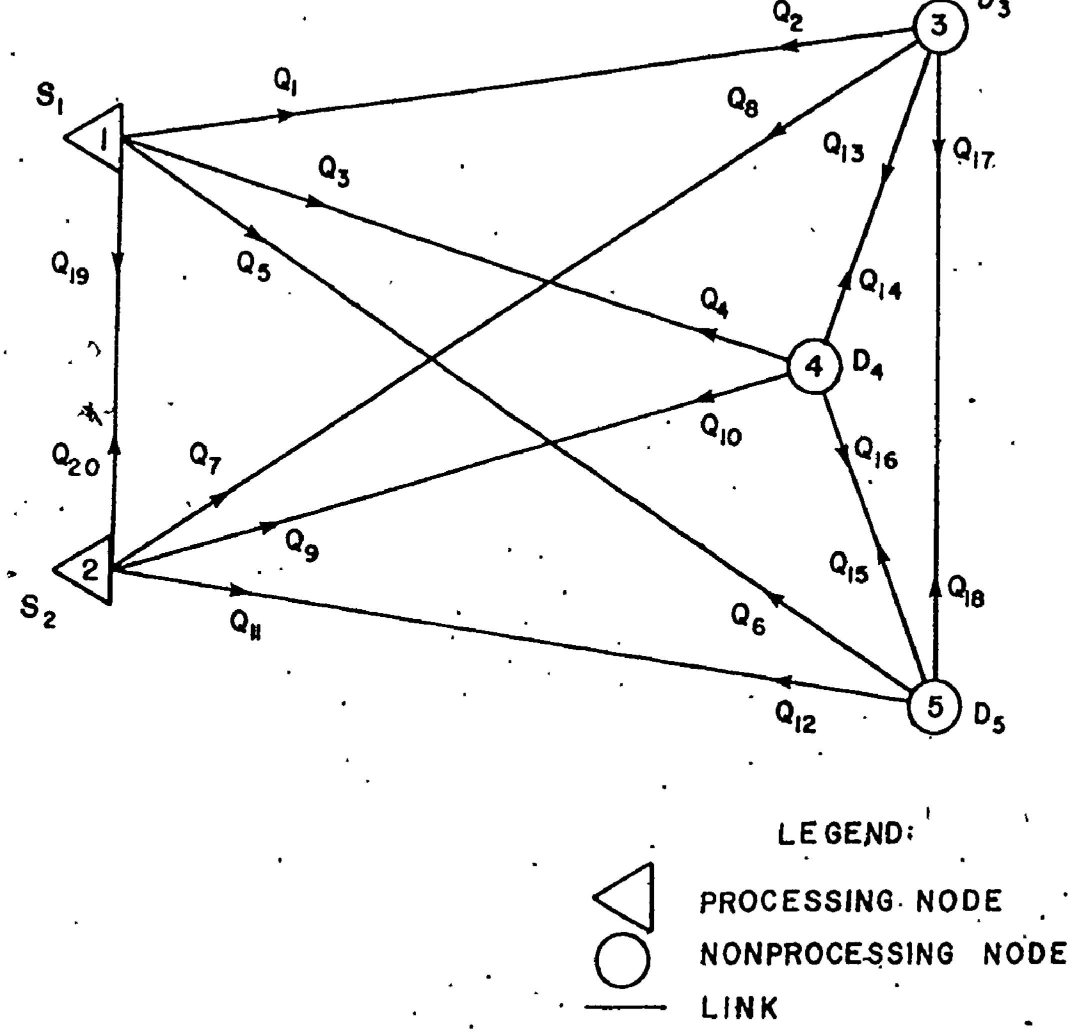
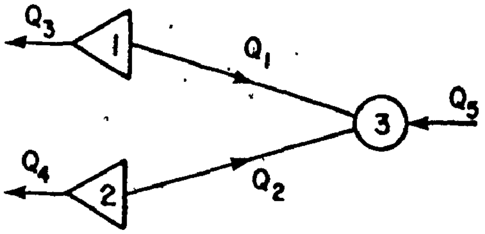


FIGURE 2·4 – DISTRIBUTION NETWORK – 5 NODES
ALL POSSIBLE LINKS INCLUDED



LEGEND:

◁ PROCESSING NODE

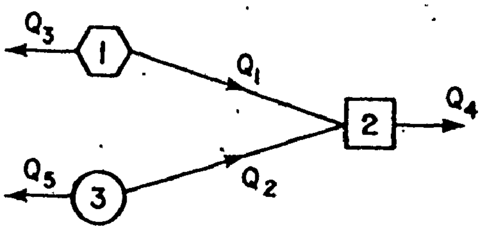
○ NONPROCESSING NODE

— LINK

STRUCTURAL MATRIX

| NODE | FLOW VARIABLE $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | STIPULATION |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 | $S_1$ |
| 2 | 0 | 1 | 0 | 1 | 0 | $S_2$ |
| 3 | -1 | -1 | 0 | 0 | -1 | $-D_3$ |

FIGURE 2·5(a) – DISTRIBUTION NETWORK AND MATRIX – 3 NODES



LEGEND:

⬡ INPUT AND PROCESSING NODE

○ INPUT NODE

□ PROCESSING NODE

STRUCTURAL MATRIX

| NODE | FLOW VARIABLE $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | STIPULATION |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 | $S_1$ |
| 2 | -1 | -1 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 | $S_3$ |

FIGURE 2·5(b) – COLLECTION NETWORK AND MATRIX – 3 NODES

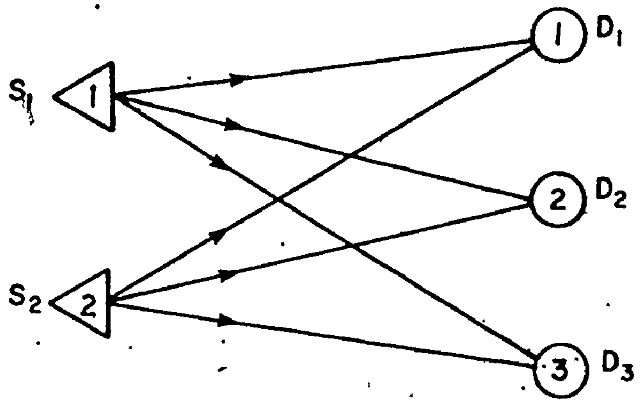


FIGURE 2·6(a) – DISTRIBUTION NETWORK – 5 NODES

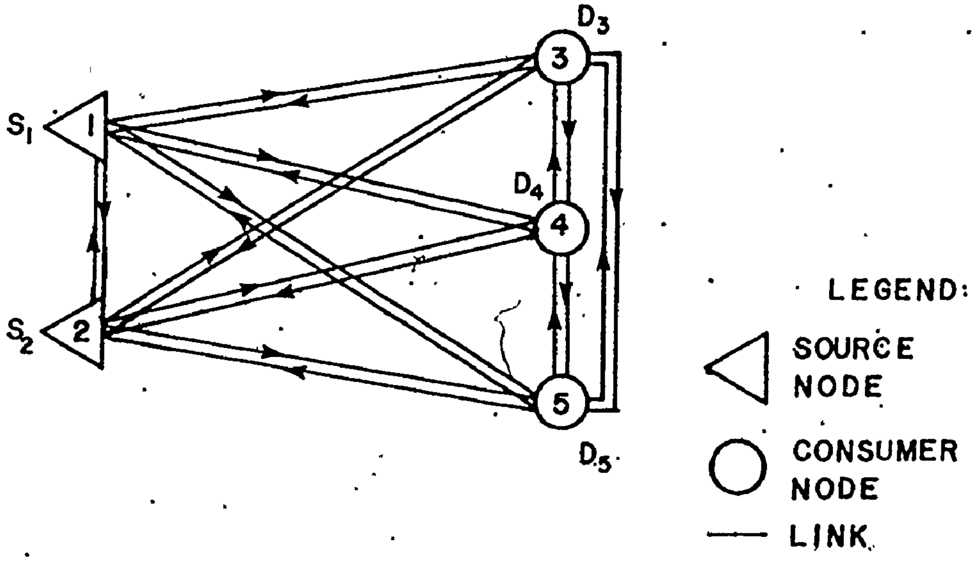
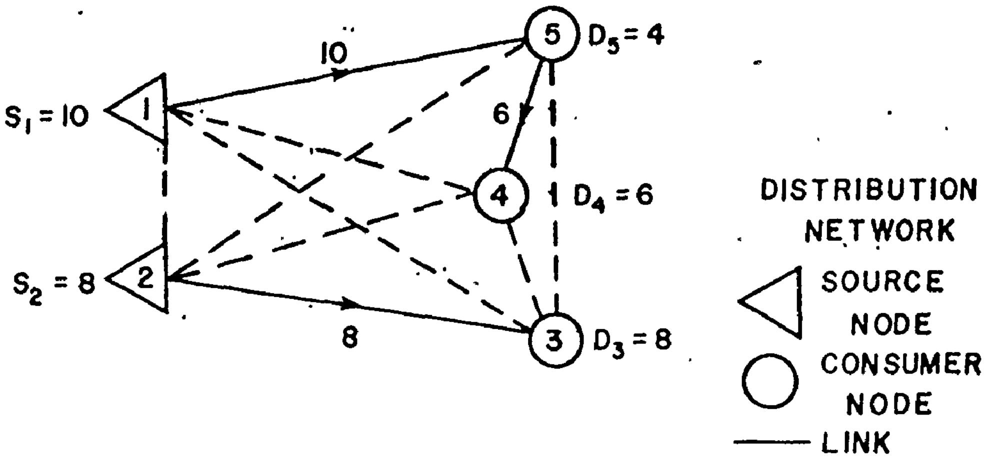


FIGURE 2·6(b) – DISTRIBUTION NETWORK – 5 NODES (WITH TRANSSHIPMENT)



NODAL CONNECTIVITY MATRIX

| | | | | DEMAND | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | |
| SUPPLY | 1 | 18 | 0 | 0 | 0 | 10 | 10 + 18. |
| | 2 | 0 | 18 | 8 | 0 | 0 | 8 + 18 |
| | 3 | 0 | 0 | 18 | 0 | 0 | 18 |
| | 4 | 0 | 0 | 0 | 18 | 0 | 18 |
| | 5 | 0 | 0 | 0 | 6 | 12 | 18 |
| | | 18 | 18 | 8<br>+<br>18 | 6<br>+<br>18 | 4<br>+<br>18 | |

FIGURE 2·7 – CONNECTIVITY MATRIX FOR A SPECIFIED SOLUTION

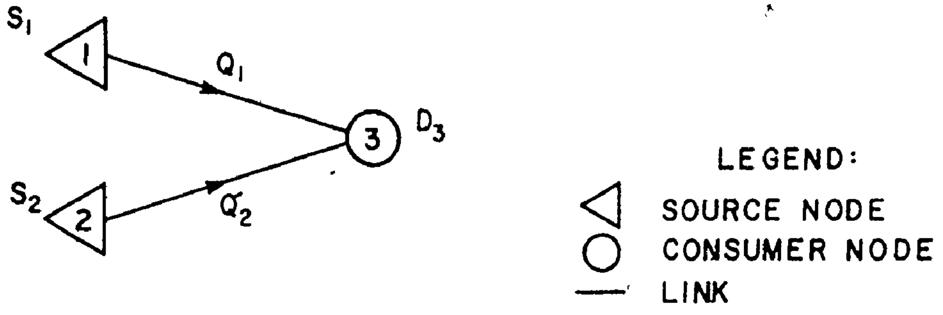


FIGURE 2.8 - SIMPLE DISTRIBUTION NETWORK

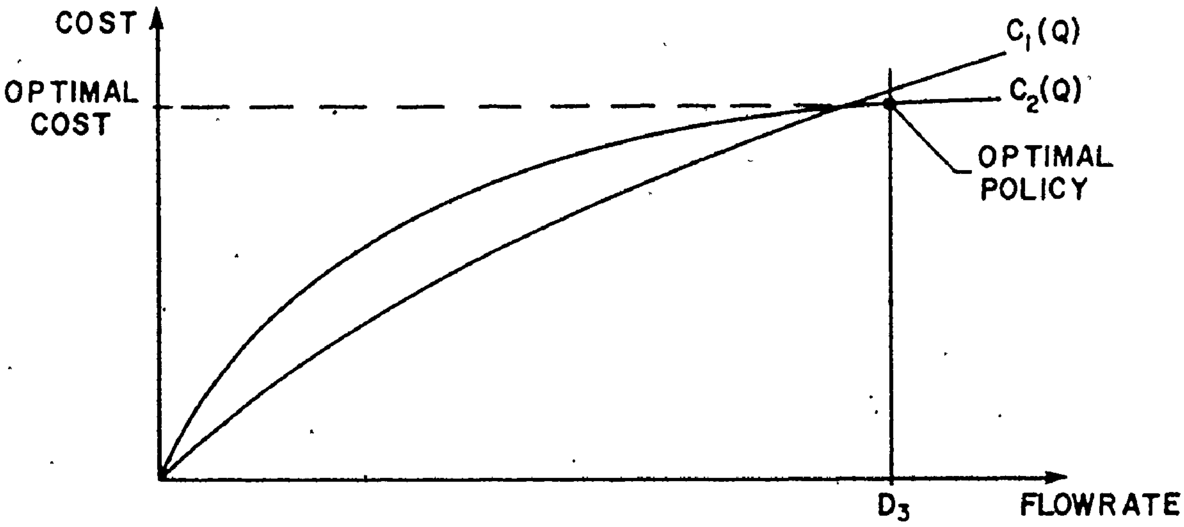


FIGURE 2.9 - TRANSPORTATION COSTS FOR NETWORK IN FIGURE 2.8

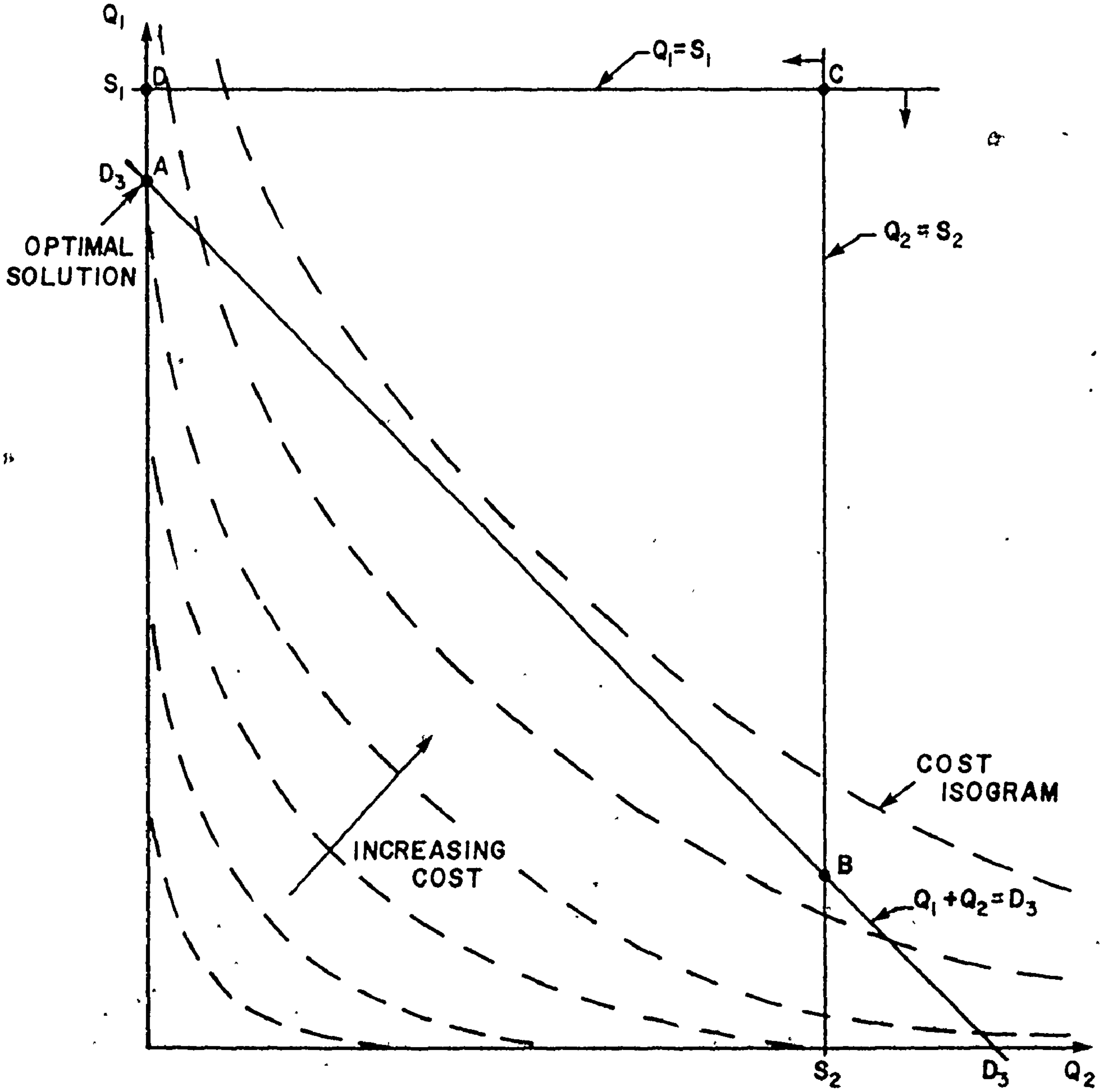


FIGURE 2·10 — GRAPHICAL SOLUTION OF A TWO VARIABLE PROBLEM

# TABLE 2·1 – STRUCTURAL COEFFICIENTS AND ORIGIN–DESTINATION PAIRS

STRUCTURAL COEFFICIENT MATRIX

|  |  | FLOW VARIABLES |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | SLACK VAR. |  | ARTIFICIAL SLACK VAR. |  |  | STIP-ULATIONS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |  |
| NODE NO. | 1 | 1 | -1 | 1 | -1 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | 0 | 0 | 0 | 0 | $S_1$ |
|  | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | -1 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 1 | 0 | 0 | 0 | $S_2$ |
|  | 3 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | $-D_3$ |
|  | 4 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 1 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | $-D_4$ |
|  | 5 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 1 | -1 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | $-D_5$ |

FLOW VARIABLE ORIGIN-DESTINATION PAIRS

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| U/S Node | 1 | 3 | 1 | 4 | 1 | 5 | 2 | 3 | 2 | 4 | 2 | 5 | 3 | 4 | 5 | 4 | 3 | 5 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| D/S Node | 3 | 1 | 4 | 1 | 5 | 1 | 3 | 2 | 4 | 2 | 5 | 2 | 4 | 3 | 4 | 5 | 5 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |

TABLE 2·2(a) — ORIGIN-DESTINATION MATRIX FOR A NETWORK WITH NO TRANSSHIPMENT

| | | | |
|---|---|---|---|
| $Q_{11}$ | $Q_{12}$ | $Q_{13}$ | $S_1$ |
| $Q_{21}$ | $Q_{22}$ | $Q_{23}$ | $S_3$ |
| $D_1$ | $D_2$ | $D_3$ | |

Supply Nodes

Demand Nodes

where:

$Q_{ij}$ = Flow from Node i to Node j

$S_i$ = Supply at Node i

$D_i$ = Demand at Node i

TABLE 2·2(b) — ORIGIN-DESTINATION MATRIX FOR A NETWORK WITH TRANSSHIPMENT

| | | | | | |
|---|---|---|---|---|---|
| $Q_{11}$ | $Q_{12}$ | $Q_{13}$ | $Q_{14}$ | $Q_{15}$ | $S_1 + T$ |
| $Q_{21}$ | $Q_{22}$ | $Q_{23}$ | $Q_{24}$ | $Q_{25}$ | $S_2 + T$ |
| $Q_{31}$ | $Q_{32}$ | $Q_{33}$ | $Q_{34}$ | $Q_{35}$ | T |
| $Q_{41}$ | $Q_{42}$ | $Q_{43}$ | $Q_{44}$ | $Q_{45}$ | T |
| $Q_{51}$ | $Q_{52}$ | $Q_{53}$ | $Q_{54}$ | $Q_{55}$ | T |
| T | T | $D_3$<br>+<br>T | $D_4$<br>+<br>T | $D_5$<br>+<br>T | |

where:

T = Maximum possible transshipment

$$= \sum_{1}^{2} S_i = \sum_{3}^{5} D_i$$

## TABLE 2·2(c) — COST MATRIX FOR A NETWORK WITH TRANSSHIPMENT

| | | | | |
|---|---|---|---|---|
| $C_1(-Q)$ | $C_{12}(Q)$ | $C_{13}(Q)$ | $C_{14}(Q)$ | $C_{15}(Q)$ |
| $C_{21}(Q)$ | $C_2(-Q)$ | $C_{23}(Q)$ | $C_{24}(Q)$ | $C_{25}(Q)$ |
| $C_{31}(Q)$ | $C_{32}(Q)$ | $C_3(-Q)$ | $C_{34}(Q)$ | $C_{35}(Q)$ |
| $C_{41}(Q)$ | $C_{42}(Q)$ | $C_{43}(Q)$ | $C_4(-Q)$ | $C_{45}(Q)$ |
| $C_{51}(Q)$ | $C_{52}(Q)$ | $C_{53}(Q)$ | $C_{54}(Q)$ | $C_5(-Q)$ |

# CHAPTER 3

# NONLINEAR SOLUTION TECHNIQUES

## 3.1 Introduction

The preceding chapter describes the specification of a generalized mathematical model for the solution of econometric network problems.

In this chapter, a number of solution algorithms are investigated for possible inclusion in the model. Since most nonlinear solution algorithms which could be used for problems considered here use a problem formulation comprising an objective function subject to a set of constraints, this formulation is adopted for the mathematical statement. By adopting this format, a wider choice of solution algorithms for the model is made possible.

To consider alternate solution algorithms, the properties of the problem solution are first reviewed. The general criteria required for useful nonlinear solution algorithms are then outlined as a set of guidelines to be followed. Following a general introduction to nonlinear programming, some specific techniques are investigated only so far as to determine if they represent feasible techniques for the problem. A useful solution algorithm is then chosen for use in the development of the econometric model.

## 3.2 Choosing a Nonlinear Solution Technique

Through development of the generalized mathematical statement for network problems and investigation of a simplified problem, the following important solution properties are found:

1) The linear solution constraints form a convex solution set.

2) The use of separable, concave, monotonic cost functions results in an optimal solution which lies on a vertex of the feasible solution region.

3) Linear segments joining adjacent vertices in the solution region may have higher costs than at either vertex.

These solution properties were demonstrated in Chapter 2, section 2.6 and illustrated in figure 2.10. In this study, problem types which exhibit these solution properties are referred to as "Concave Programming" problems.

Concave programming, as used here, relates to problems involving the minimization of a concave objective function subject to a set of linear constraints. Due to the nature of the problem, the cost isograms formed by the objective function in the region of interest have the same overall curvature as the convex solution set near the location of a minimal solution (see figure 3.1 (a)). This property leads to the existence of local optima. Analogous to this case is the problem of maximizing a convex objective function subject to linear constraints. In this case, the curvature of the objective function is reversed and the region of interest is on the opposite side of the convex solution set (see figure 3.1 (b)). It can be seen that the solution properties of these problem types are similar so that local optima can exist in either case.

"Convex Programming", on the other hand, involves the minimization of a convex objective function subject to linear constraints as illustrated in figure 3.2 (a). Since, in the region of interest, the sense of curvature of the objective function and solution set is opposite, the isogram representing the minimum cost can touch the convex solution set at only one point. Convex programming, therefore, yields a global optimum. The optimum may not necessarily lie on a vertex of the convex set. Maximization of a concave objective function subject to linear constraints exhibits similar properties with the optimum being global (see figure 3.2 (b)).

Considering the availability of convex programming algorithms and the possibility of a guaranteed global solution (9, 12), it is worthwhile to determine if the algorithms could be used for network problems. It is possible to transform a concave objective function to a convex form by sign reversal but the optimization strategy must also be reversed to maximization, i.e.:

$$\text{Min}\ (Z) \equiv \text{Max}\ (-Z)$$

where: $Z$ = objective fct.

The problem now takes the form of the maximization of a convex objective function subject to linear constraints which, unfortunately, still comes under the category of concave programming as illustrated in figure 3.1 (b). Transformation of the problem does not lead to a global solution implying that convex programming cannot be applied to problems of this type and solution algorithms intended for specific application to convex problems cannot be used.

In general, computational efficiency and accuracy of a solution algorithm are mutually incompatible and several criteria are required to describe the effectiveness of any procedure for the solution of network problems. The most important criteria are:

1) Execution time: Closely related to this is the number of functional evaluations required to attain the final solution.

2) Computer storage requirements.

3) Accuracy.

4) Computational stability: The ability of the algorithm to detect errors in the problem or diagnose nonconvergence.

5) Ease of use: It must be remembered that the total cost of problem solving includes the time taken for problem formulation and coding.

6) Form of the objective function: Certain solution algorithms (i.e. gradient search techniques) may require differentiable functions.

When evaluating the above criteria for different solution algorithms, extensive testing may be required, especially in determining the execution times, accuracy of the solution and stability of the algorithm. Himmelblau (19) reports on an extensive study comparing the performance of nonlinear solution algorithms which is helpful in determining the usefulness of certain algorithms.

Computer storage requirements may limit the size of problem which can be solved. For example, experience has shown that the storage requirements for a typical 45 node, 71 link network using a linear formulation can exceed the capabilities of many systems. This leads to the use of storage saving techniques which generally impose a penalty in execution costs.

In view of the importance of storage requirements, two indices are developed which are used to compare the storage requirements of different solution algorithms when it appears that the storage may be an important criteria. To provide a basis for comparison, the indices are initially evaluated assuming that the problem may be presented in linear programming form.

(1) Mathematical Statement Size

Using the mathematical model defined by equation 2.10, one cost function is required for each of the NQ flow variables, where NQ = 2 x number of links + number of nodes (N). The problem constraints are formed by N constraint equations. With a linear formulation, the following arrays would result:

| | | |
|---|---|---|
| Cost array | - | (1 x NQ) |
| Structural matrix | - | (N x NQ) |

(2) Solution Algorithm Storage Requirements

This index is an extension of the first. Assuming that the problem is a linear programming one, the SIMPLEX algorithm can be used (33), for which the following storage is required.

$$((2 + N) \times NQ) + ((6 + N) \times N)$$

For many network problems, the parameters NQ and N are found to be approximately related by the ratio $\frac{NQ}{N} = 4$. Using this ratio, the algorithm storage requirements for different network problem sizes may be tabulated.

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Storage | 640 | 2280 | 8560 | 33120 | 51400 |

The increase in storage is seen to be exponential with N, demonstrating the difficulty involved in solving larger network problems. The storage calculated above does not include additional storage for further required network parameters. This additional storage does not vary for different solution algorithms since it is dependent only upon the network properties.

In the sections which follow, different solution techniques are discussed and, where appropriate, comparison of the required machine storage is made using the indices developed here. The second index is more important since it provides the actual storage required by an algorithm in its application.

## 3.3 Gradient Projection Method

Solution techniques which use the method of projection, or gradient search methods, are often referred to as methods of "feasible directions" or "large-step gradient" methods. They are sometimes classed as linearization methods since, when applied to problems with linear constraints, they generally utilize the constraints in their solution technique as do linear algorithms. Some gradient search algorithms also use linearization techniques to approximate nonlinear constraints.

A gradient search method developed by J.B. Rosen is perhaps the best known, since readily available commercial computer codes of the algorithm exist (24, 32). Other common gradient projection methods include, the Generalized Gradient Search (GGS) program, developed by K.E. Cross and W.L. Kephart, the Davidon-Fletcher-Powell algorithm, the Method of Feasible Directions by G. Zoutendijk and the Generalized Reduced Gradient (GRG) algorithm by J. Carpentier and J. Abadic (19).

The general sequence of steps followed by all projection methods is as follows:

1) The algorithm starts with an initial feasible solution.

2) The feasible direction is determined in which a lower cost solution may be found.

3) A step of a specified or calculated length is taken in the feasible direction to find a lower cost solution which is still feasible.

Steps 2 and 3 are repeated using progressively smaller step sizes as required, until the region of uncertainty, so defined, is acceptably small. At this point, convergence to a minimum is assumed. The "feasible direction" is defined as the direction in which a small step can be taken to improve the objective function without violating a constraint. The algorithms differ primarily in the manner by which the feasible direction is calculated.

A form of the Rosen algorithm (24) is used in a practical application to test its convergence properties. This particular version of the algorithm is useful only for linearly constrained problems. As a starting point, the routine requires an initial feasible solution and initial step size both of which are to be defined by the user. The feasible direction is calculated to provide the greatest change in the objective function for a given step size, by means of normalized direction vector components which are determined from derivatives of the objective function with respect to the design variables. The resulting direction vector components are:

$$M_i = \frac{\pm \dfrac{\partial \text{Ʒ}(Q)}{\partial Q_i}}{\sqrt{\sum_{j=1}^{NQ} \left(\dfrac{\partial \text{Ʒ}(Q)}{\partial Q_j}\right)^2}} \qquad i = 1, NQ \qquad (3.1)$$

where: $M_i$ = $i^{th}$ direction vector component

Ʒ(Q) = objective function

The user must supply a subroutine providing the derivatives of the objective function with respect to each design variable. A new feasible solution with a lower cost than the initial solution is obtained by using the feasible direction and specified step size. Modifications to the basic procedure are made when certain conditions are encountered. If improvement is made to the objective function and the constraints are not violated, the step size in the next iteration is doubled. If the objective function is not improved during any iteration, the step size is halved and another attempt is made from the last successful solution. If an improvement in the objective function is obtained but one or more of the constraints is violated, the step size is determined which places the new point on the violated constraint(s). Once on the constraints (i.e. on the exterior of the convex solution set), the feasible direction is determined so that it lies along the constraints. The direction vector components are now calculated utilizing the constraints in the form of a Lagrangian.

$$M_i = \frac{\left[\frac{\partial Z(Q)}{\partial Q_i} + \sum_{k=1}^{\ell} \lambda_k \frac{\partial G_k(Q)}{\partial Q_i}\right]}{\left[\sum_{j=1}^{NQ}\left(\frac{\partial Z(Q)}{\partial Q_j} + \sum_{k=1}^{\ell}\left(\lambda_k \frac{\partial G_k(Q)}{\partial Q_j}\right)\right)^2\right]^{\frac{1}{2}}} \qquad i = 1, NQ \qquad (3.2)$$

where: $\ell$ = the number of the violated constraints

$G_k(Q)$ = $k^{th}$ constraint equation

and $\lambda_k$ $(k = 1, \ell)$ is determined from the following $\ell$ equations:

$$\sum_{i=1}^{NQ} \sum_{j=1}^{\ell} \left( \lambda_j \frac{\partial G_j(Q)}{\partial Q_i} \cdot \frac{\partial G_k(Q)}{\partial Q_i} \right) = - \sum_{i=1}^{NQ} \left( \frac{\partial G_k(Q)}{\partial Q_i} \cdot \frac{\partial Z(Q)}{\partial Q_i} \right)$$

$k = 1, \ell$

Iterative steps are continued until convergence is obtained within a limit specified by the user. In addition to the derivatives of the objective function, the user must provide subroutines to specify the objective function and any constraints. Other user specified variables include the number of independent variables, the number of constraints and a step size accuracy index (i.e. the accuracy to which the constraints are to be defined).

A sample problem is used for a brief test of the Rosen algorithm's convergence properties. The problem is identical to the three node network problem illustrated in figure 2.8. The stipulations $S_1$, $S_2$ and $D_3$ are given the values 2.2, 1.6 and 2.0 MGD respectively. The nodal states and link lengths are:

| Node | Elevation (ft.) | Link | Length (miles) |
|---|---|---|---|
| 1 | 100 | 1-3 | 4 |
| 2 | 505 | 2-3 | 5 |
| 3 | 400 | | |

In developing the objective function, the general cost equation used to calculate the conveyance costs is:

$$\text{Cost} = (15 \cdot XL \cdot Q^{\frac{1}{2}}) + (200 \cdot Q(0.004 \cdot XL + STDS - STUS)) \qquad (3.3)$$

where: XL = length (ft.)

Q = flowrate (MGD)

STDS = elevation downstream (ft.)

STUS = elevation upstream (ft.)

Using this equation, the conveyance costs for the two flow variables in the example are obtained through the following relations:

$$Q_1 \ : \ \text{Cost} = 316800 \ . \ Q_1^{\frac{1}{2}} + 76896 \ . \ Q_1 \qquad (3.4)$$

$$Q_2 \ : \ \text{Cost} = 396000 \ . \ Q_2^{\frac{1}{2}} + 120 \ . \ Q_2$$

The objective function and constraints are therefore:

$$\underset{Q^*}{\text{Min}} \ Z = 316800 \ . \ Q_1^{\frac{1}{2}} + 76896 \ . \ Q_1 + 396000 \ . \ Q_2^{\frac{1}{2}} + 120 \ . \ Q_2 \qquad (3.5)$$

$$\text{subject to:} \quad Q_1 \leq 2.2$$

$$Q_2 \leq 1.6$$

$$Q_1 + Q_2 \geq 2.0$$

Inequalities are used for the constraints to facilitate choosing alternative initial solutions. The constraints bound a convex set which contains the feasible solution set. Figure 3.3 provides a graphical representation of the problem using the same format as in figure 2.10.

An optimization package is formed by writing the subroutines which define the problem objective function, constraints and derivatives of the objective function for use by the Rosen algorithm. The program is tested for sensitivity to the initial solution by making subsequent runs with different starting points. Two of the initial solutions used are: $Q_1=2.0$, $Q_2=0.8$ and $Q_1=0.8$, $Q_2=1.4$. These starting points are located on figure 3.3 along with the paths followed by the algorithm from each. The algorithm converges to different solutions in each case demonstrating that it is sensitive to the initial solution. The Rosen algorithm therefore cannot guarantee convergence to a global solution.

A further restriction involving the use of the Rosen algorithm lies in the form of the objective function. A subroutine is required which supplies the derivatives of the objective function. This requirement can be easily met if the objective function is in an explicit form. However, in most practical cases some design procedures need to be carried out before costs can be calculated. An example of this is in the calculation of costs for a pipeline in a regional water supply network. Depending upon the nodal states, the flow of water between nodes is conveyed by gravity or under an induced head through pumping. When the link comprises a pump and pipeline, the pump and pipe sizes are not uniquely defined as a function of the flow conditions; instead, a great number of alternative solutions exist. For a given flow, small pipe sizes have high head losses which result in a larger pump size requirement. A decrease in pump size results in a saving in pumping costs but a larger pipe is required, resulting in higher costs for pipe materials and installation. A suboptimization procedure is therefore required to determine the most economical trade-off between pipe and pump sizes. If the friction head along the pipeline is the design variable, a plot of link cost versus friction head can be formed to facilitate the suboptimization.

A typical plot is shown in figure 3.4. The least costly configuration can then be found by a single variable optimization technique such as the Fibonacci method (24). This suboptimization technique provides a very accurate method of designing the links for regional water supply systems. Similar trade-offs in design may be necessary for other network systems. In using solution algorithms which require an explicit expression of the objective function, it would be necessary to calculate finite difference approximations to each derivative by multiple evaluation of the objective function, or by curve fitting techniques. When considering solution algorithms which require an explicit objective function, therefore, the possibility of needing further routines to generate the required expressions must be considered.

An advantage of using the Rosen algorithm is that no changes need to be made to the mathematical statement of the problem for its application. The algorithm would therefore fit into the model objectives well. The storage requirements of the algorithm are calculated through the use of the relation:

$$\text{Storage} = (N \times (NQ + 1)) + (4 \times NQ)$$

The storage required for different problem sizes as shown in the last section are:

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Storage Units | 570 | 1940 | 7080 | 26960 | 41700 |

The storage requirements are very similar to that of a linear formulation implying that in terms of storage only, relatively large network problems may be handled as easily as by the linear programming method.

Gradient search techniques are applicable to both convex and concave programming problems and, therefore, do not take advantage of the property that the optimal solution lies on a vertex. These methods will most likely be more expensive computationally since the surface of the feasible region is explored and thus many more steps are required to reach the solution. Furthermore, the solution obtained is not necessarily a global one but may be a local optimum.

## 3.4 Successive Linear Approximation

The method described here is a successive linear approximation method originally developed by Griffith and Stewart (15) for the solution of petroleum industry optimization problems. The method is designed to be used in problems with a concave objective function having continuous first partial derivatives and convex constraint equations. Siddal (33) has developed an algorithm, APPROX, after this method which is included in a subroutine package developed for design use. This method can be used for problems with both a nonlinear objective function and nonlinear constraints if required. The APPROX algorithm starts with a feasible point where the problem functions are approximated by an expansion in a Taylor's series about the point. Linearity is attained by dropping the nonlinear terms. The nonlinear problem can be stated mathematically in the following general form:

$$\min_{Q^*} Z = Z(Q) \qquad (3.6)$$

subject to: $G_j(Q) = 0 \qquad j = 1, N$

where: $Z(Q)$ = objective function (in terms of all flowrates $Q_i$, $i = 1, NQ$)

$G_j(Q)$ = $j^{th}$ constraint (in terms of all flowrates $Q_i$, $i = 1, NQ$)

The constraint equations are altered slightly by placing the stipulations on the left hand side. The linear formulation is then given by:

$$\min_{Z^*} Z = Z(Q^o) + \sum_{j=1}^{NQ} (Q_j - Q_j^o) \frac{\partial Z(Q^o)}{\partial Q_j} \qquad (3.7)$$

subject to:

$$G_i(Q^o) + \sum_{j=1}^{NQ} (Q_j - Q_j^o) \frac{\partial G_i(Q^o)}{\partial G_j} = 0 \qquad i = 1, N$$

where: $Q_j^o$, $j = 1, NQ$ = feasible solution elements

or, alternately:

$$\underset{Q^{*}}{\mathrm{Min}} \quad Z - Z^{o} = \sum_{j=1}^{NQ} V_j \cdot \delta Q_j \tag{3.8}$$

$$\text{subject to:} \quad \sum_{j=1}^{NQ} U_{ij} \cdot \delta Q_j = -G_i^{\ o} \qquad i = 1, N$$

$$\text{where:} \quad V_j = \frac{\partial Z(Q^o)}{\partial Q_j}$$

$$\delta Q_j = Q_j - Q_j^{\ o}$$

$$Z^o = Z(Q^o)$$

$$U_{ij} = \frac{\partial G_i(Q^o)}{\partial Q_i}$$

$$G_i^{\ o} = G_i(Q^o)$$

The problem is now in a linear form in terms of the first order corrections $\delta Q_j$ which is solvable by a linear programming algorithm. Further constraints are required in addition to the ones inherent in the problem, namely:

$$|\delta Q_j| < m_j \qquad j = 1, NQ \tag{3.9}$$

These constraints limit the changes in Q to a small amount $m_j$ since the linearization is only valid in a small interval around the specified solution. Another important point which should be noted is that $\delta Q_j$ is not necessarily positive, therefore the variables are substituted within the algorithm by the equivalent relation:

$$\delta Q_i = \delta Q_i^+ - \delta Q_i^- \qquad (3.10)$$

$$\text{where: } \delta Q_i^+ , \delta Q_i^- \geq 0$$

In using APPROX, the objective function and constraints must be made available in a subroutine form. The objective function, therefore, does not have to be explicit but can use additional design routines before calculation of the objective function value, if necessary. The variables defining the problem size must be specified as well as various parameters defining step sizes and allowable variable tolerances. The additional constraints involving the change in Q are built into the algorithm as well as the possibility of negative Q values. Implementation of the algorithm is therefore fairly straightforward due to this feature and also the fact that the same mathematical statement formulation is used as that adopted for the model (eqn. 2.10).

The array storage required by APPROX is defined by the relation:

$$\text{Storage} = ((33 + 8N + 12NQ) \times NQ) + ((7 \times N) \times N)$$

Therefore, the storage for different problem sizes as derived in the previous sections are:

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Storage Units | 24420 | 95040 | 374880 | 1488960 | 2323200 |

The storage required is significantly larger than that for the Rosen algorithm or SIMPLEX implying that the application of APPROX imposes a much lower limit on the size of problem that can be solved.

It is suspected that the execution times are similar for the Rosen and APPROX algorithms. The search is not restricted to the vertices in either algorithm and APPROX must calculate differentials upon each iteration as does Rosen. The APPROX algorithm may not require as many iterations to converge but this depends upon the allowable range, $m_j$, for Q in each iteration. The APPROX algorithm is easier to apply since it does not require a subroutine with the first derivatives of the objective function. Unfortunately, the possibility of convergence to a local optimum still exists with the APPROX algorithm.

## 3.5 Linear Separable Programming

Linear separable programming in general converts a nonlinear problem statement to a linear one so that a linear programming solution algorithm can be used. Separable programming can be used effectively to obtain approximate solutions to a very large class of nonlinear programming problems (8, 26). The validity and accuracy of linear approximation is often difficult to determine and must be dealt with carefully.

For this type of linear approximation, the objective function should be separable, that is, it must be possible to express the objective as the summation of a series of functions of a single design variable:

$$Z(Q) = \sum_{i=1}^{NQ} C_i(Q) \tag{3.11}$$

This condition holds in the case of network problems since independent cost functions are formed for each design variable.

The approximation used is called "piecewise linear approximation" or "polygonal" approximation and involves the fitting of a series of connected line segments to a given curve. Figure 3.5 illustrates how this may be done to approximate a typical concave cost curve. There are a number of variations on the formulation used to convert a nonlinear problem to a piecewise linear approximation. All, however, use the same basic method of connected linear segments.

To provide some insight to the restrictions of linear separable programming, a typical formulation is investigated. Figure 3.5 shows a typical nonlinear function of a given flowrate variable which is approximated by four line segments. The cost, or functional value, C', corresponding to a flowrate, Q', at any point on the linear segments is given by:

$$C' = C_i + \frac{C_{i+1} - C_i}{Q_{i+1} - Q_i} (Q' - Q_i) \qquad Q_i \le Q' \le Q_{i+1} \tag{3.12}$$

$$i = 0, 1, 2, 3$$

where: $C_i$ = Cost at point $Q_i$

$Q_i$ = Flow at $i^{th}$ junction joining two linear segments

$Q'$ = Flow value where $C'$ is being evaluated

By using this equation, the approximated cost can be calculated between the end points i and i+1 corresponding to the Q' value. The $C_i$ and $C_{i+1}$ values are predetermined and upon inspection, it is apparent that the fraction $(Q' - Q_i) / (Q_{i+1} - Q_i)$ is a number between 0 and 1 when Q' is between $Q_i$ and $Q_{i+1}$. Equation 3.12 can therefore be simplified in the following manner:

$$\text{let} \qquad \lambda = \frac{Q' - Q_i}{Q_{i+1} - Q_i} \tag{3.13}$$

so that: $C' = C_i + (C_{i+1} - C_i)\ \lambda$

and by letting $\lambda_i = 1 - \lambda$ and $\lambda_{i+1} = \lambda$,

equation 3.13 becomes:

$$C' = \lambda_i\ C_i + \lambda_{i+1}\ C_{i+1} \tag{3.14}$$

subject to: $\lambda_i + \lambda_{i+1} = 1$

where: $\lambda_i\ ,\ \lambda_{i+1} \geq 0$

or (the equivalent): $Q_i \leq Q' \leq Q_{i+1}$

Graphically, $\lambda_i$ and $\lambda_{i+1}$ may be seen to be weighting factors whereby C' is expressed in terms of the C values at the extremities of the line segment. Extending this notion to include all of the line segments we obtain:

$$C' = \sum_{i=0}^{4} \lambda_i C_i \qquad \text{where: } \sum_{i=0}^{4} \lambda_i = 1 \qquad (3.15)$$

$$\lambda_i \geq 0, \ i = 1, 4$$

$$\text{and} \quad Q = \sum_{i=0}^{4} \lambda_i Q_i$$

Since the $Q_i$ values are predetermined for each flow variable, the design variable Q is uniquely defined by the weighting factors $\lambda_i$. Thus the problem is restated in terms of the unknowns $\lambda_{ij}$ (i = 1 - 4 say, j = 1, NQ) by replacing each flow variable with a set of weighting factors.

Two important additional restrictions must be introduced for this formulation to be valid:

1) Not more than two of $\lambda_i$ can have nonzero values.

2) The nonzero $\lambda$'s must be adjacent.

If, by chance, the flowrate being costed is coincident with one of the intersection points, $Q_i$, i = 1, 4, the value of $\lambda_i$ is 1.0 and all other $\lambda$'s = 0.0.

Each cost function can be transformed to a piecewise linear function using the formulation of equations 3.15 but certain initial steps are required. Firstly, the linear segments must be chosen to minimize the error or differences between the real and linearized curve. This may involve graphing or otherwise preprocessing the curves as in figure 3.5. The range of each design variable must be decided upon, which in turn defines the end point of the linear segments.

In the formulation, the constraints are altered through the replacement of each flow variable, Q, by the last equation in 3.15. Each flow variable is therefore replaced by a set of new variables, $\lambda_i$, with the number in the set depending upon the number of linear segments used in the corresponding cost function. The piecewise linear formulation of the problem is solvable by a linear programming algorithm as long as certain modifications are made to ensure that not more than two $\lambda_i$ variables, which belong to the same flow variable will be nonzero and they must be adjacent. Miller (14) discusses a modified version of the SIMPLEX algorithm which can be used for this application.

It is important to notice that, in this method of formulation, the number of design variables is increased greatly and in direct proportion to the number of linear segments used in the approximation. Also, an extra constraint equation (i.e. $\Sigma\ \lambda_i = 1$) is introduced for each flowrate variable.

A linear separable method presented by G.B. Dantzig (9) uses upper bounds to replace the additional constraints required in the above method thereby not increasing the problem size. Instead of specifying the end points of the linear segments, the slopes and lengths of the segments are used to define the nonlinear functions. This method can be used only for convex programming since the slopes of successive chords on the approximated cost curve must increase with the value of the design variable. As the design variable increases, the minimum objective value is obtained by moving along the polygonal expression from the origin. Due to the increasing slope of the cost curve,points close to the origin yield optimal solutions. With a concave function, however, the slope decreases with increasing Q and the segment farthest away from the origin yields the lowest increase in cost with an increase in Q. Therefore, when using slopes for a concave objective function, the segments are not assigned sequentially and the method fails.

The main disadvantage of this optimization strategy still remains as the increase in the number of variables and constraints, hence increasing storage requirements. The size of the modified mathematical statement depends upon the number of linear segments used to represent the objective function. If it is assumed that three segments are used, then four new variables replace each original flow variable. Also, an additional constraint is introduced for each set of new variables replacing the old. The new array sizes for the mathematical statement are:

| | |
|---|---|
| Cost array: | (4NQ x 1) |
| Structural matrix: | (N x 4NQ) |
| Additional constraints: | (NQ x 4NQ) |

The modified linear algorithm required to solve the linear separable problem most likely requires additional storage to satisfy the restrictions. However, assuming that the storage requirements to not change from that of SIMPLEX, the array storage for different size problems is given by:

$$\text{Storage} = ((2 + N + NQ) \times 4NQ) + ((6 + N + NQ) \times (N \times NQ)).$$

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Storage Units | 11120 | 42646 | 170480 | 676960 | 1056200 |

The storage requirements are generally twenty times greater than for a linear formulation implying that the limiting problem size is substantially smaller.

The execution times involved are likely to be higher than in a linear problem since the introduction of additional variables and a finer grid of the polynomial expression results in an increase in the number of SIMPLEX steps to solution.

In his paper on separable programming, C.E. Miller (14) reports on the limitations of the method:

> "There is, in general, no way to show that the particular solution produced by the algorithm is a global optimum. Ordinarily one has smooth nonlinear functions, and the polygonal functions are merely approximations to local optima of the underlying smooth model, even though the objective value is nearly optimum. Estimating the extent of such discrepancies is a difficult problem, intimately related to an exhaustive sensitivity analysis of the model. The previously mentioned tricks can increase model size substantially and thus restrict the size of the problem which can be computed economically. The number of simplex steps to optimum is increased by the use of many sets of special variables, and is further increased by the use of fine grids in the polygonal approximations".

This method, therefore, cannot ensure convergence to a global optimum, but within its limitations the method has, in practice, produced local solutions which are useful.

## 3.6 Fixed Charge Algorithm

The fixed charge method is a variation of linear separable programming in which the cost function is approximated by a single linear segment plus a fixed charge. Figure 3.6 demonstrates how a concave cost curve can be approximated in this manner. The cost function is actually represented by two linear segments, one being a vertical line of length B forming the fixed cost and the second line providing a rough approximation to the slope or relationship between cost and flow. The two segments retain the properties of concavity observed with the original function.

W. Hirsch and G.B. Dantzig (9) present a mathematical formulation of the fixed charge problem for separable cost functions. The cost takes the general form:

$$C = \begin{matrix} KQ + B & \text{if } Q > 0 \\ 0 & \text{if } Q = 0 \end{matrix} \qquad (3.16)$$

This can be represented by the following formulation:

$$C = KQ + \delta\ B \qquad (3.17)$$

subject to: $Q \leq \delta\ u$

where: $\delta = 0, 1$

and $u$ = An upper bound on Q

The constraint "$Q \leq \delta\ u$" ensures that Q=0 if δ=0. When the formulation in 3.17 is used to replace the cost functions in a network problem, a new set of variables (δ) is introduced, the number being equal to the number of design flows. Two additional constraints are also introduced for each δ.

The fixed charge formulation is similar to a strictly linear problem due to the occurrence of the single "KQ" term to represent the rate of change of cost with respect to the flowrate. The only deviation from a strictly linear problem is the existence of a fixed charge B. It appears therefore that the method allows for a global solution since the objective function is planar, as opposed to concave, and that the fixed charge only introduces a variable datum, the location of which depends upon which variables are in the solution.

Although the fixed charge method appears useful for network problems, since it results in a global solution, there are inherent disadvantages which must be considered. As with separable programming, the number of variables and constraints to the problem increase which can result in storage difficulties with larger networks. More importantly, the fixed charge formulation is a very poor approximation to the actual shape of the cost functions and cannot accurately represent the curvature along the cost curves. Furthermore, because of the poor representation of cost functions, the global solution obtained may not be close to the true optimum for the problem depending upon the approximation used. Another disadvantage of this formulation is the occurrence of mixed integer programming which is more complex than real variable programming. It appears, therefore, that for network problems the disadvantages with fixed charge programming outweigh the advantage of obtaining a global solution which may not be the true optimum.

## 3.7 Iterative Linear Programming

"Iterative Linear Programming" (ILP) is a name used for a solution technique which uses a linear approximation to a nonlinear objective function and the SIMPLEX algorithm in an iterative manner to obtain a solution to concave programming problems. The storage requirements are only slightly larger than that of a linear formulation using SIMPLEX and through testing has generally been found to have good convergence properties.

The ILP algorithm was implemented in an early version of the econometric model for an application to a regional water supply network (37). The method was designed to take advantage of the relatively small storage requirements, widespread availability and computational efficiency of the SIMPLEX algorithm. Through the use of the SIMPLEX algorithm, the method confines its search to the vertices of the solution space resulting in a more economical technique.

The ILP algorithm proceeds by calculating a cost for each design variable based on an initial assumed value or, in the case of zero flowrates, a very small default value. A cost coefficient is then calculated for each design variable by dividing each corresponding cost by the real or assumed flowrate. The linear approximation to the nonlinear cost function is therefore a secant formed between the origin and the point on the cost curve associated with the assumed flowrate. With the cost coefficients representing an approximation to the original cost function, the problem is now in the general linear form:

$$\min_{Q^*} Z = \sum_{i=1}^{NQ} C_i Q_i \qquad (3.18)$$

subject to: $\bar{A} . \bar{Q} = \bar{B}$

where: $C_i$ = cost coefficient for design variable $Q_i$

and $\bar{A} . \bar{Q} = \bar{B}$ are the constraint eqns.

This linear formulation is readily solvable by SIMPLEX or an equivalent linear programming algorithm. Subroutine SIMPLE (33) is used in this case which implements a version of the revised SIMPLEX algorithm. After a solution has been obtained, the new flowrates are compared with the original values. If the flowrates differ significantly, the cost coefficients are recalculated based upon the new solution and the linear programming algorithm is again applied. This iterative procedure is used until the vector of calculated flow variables is identical, within design tolerances, to that from which the cost coefficients were calculated.

When applying ILP, a slight modification must be made to the mathematical formulation. Due to the nature of SIMPLEX, the cost coefficients calculated must represent the design variables. In collection networks, however, the quantity of material processed at a processing node, i, is given by the difference between the stipulation and the slack variable at that node, or $S_i-Q_{NQ-N+i}$. Since SIMPLEX associates the cost coefficient for processing costs with the slack variable, $Q_{NQ-N+i}$, only and not the true processing quantity, $S_i-Q_{NQ-N+i}$, the sign of the cost coefficient must be reversed. This change does not affect the solution mechanism since the stipulation, $S_i$, is a constant. The true processing quantity must be used, however, in the calculation of the cost coefficients. This modification is required only for processing nodes in distribution networks since in collection networks the processing quantity is represented by the slack variable only. Furthermore, the modification does not present any difficulties in the application of the ILP algorithm since it can be done within the algorithm itself, thereby avoiding any changes required for the basic model.

Good results were obtained in the specific study to which the model was applied. In further testing with problems of practical size, convergence was both successful and rapid. For most problems, convergence was attained in three to four iterations. As with other techniques, however, the method does not guarantee convergence to a global optimum, hence the optimum found may be a local one. It is important to realize that the local optimum exists in the nonlinear problem and that the solution to the linearized problem is global, guaranteed by the SIMPLEX algorithm. The ILP algorithm is reluctant to look beyond a local minimum due to apparently high cost coefficients assigned to a zero valued flow variable. This is due to the relatively low flowrate value used to calculate the cost coefficient. Understanding of this concept is facilitated through the use of an example. A three node distribution network is shown in figure 2.8 with its corresponding cost functions plotted in figure 3.7 (a). For simplicity, only transportation costs are included in the problem. For an initial approximation, it is assumed that $Q_1=Q_2=D_3/2$. Figure 3.7 (a) illustrates the cost coefficients for this approximation. Since $C_1$ is less than $C_2$, the linear programming algorithm chooses the solution, $Q_1=D_3$ and $Q_2=0.0$ (assume $S_1$, $S_2$, $\geq D_3$). Upon the second iteration, the cost coefficient, $C_1$, is re-evaluated using $Q_1=D_3$ and is, therefore, reduced to the value represented by the linearized cost function as shown in figure 3.7 (b). The value of $C_2$ depends upon the flow replacement value used, if $D_3/2$ is used again, $C_2$ remains unchanged. Since $C_1$ is still smaller than $C_2$, the solution determined by the linear algorithm remains the same, namely, $Q_1=D_3$ and $Q_2=0.0$. The optimal solution, $Q_2=D_3$ and $Q_1=0.0$; (see figure 3.7 (b)) cannot be found by linear programming unless the flow replacement value used for $Q_2$ is large enough to yield a $C_2$ value smaller than $C_1$. The lower limit to this flowrate, $Q_{limit}$, is located at the intercept of the nonlinear $C_2(Q)$ curve and the linear approximation to $C_1(Q)$ at $Q_1=D_3$ (see figure 3.7 (b)). This small problem illustrates the importance in choosing the flow replacement value. The sample problem also seems to suggest that the use of relatively high replacement values for flowrates equal to zero would force the solution algorithm to consider alternate solutions and make the convergence to a global optimum more likely. The use of high replacement values (values greater than the actual solution flowrates), however, introduce economics of scale which cannot be realized in the actual solution. This implies that, during the solution iterations, alternate solutions that appear less costly than the current solution may actually be more costly than the present solution. If this condition is encountered, cycling can occur between two or more closely priced solutions.

The choice of the flow replacement value for flowrates equal to zero is very important since a low value can lead to convergence to a local optimum and a high value can lead to cycling. The ILP algorithm provides two methods by which the user can specify this value.

1) The user can define a constant to be used for all zero valued flowrates.

2) The flow replacement value can be determined by the algorithm, using an averaged value, based upon the nonzero flowrates in the present solution.

If the user elects to choose the replacement value, care must be taken in using a realistic value.

The ILP algorithm uses relatively low execution times since convergence occurs after a small number of iterations and the nature of linearization of the objective function and use of SIMPLEX results in an economical operation. The storage requirements for this method are only slightly larger than that for a strictly linear formulation since the algorithm only uses one additional array (1 x NQ) in size. Through application to network problems of typical size, convergence problems were not encountered but were generally confined to problems of small size (i.e. 5 nodes, 15 links). Although the solutions converged upon may be local in nature, they were generally found to be useful (i.e. differing only slightly from the global solution).

## 3.8 Exhaustive Vertex Search

An exhaustive vertex search entails the evaluation of all useful solutions to the nonlinear problem and calculating the corresponding cost. By using a logical bookkeeping procedure, the optimal solution can be found. This method can be applied to network problems since the optimal solution lies on a vertex which is also called a basic solution (Appendix A). A procedure for determining basic solutions can easily be developed as shown in Chapter 5.

The major disadvantage of an exhaustive search is the number of solutions which must be determined and hence the number of objective function evaluations required. An approximation to the number of evaluations can be developed generally. The distinguishing property of a basic solution is the existence of N nonzero variables in the solution (Appendix A). Therefore, the number of basic solutions is given as the number of ways of combining NQ variables N at a time, C(NQ, N). However, not all of the solutions are useful and only a fraction of the total require evaluation of the objective function. What constitutes a useful solution is covered in Chapter 5. Testing has shown that the upper limit on the number of useful solutions is approximately 0.4 C(NQ, N) when the ratio of NQ to N is 4. Approximations of the number of objective function evaluations required for different problem sizes, can therefore be formed.

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Number of Evaluations | $3.4\text{x}10^{8}$ | $1.4\text{x}10^{18}$ | $3.5\text{x}10^{37}$ | $2.9\text{x}10^{76}$ | $8.8\text{x}10^{95}$ |

Although this method ensures a global solution, the execution times required for its use are too large to be practical. Assuming that each evaluation would take 0.1 seconds of computer time, even a 10 node network problem would take a year of computer time to solve. Clearly, the execution times required by the exhaustive vertex search method are impractical.

## 3.9 Conclusions

Of all the algorithms investigated, only the exhaustive search and fixed charge algorithms converge to a global solution. The optimal solution determined by the remaining solutions can be local. Unfortunately, the exhaustive search method requires impractical execution times and the global solution determined by the fixed charge algorithm may be very different from the true global optimum due to the inaccuracies in the cost function approximation. In relation to storage requirements, the Rosen and ILP algorithms are the most attractive. The APPROX and ILP algorithms are the easiest to use with the problem formulation whereas the Rosen and Linear Separable algorithms require explicit cost functions imposing a restriction on their use.

Through testing, the ILP algorithm has proved to be useful for network problems of practical size. Execution times are also found to be acceptable due to the use of the SIMPLEX linear programming algorithm and low number of iterations to solution. Since it appears most favorable, the ILP algorithm is chosen for initial use in the econometric model.

Due to the shortcomings of the ILP method, it is desirable to evolve a strategy to improve the usefulness and value of the solution method. Two main approaches are incorporated which compliment each other in producing a viable package. On the one hand, the program package is designed to facilitate a heuristic approach to problem solving so that, if the solution algorithm cannot guarantee a global solution, it is at least easy to check the feasibility and optimality of alternative solutions. In this way, confidence in the computer solution may be gained by comparing it with a number of intuitive alternatives. With increasing number of alternatives, either an improved solution is found or increased confidence in the computer solution is generated.

The other main thrust in providing a reliable solution technique is to explore the possibility of developing a new solution algorithm which hopefully combines the efficiency of the SIMPLEX method with some guarantee of global optimality. To this end, the program package is constructed on modular lines to make the substitution of alternative routines a relatively simple matter. A specific example of such an alternative algorithm is developed and discussed in Chapter 5.

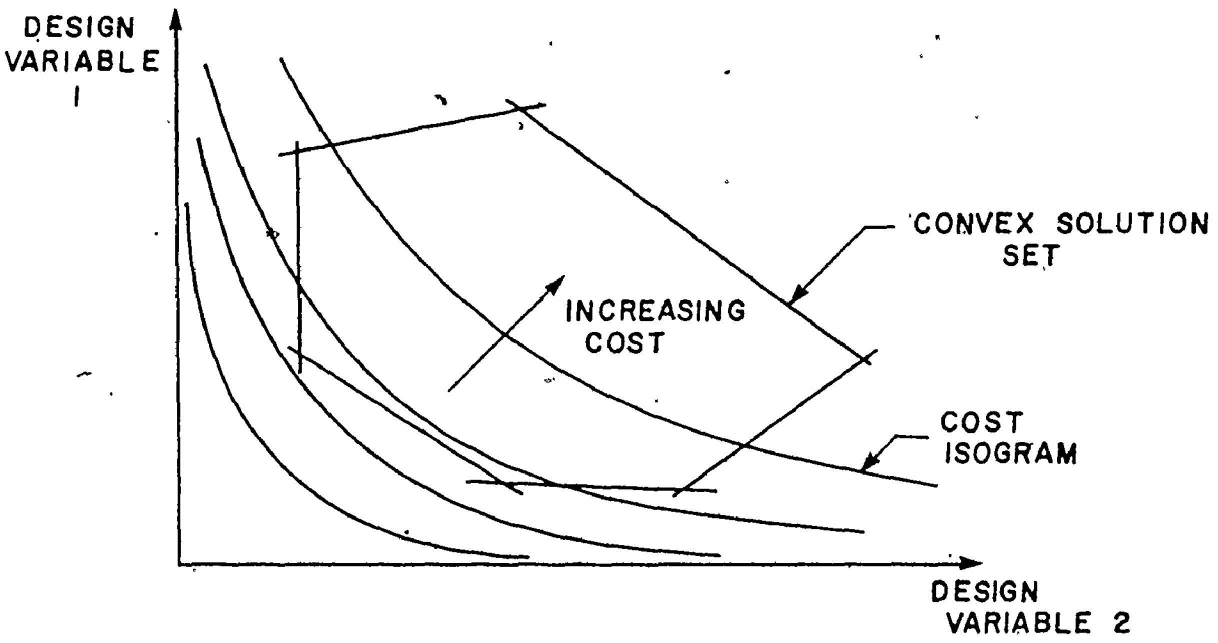


FIGURE 3·1(a) — CONCAVE PROGRAMMING PROBLEM (MINIMIZATION)

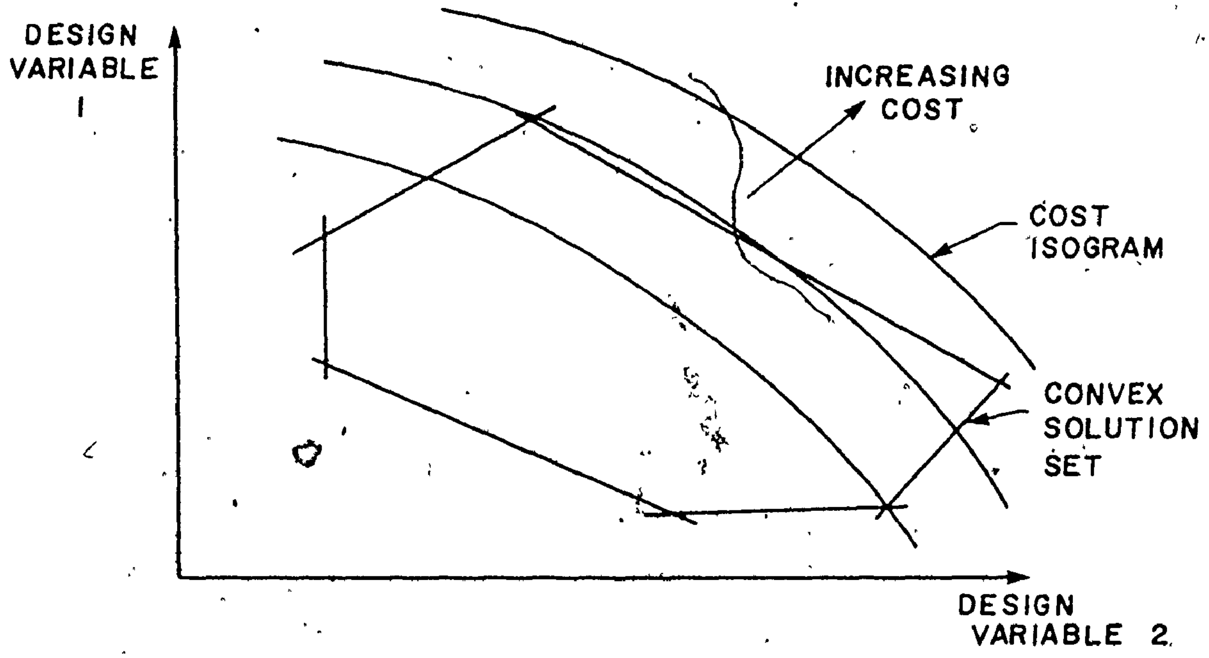


FIGURE 3·1(b) — CONCAVE PROGRAMMING PROBLEM (MAXIMIZATION)

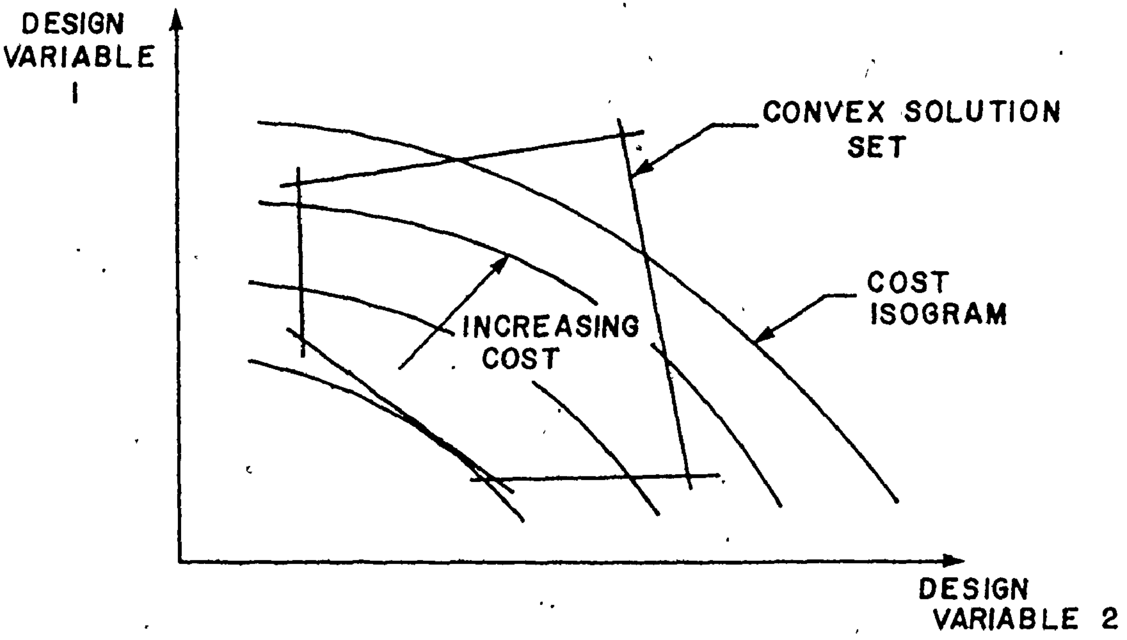


FIGURE 3·2(a) – CONVEX PROGRAMMING PROBLEM (MINIMIZATION)

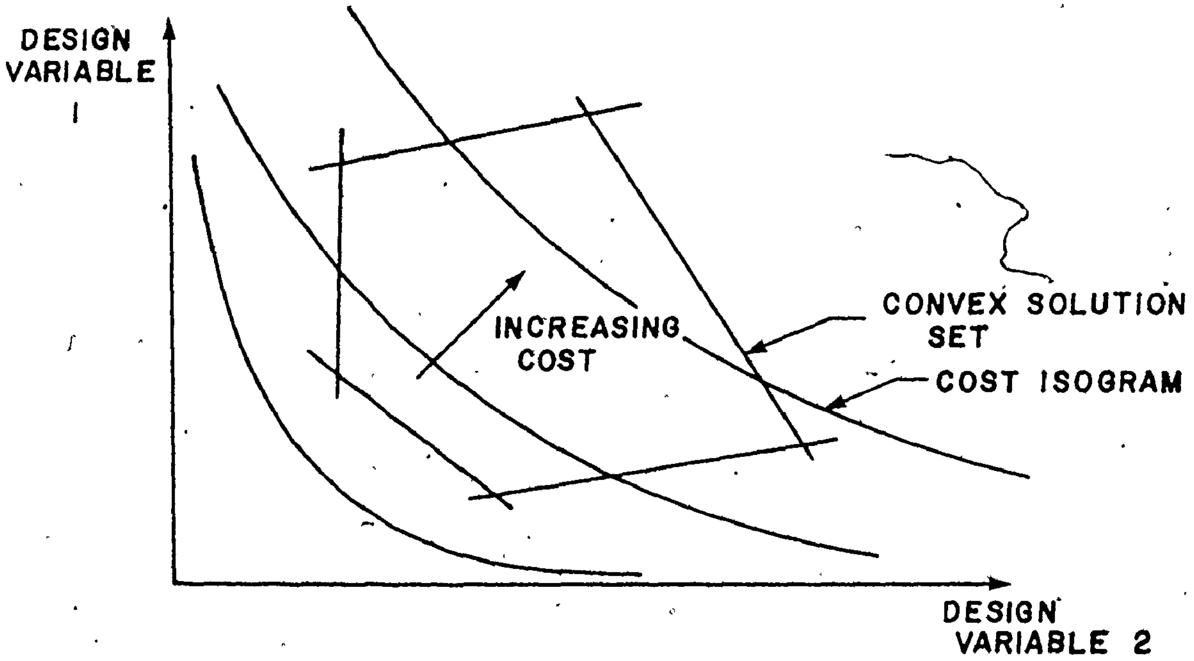


FIGURE 3·2(b) – CONVEX PROGRAMMING PROBLEM (MAXIMIZATION)

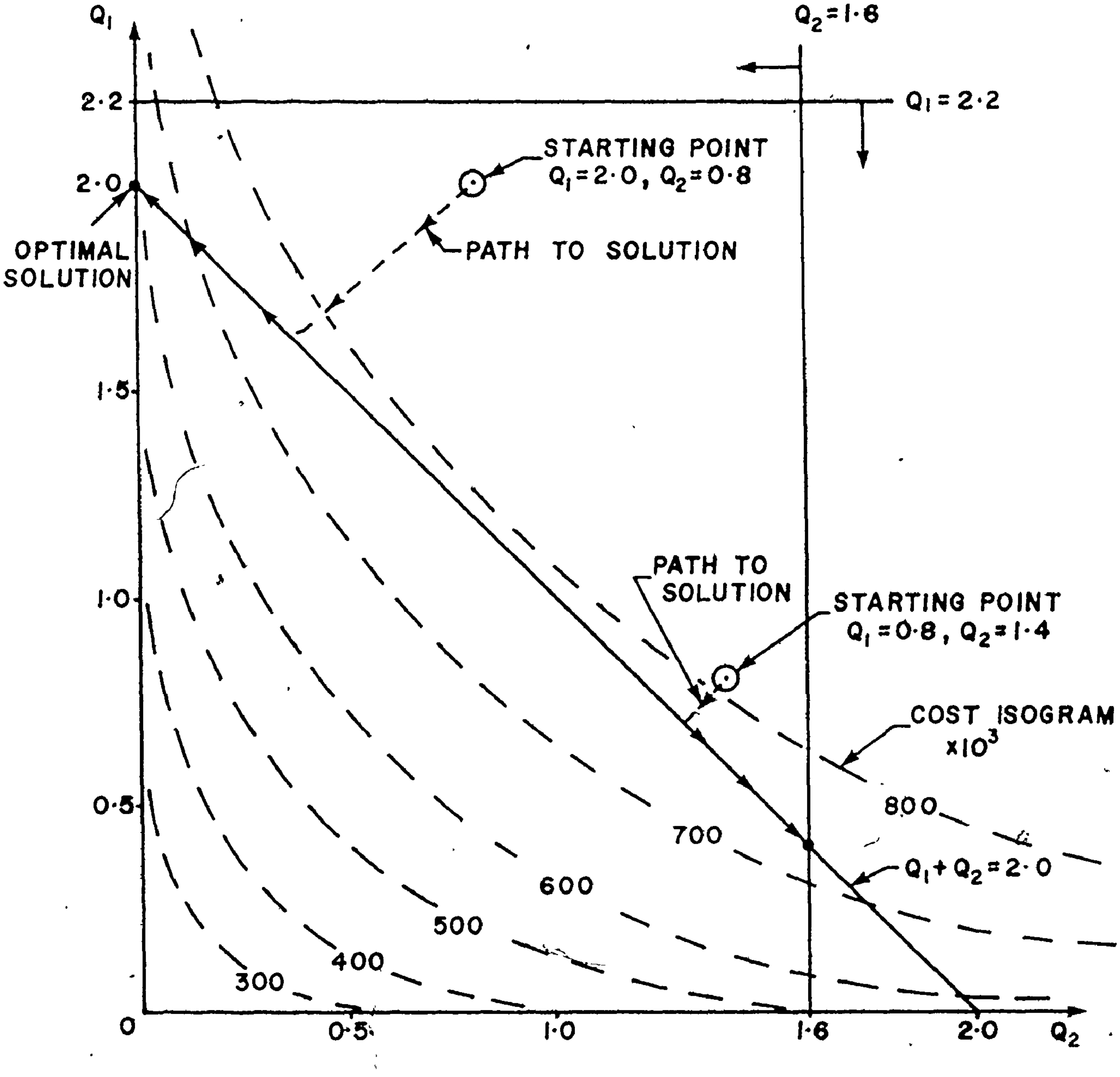


FIGURE 3·3 — SOLUTION PATHS FOLLOWED BY THE ROSEN ALGORITHM WHILE SOLVING A NONLINEAR PROBLEM

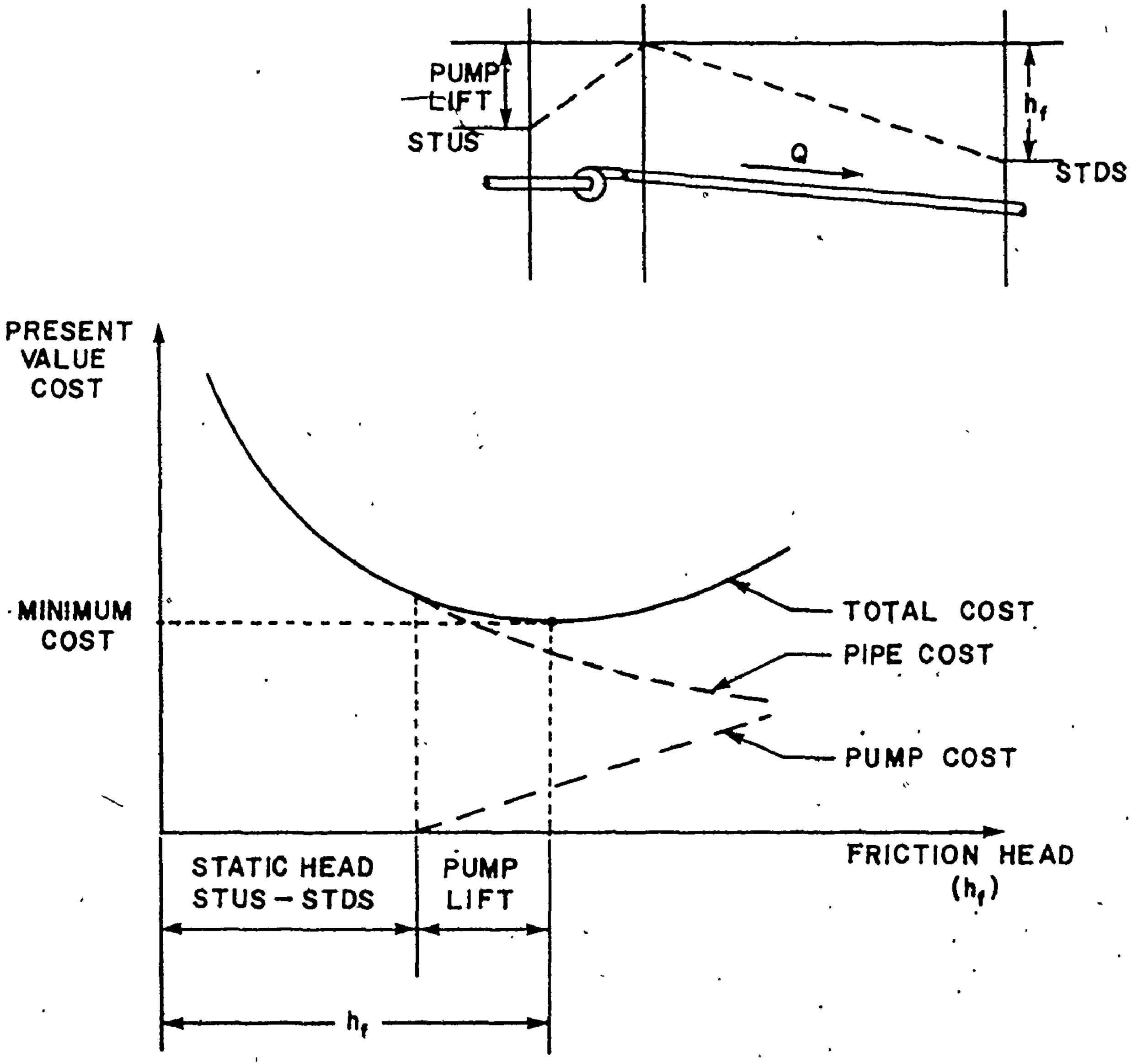


FIGURE 3·4 — SUBOPTIMIZATION OF A PIPELINE WITH A BOOSTER PUMP

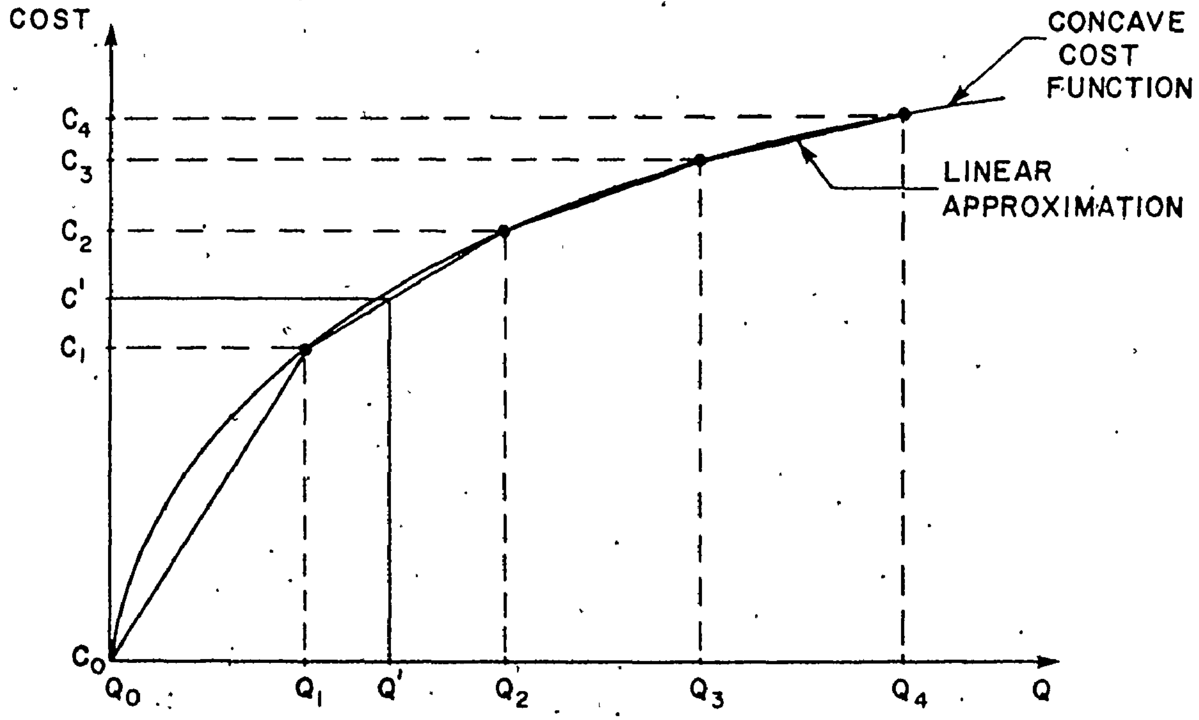


FIGURE 3·5 — PIECEWISE LINEAR APPROXIMATION OF A CONCAVE COST FUNCTION

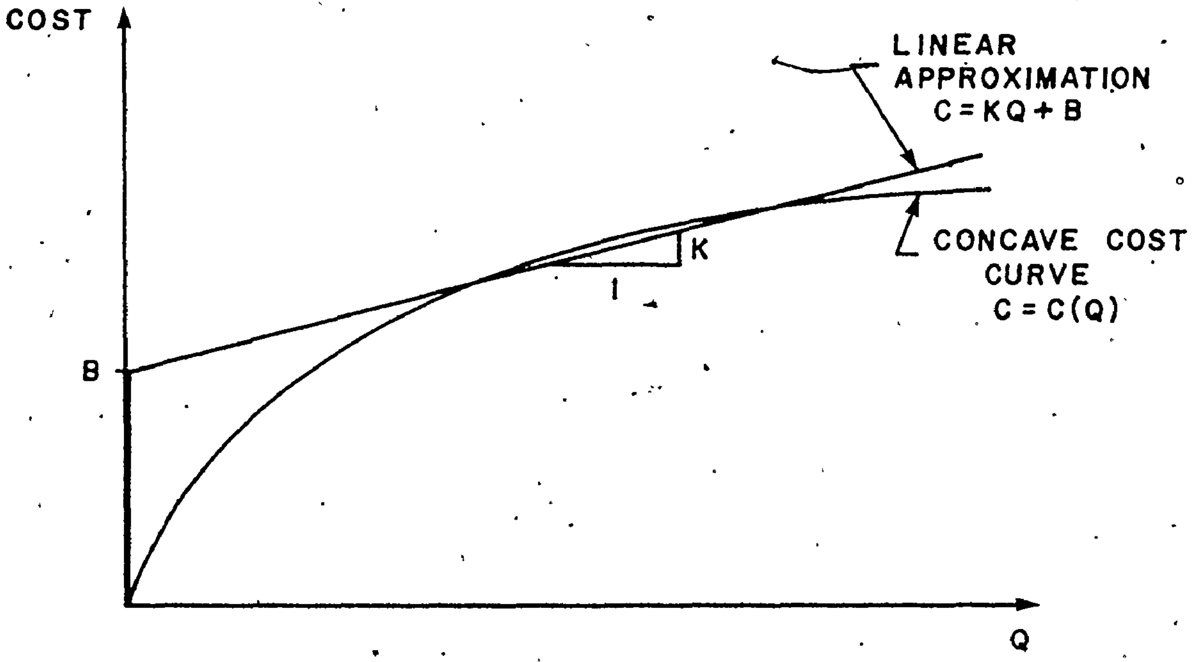


FIGURE 3·6 — LINEAR SEGMENT APPROXIMATION OF A CONCAVE COST FUNCTION

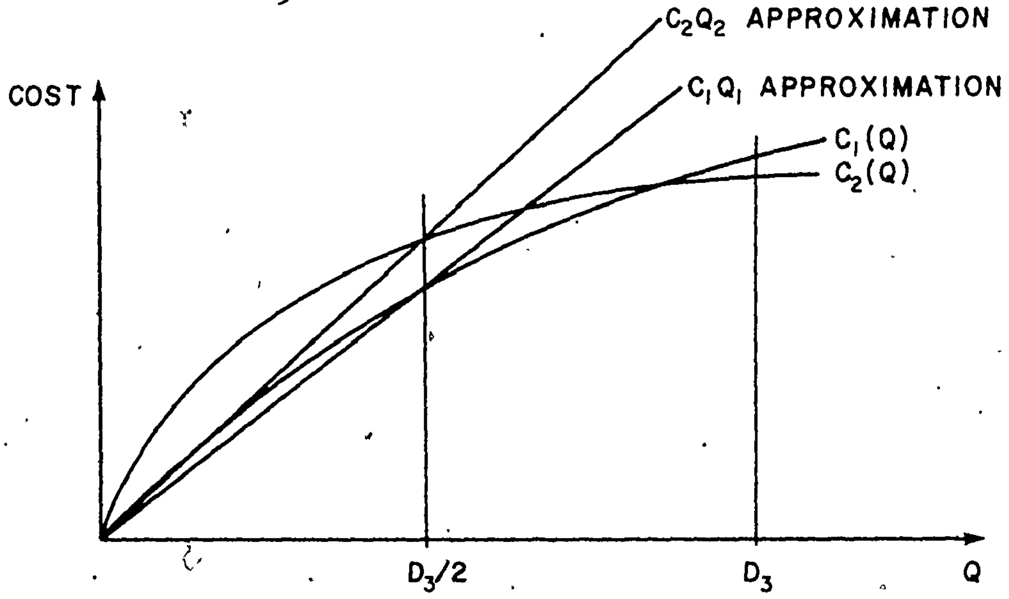


FIGURE 3·7(a) — LINEAR APPROXIMATION TO CONCAVE COST FUNCTIONS

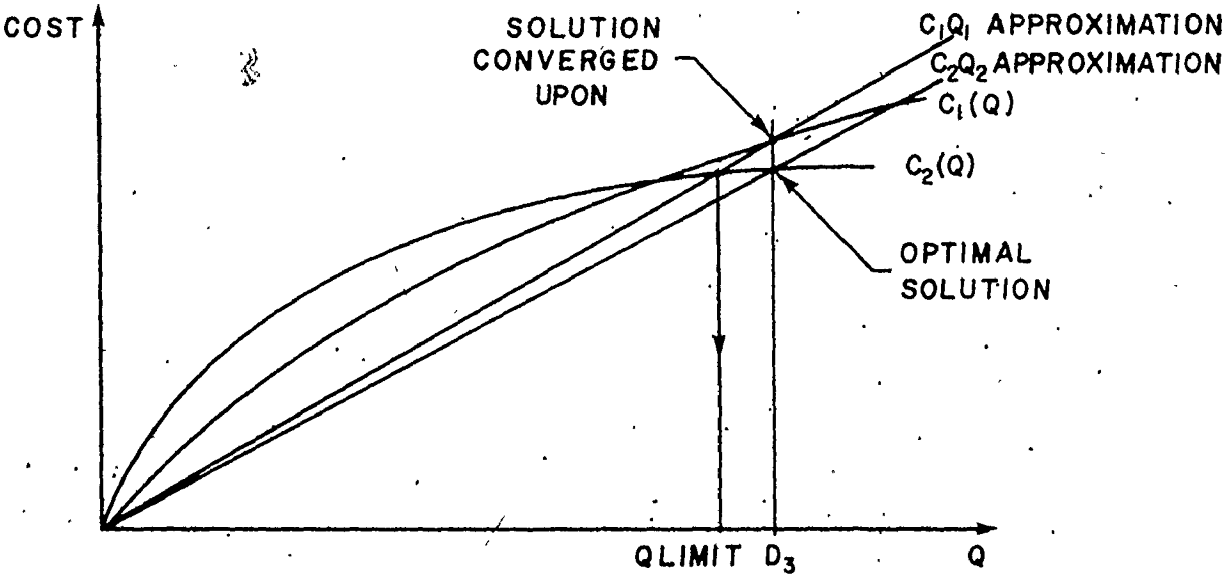


FIGURE 3·7(b) — LINEAR APPROXIMATION TO CONCAVE COST FUNCTIONS

# CHAPTER 4

# ECONOMETRIC MODEL DEVELOPMENT

## 4.1 Heuristic Approach

In this chapter, an econometric model is developed for solving network problems. The problem definition used by the model is provided by the mathematical statement developed in Chapter 2. The ILP algorithm is used as the initial solution routine to determine the optimal solution.

Before the model is constructed and tested, its objectives are summarized. In general, a model is required which allows the engineer to combine intuitive and common sense approaches to problem solving, with the capacity to solve accurately and rapidly, a large number of alternatives from which an optimal solution may be selected.

A heuristic approach is required which allows the input of engineering common sense in conjunction with an optimization algorithm to obtain accurate results. The importance of a heuristic approach is enhanced by the dilemma encountered in the last chapter when trying to locate a solution algorithm which yields a global solution, and yet, is practical to use for network problems. A.A. Kuehn and M.J. Hamburger (23) describe a heuristic program which determines the optimal location of warehouses. The solution technique is not particularly applicable to the network problems investigated in this study but the paper provides a good general description of the importance in using a heuristic approach in the construction of a model for regional network problems. Using this type of approach places the emphasis on making the model capable of answering as many practical questions as possible rather than in producing a model which is only directed towards finding the optimal solution.

A heuristic approach to problem solving is particularly valuable since it allows the user to compare the relative merits of alternative solutions which may differ only marginally in terms of objective function calculations. In this way, engineering judgment may be brought to bear on alternatives which involve intangible costs or benefits.

Specific objectives can be formed which follow the heuristic concept and, if followed, result in a model which provides answers to many questions which arise in network problems.

Model objectives:

(1) The package should enable the user to easily construct an econometric model for a distribution or collection network using arbitrarily defined cost functions. The construction of the model should require only a minimum of

development and ad hoc programming on the part of the user, but be flexible enough to allow the inclusion of complex design and cost routines if desired.

(2) The model should provide a simple means of defining the properties and topology of the network system while retaining generality of treatment. The model arrays should contain the data required to set up the mathematical statement of the problem and provide the cost functions with data common to all network problems.

(3) To provide scope for the creation of relatively complex and design-oriented cost functions, the transfer of data within the model should follow two distinct routes:

(i) The variables which are common to all network problems of this type should be transferred through subroutine parameter lists.

(ii) Variables associated with the cost functions, including any intermediate design parameters, should be stored in and transferred through labelled COMMON block storage.

In this way, all standard parameter arrays may be dimensioned dynamically by means of a user defined driving program and ad hoc parameters need appear and be dimensioned only where necessary.

(4) Through the use of time-share computing, the model should give the user the capability of changing the initial values assigned to the flow variables, providing a means of testing the sensitivity of the solution to the initial approximations.

(5) The model should give the user the capability of changing the properties or topology of the network system during a time-sharing session or between successive batch runs. The model should also facilitate modification of the network configuration to enable the user to determine the marginal cost of nodal communities. In making changes, it should be possible to alter the solution so as to quantify intangible factors (e.g. to force the adoption of a socially desirable solution by manipulation of a penalty cost representing some intangible socio-political pressure). It follows that the model should provide the choice of bypassing

the optimization algorithm to obtain a solution cost, thus allowing the user to obtain the costs of a number of different trial solutions.

(6) The package should be modular in construction to facilitate the use of alternative optimization strategies, as they become available. Modular construction also facilitates the use of user defined routines for the cost functions.

(7) The package must include simple and practical guidelines for the use of the model and, most important, clear documentation for the construction of the necessary cost functions.

The program package is developed using the set of objectives outlined above as a guideline. Consideration is also given to factors such as execution times and computer storage used.

## 4.2 The Network Package "NETSOL"

The program package is subdivided into various computational routines to provide a modular construction. The routines are chosen to each carry out a specific task or group of similar tasks. Firstly, a driving program defines the array storage required and calls an organizational routine. The organizational routine accesses the package's computational routines in the proper sequence, to make up the body of the model. One of the computational routines is used to define constants used in the program. Another subroutine sets up the mathematical statement of the program in arrays and stores other parameters defining the network in arrays. Two routines facilitate changes to data during execution, one to alter flowrate assumptions and one to modify data defining the network. Two sets of routines are required to set up the cost functions, one for processing and one for transportation costs, and calculate an array of corresponding cost coefficients. A single routine or series of routines contain the optimization algorithm, the number of which depends upon the algorithm design. Finally, a routine is included which displays the system solution in terms of design variable values and costs.

An overall flowchart of the computational sequence in the program package is illustrated in figure 4.1. A listing of the program is contained in Appendix C and a flowchart of each routine in Appendix D. A user's guide to the model is given in Appendix E, which outlines the purposes and describes the structure of the program. The required format for input data is shown and a sample of the output display is provided. Further discussion of the program routines is contained in the following sections to facilitate their understanding.

### 4.2.1 Program NETWRK

This is the driving program for the package. Throughout the subroutines, dynamic dimensioning is used for the arrays which are common to all network problems and hence transferred through parameter lists. The arrays are dimensioned explicitly in the driving program and the allotted array sizes must, therefore, be adequate for the network problem being solved.

The input peripheral channel numbers are defined with the channels being:

(1) the file containing the data defining the network,

(2) the peripheral channel from which the interactive program commands are read, and

(3) the channel which the program uses for display of output.

The program reads the parameters defining the size of the network, the number of nodes, number of processing centres and the number of links, from the file containing the network data. The program title is then printed on the output file and the organizational subroutine, NETSOL, is called.

### 4.2.2 Subroutine NETSOL

This subroutine first sets up the dynamic allocation of the array storage. The routine then calls each of the computational routines in the proper order, namely, subroutine CNSTIN, DEFINE, QINIT, MODIFY, SOLVE, and REPORT (see figure 4.1). After execution passes through subroutine REPORT it moves into subroutine CNSTIN for a second run. In the second and subsequent runs, execution bypasses subroutine DEFINE. The cycle is repeated until execution is terminated in subroutine QINIT or MODIFY.

The computational sequence can be altered in subroutine MODIFY by specifying a command (BACK) to divert execution back into subroutine QINIT. Also, if an error is detected in the optimization routine, SOLVE, execution bypasses subroutine REPORT and proceeds to QINIT for data changes.

### 4.2.3 Subroutine CNSTIN

This is a user supplied subroutine which facilitates the input of variables which are not common to all network problems and are, therefore, not defined through the main input routine, DEFINE. This data could include design information required in the cost routines such as interest rates, convergence criteria, flow resistance parameters, etc. The variables do not appear in the parameter list, but are transferred through labelled common block. The use of labelled common block makes available or transfers between user defined portions of the package (i.e. COST1, COST2, REPORT and CNSTIN), data unique to a particular design application. This facilitates application to different network problems without requiring program changes. If only a small amount of data is required for the cost functions, an alternate method of supplying it could be accomplished by introducing the additional value using data statements or simple assignment statements in the cost routines.

Subroutine CNSTIN also facilitates the definition of the flowrate constant QLIMIT. This quantity is used in the program to provide a limiting flowrate below which flowrates are insignificant to the problem. The need for this parameter is discussed in later sections. QLIMIT plus the input and output peripheral numbers, being common to all problems, comprise the parameter list. This routine is particularly useful if the model user wishes to program interactive statements for use during time-sharing sessions in which certain of the cost function parameters are to be varied between successive runs. To facilitate the parameter changes, the main computational loop in NETSOL includes subroutine CNSTIN. A YES/NO command option allows the bypassing of input parameters and avoids the repetition of input when not required.

### 4.2.4 Subroutine DEFINE

Subroutine DEFINE is the main input routine. It reads the data defining the network from a prescribed data file or input device. The data must be written on the data file in the format as shown in Appendix E. This data is divided into two main catagories, (1) nodal definitions and (2) link definitions.

(1) The nodal definitions consist of the nodal numbers, names, states (e.g. elevation) and stipulations which, upon input, are stored in arrays. As mentioned previously, the nodes must be numbered so that the process-

ing nodes are given the initial designations to facilitate the construction of the structural matrix. The nodal data can be read in any order, however, since the node numbers are included in the data.

(2) The next set of data read and stored in arrays defines the network links. Each line of data defines the upstream and downstream nodal numbers of a link, the upstream and downstream link lengths and upstream and downstream initial flow approximations. It is assumed that flow may occur in either direction between nodes and hence the "upstream-downstream" designation is arbitrary. Two flow variables are therefore defined in each line of data. The flow variable numbering is dictated by the order of the data defining the links, more specifically, the first line of data defines the first and second flow variable, the second line defines the third and fourth variable and so on. The capability of being able to specify different lengths for a link in its upstream and downstream direction can be useful when the link properties are directionally dependent. If an empty field is

> left where the second link length is to be specified, the program assumes that the upstream and downstream lengths are equal. The upstream and downstream initial flow approximations are read next, at least one of which must be zero in a feasible solution. Empty fields are acceptable at these locations, implying zero valued initial flowrates. This requirement need not be strictly observed in the initialization, depending upon the optimization algorithm requirements.

The arrays containing the origin-destination pairs for the flow variables are used to set up the structural coefficient matrix. This matrix along with the array of stipulations represents the linear constraints of the problem. The stipulations are checked to determine if the network is of the collection or distribution type (i.e., if negative stipulations exist, it is a distribution network; if not, it is a collection network). When the network classification is determined, a flag is set which is used by other routines to identify the type of network problem. Structural coefficients are assigned to the structural matrix for slack and artificial slack variables, the sign of which depends upon the type of network. A perturbation technique is applied to the stipulation array, changing zero stipulations to a small insignificant value. A perturbation value of ± 0.1 x QLIMIT is used, where QLIMIT is a limiting flowrate defined in CNSTIN. The sign of the value conforms to the sign of the nonzero stipulations of the same group. This value can be changed easily by the user by altering QLIMIT in subroutine CNSTIN. The perturbation technique is used to avoid degeneracy, the consequences of which are discussed in Chapter 5 (see page 161).

The slack and artificial slack variables are then assigned values to establish continuity at the nodes with respect to the initial values set to the other flow variables. This forms a continuous initial solution which can be used in subsequent routines.

### 4.2.5 Subroutine QINIT

This subroutine facilitates making changes to the initial flowrate assumptions, along with offering other flowrate related interactive command options. Upon entering the subroutine, the allowable command options are printed out and the user is invited to select one of the options by typing in the appropriate code. The allowable commands are:

| | | |
|---|---|---|
| GO | ........ | TO PROCEED TO NEXT ROUTINE |
| LIST | ........ | TO LIST NONZERO FLOWRATES |
| ALTER | ........ | TO CHANGE INITIAL FLOWRATES |
| SOLVE | ........ | TO OBTAIN A PROGRAM SOLUTION USING THE FLOWRATES PRESENTLY CONTAINED IN THE FLOW ARRAY |
| STOP | ........ | TO END RUN |

Each command is further described in Appendix E.

When obtaining a listing of the nonzero flowrates, the slack variables are listed as well as the design flowrates. This aids the user in checking continuity in the currently stored values. The slack variables are identifiable by having only a single node number printed with the corresponding flowrate.

The ALTER command facilitates testing the sensitivity of the optimal solution to the initial solution specified. Also, when used in conjunction with the SOLVE command, it provides for the calculation of costs for specified trial solutions. Certain feasibility checks are performed by the routine to check user supplied data. If the number of changes requested exceeds the number of nodes in the network, a message is printed requesting the user to respecify the number of changes. When each set of node numbers and flowrate is specified, the upstream and downstream node numbers are checked against the possible nodal pairs stored in the program arrays. If a nonexistent nodal pair is specified, a message is printed out requesting the user to retype the last change. Finally, after all of the changes have been made, continuity is checked at each node. If continuity is not established, a diagnostic is printed out warning the user of the error. It is left to the user to make additional changes to ensure continuity, if it is required. The only part of the model which may require a continuous solution, however, is the optimization algorithm and continuity may not be required depending upon the technique used.

The ALTER command allows the user to specify slack variable values by typing in the corresponding node number as the U/S node, a zero for the D/S node and the required flowrate following. This capability can aid the user in specifying a solution which is continuous.

The SOLVE command prompts the calculation of solution costs without having to go through the optimization technique. Prior to calculating the solution cost continuity is checked and, if not met, the slack variables are recalculated to establish continuity. This ensures that the processing quantities are correct when the costs are calculated. A further check is made to guard against a slack variable calculated as being negative. A negative value results in an infeasible solution and, if encountered, execution diverts back to the beginning of the routine after printing an appropriate diagnostic.

Subroutine QINIT is intended mainly for time-sharing sessions but may also be used in batch mode if care is taken. In batch mode, difficulties may arise through the need for additional input if data errors are detected. The additional input required is easily accommodated through time-sharing use. The user may enter any number of the allowable commands while in this routine until terminated by either a GO or STOP command.

### 4.2.6 Subroutine MODIFY

This subroutine allows the properties or topology of the network systems to be modified during program execution. The allowable commands are:

| | |
|---|---|
| GO | TO PROCEED TO NEXT ROUTINE |
| LINKS | TO DELETE OR RESTORE LINKS |
| NODES | TO DELETE OR RESTORE NODES |
| STIP | TO CHANGE STIPULATIONS |
| STATE | TO CHANGE NODAL STATES |
| BACK | TO GO BACK INTO "QINIT" |
| STOP | TO END RUN |

A description of each command is provided in Appendix E.

The LINKS command is useful for testing the sensitivity of the network solution to changes in the network configuration. Upon the specification of link modifications, the input data is checked as in subroutine QINIT and error messages printed out if errors are detected. After each deletion (or restoration) is requested, the subroutine checks to ensure that the link is not already deleted (or restored); if it is, an appropriate diagnostic message is printed requesting the user to respecify the change. Any link deleted or restored using this command is directional and, therefore, if the flow from node A to node B is prevented by deleting the link connecting nodes A and B, flow can still exist from node B to node A unless that directional link is deleted as well. The term "restored" is used since this command cannot be used to introduce links which are not defined in the initial network data file, but can only restore links which have been deleted through previous use of the LINKS command. The subroutine deletes a link by introducing a large penalty cost (1.0 x $10^{10}$), which is added to the cost of the design variable associated with the deleted link. If the link is later restored, the appropriate penalty cost is reduced to zero.

The NODES command is useful in determining the marginal cost of nodal communities by enabling the user to delete a community from the network. As with the LINKS command, the input data is checked for feasibility and compatibility with previous commands. A node is deleted from the system by eliminating the processing capability, demand or supply as appropriate. Deletion of a node, in this manner, is accomplished by lowering its stipulation value to the perturbation value used by subroutine DEFINE for junction nodes (0.1 x QLIMIT). The reason for not reducing the stipulation to zero is covered in Chapter 5 (see page 161). In this way, the node is still connected to the network but only as a junction node.

As discussed in Chapter 2, the mathematical statement of the problem is constructed so that null, or junction nodes, are grouped with the nonprocessing nodes. If a processing node is deleted using the NODES command, the resulting junction node is in the processing node group. This does not affect the mathematical statement of a distribution network since the processing node deleted becomes a supply node with an insignificant amount of available supply. In a collection network, however, the stipulation represents the amount of material to be collected and processed. Reducing the stipulation in this case does not restrict the processing capabilities. To restrict processing at that node, the routine assigns a penalty cost to the slack variable associated with it, which is subsequently added to the processing cost of the node.

To restore a node, the routine changes the appropriate stipulation back to its original value and zeros any penalty cost assigned. As with the LINKS command, a check is made to ensure that the node restored was defined in the network initially and was deleted by previous use of the NODE command.

The STIP command is similar to the NODES command since its use results in the alteration of the nodal stipulation values. With the STIP command, however, the user can change the stipulation to any value required. As with other commands, the number of changes requested and the nodal numbers specified are checked for feasibility. The stipulation values given are checked for the proper sign (i.e. negative for nonprocessing nodes in distribution networks and positive for all other nodal types). If the stipulation sign is incorrect, an error message is printed requesting the user to respecify the last change.

Through the use of the STIP command, the user can reduce a stipulation to zero. As mentioned previously, zero stipulations may not be desirable so, upon input of a zero value, a diagnostic message is printed warning the user that zero stipulations may cause instability in the solution routine.

The STATE command allows the user to alter this property of the network. For example, in a water supply network, this command can be used to change the pressure elevation at the nodes which is useful for investigating the effects of consolidating necessary pumping to a smaller number of locations or in smoothing out the pressure elevation along a pipeline connecting a number of nodes. The input data for this command is, of course, checked for feasibility.

The BACK command is useful in obtaining solution costs and designs for the network by allowing access to subroutine QINIT. After modifications have been made through the use of other commands in this routine, execution may be diverted to QINIT where the SOLVE command is available. This can be done since the changes made by subroutines QINIT and MODIFY are not local to each routine but are transferred to the rest of the model via the subroutine parameter lists.

As with subroutine QINIT, this routine is intended primarily for time-sharing use but batch mode can also be used if required. A particularly important aspect of the routine lies in the use of the BACK command. The BACK command allows the user to alternate between the QINIT and MODIFY subroutines to change initial flow assumptions, make changes to the network topology or properties and obtain a number of network designs and costs for the modified parameters. In this way, the package may be employed for simulation studies rather than optimization.

### 4.2.7 Subroutine SOLVE

The objective of subroutine SOLVE is to determine the optimal (minimum cost) policy for a problem having a nonlinear cost function subject to linear constraints. A variety of possible techniques were investigated in Chapter 3 with the Iterative Linear Programming method being chosen as the most effective. A new nonlinear solution algorithm, which is also quite useful is presented in Chapter 5. The solution algorithm is constructed in a subroutine form so that it can be easily replaced if the user wishes to use alternate methods, thus facilitating the construction of a library of solution algorithms which can easily be interchanged in the model. This method is useful since different algorithms can have widely varying properties (i.e. convergence, storage requirements, execution times, etc.).

The ILP algorithm is set up in subroutine form as an initial solution algorithm for the package. Two options are available to the user when using the ILP routine. The first is to obtain intermediate printout of cost data during the algorithm iterations. When requested, the routine prints out the flow variable numbers, upstream and downstream nodal numbers, flowrates and costs for all flow variables during each iteration. When intermediate printout is not requested, only the iteration numbers are printed during execution. The second option gives the user the choice of using the variable QLIMIT to replace the zero valued flowrates or an averaged flowrate as calculated by subroutine REPLCQ. The significance of this choice is discussed in Chapter 3 (see page 101).

To avoid the possibility of looping, the number of iterations is limited to five. If convergence is not obtained after five iterations, the routine prints a message asking the user to type in "YES" if five more iterations are wanted. The message is repeated after each set of five iterations until the user types a "NO". This procedure is not normally required since, in practice, it is found that convergence usually occurs after three or four iterations. The variable, QLIMIT, is used to define the allowable tolerance of the optimal solution. If, during a program iteration, the flowrates in the present solution differ from the flowrates in the solution from the previous iteration by less than QLIMIT, it is assumed that the optimal solution has been reached. When the optimal solution is found, execution returns to subroutine NETSOL.

Subroutines LNKCST and CENCST are called to calculate the cost coefficients for the design variables. Both routines are described in detail in later sections. The linear programming solution used in the algorithm is subroutine SIMPLE (33) which contains a form of the revised SIMPLEX linear programming algorithm. At McMaster University, this subroutine is supplied by the computer system library, OPTISEP.

### 4.2.8 Subroutine LNKCST

This routine organizes the calculation of real and artificial costs associated with transportation in the network links. The costs are calculated by subroutine COST1 for each flow variable which represents flow through a link. The calculated cost is divided by the respective flow to form an element in the cost coefficient array. The nonzero artificial (or penalty) costs, which can arise from the use of the LINKS command in MODIFY, are added to the appropriate cost coefficients.

### 4.2.9 Subroutine COST1

Subroutine COST1 is a user supplied subroutine which calculates the cost of transportation in the network links as a function of the flowrate, link length and upstream and downstream state variables. In accordance with the theory utilized in the model, the cost functions must be concave and separable. As mentioned in section 1.2, special cases may arise where an upper portion of the cost curve is convex (e.g. where the capacity of a transportation route is exceeded). Construction of cost functions of this type should be avoided and if cases are encountered where the capacity of a transportation link may be exceeded, special measures could be built into the routines to output a warning.

The required subroutine calling statement and parameter definitions for COST1 are given in Appendix E. Additional design data may be transferred in or made available to other user supplied routines through the use of labelled common block as mentioned in the discussion of subroutine CNSTIN. Typical use in this routine would include the access of parameters defined in CNSTIN and transference of design data, such as pump data and pipe head losses in water supply networks, to subroutine REPORT for subsequent printing in the final output.

### 4.2.10 Subroutine CENCST

Subroutine CENCST organizes the calculation of real and artificial cost and cost coefficients associated with nodes. The routine first calculates the quantities of material processed at the processing nodes (i.e. stipulation minus the slack variable for distribution networks and the slack variable for collection networks). Subroutine COST2 is then called to calculate the corresponding costs for each processing node. The cost coefficients are then calculated and any nonzero artificial costs are added, forming the elements of the cost coefficient arrays for the processing nodes. A large penalty cost is set to the elements of the cost coefficient array corresponding to the nonprocessing nodes (i.e. artificial slack variables). This large coefficient is required by the solution algorithm used in Chapter 5.

### 4.2.11 Subroutine COST2

Subroutine COST2 is a user supplied routine which calculates the processing cost for a processing node as a function of the quantity of material processed. As with transportation costs, the processing cost functions must be concave and separable. The subroutine calling statement and definitions of the parameters are given in Appendix E. Labelled common block can be used to transfer in any additional data required in the design calculations (i.e. from subroutine CNSTIN). Similarly, data determined during the course of the design can be transferred to other user defined routines such as subroutine REPORT.

As mentioned previously (section 2.3), in special cases, the designer may wish to allow for the transshipment of untreated material between processing nodes for subsequent treatment at the downstream node. An example of this could be found in water supply networks where two or more water sources are required but it may be more economical to treat the water at a single processing node. General provision is not made for amassed treatment in the model since it is not a property that is common to all network problems. The COST1 routine, however, can be easily designed to handle this condition. It is a simple matter to identify the flow between processing nodes, and once determined the treatment costs could be calculated accordingly. To facilitate the determination of the transshipment quantities, the flowrates and possibly nodal number arrays for the link ends must be transferred into COST1 through common block or the parameter list.

### 4.2.12 Subroutine SOLCST

Subroutine SOLCST organizes the calculation of costs for a specified solution and subsequently calculates the total cost. The subroutine accesses LNKCST and CENCST routines to calculate the cost coefficient array and in turn uses it to determine the total cost. With the package using the ILP algorithm for optimization, this routine is used only in subroutine QINIT for the SOLVE command.

Subroutine SOLCST can, however, be used for optimization algorithms which require objective function evaluations during their optimization procedure. This is encountered in Chapter 5.

#### 4.2.13 Subroutine REPORT

Subroutine REPORT organizes the output of information on the solution design and solution costs. The subroutine is constructed to enable the output to be augmented by the user, facilitating the print-out of additional data.

The standard output of the routine comprises three parts:

(1) Nodal data - This includes data for both the processing and nonprocessing nodes. For the processing nodes, the nodal name, quantity of material processed and cost are written on the output file for each node with a processing quantity greater than or equal to QLIMIT. The same procedure is followed for the display of nonprocessing node data. The amount of material processed at a nonprocessing node, however, is artificial and does not appear in the optimal solution since these quantities are forced out by high artificial costs. The capability of printing nonprocessing node data is provided in case the user assigns a nonzero quantity to the

artificial slack variables at a non-processing node either intentionally or inadvertently when specifying a solution. Following the use of the SOLVE command in QINIT, subroutine REPORT will display, along with the other data, the nonprocessing node data, providing a check for the user.

(2) Transportation data - The data displayed in this section includes the upstream node number and name, downstream node number and name, flowrate and cost for each link in the network having a flow greater than or equal to QLIMIT for the present solution.

(3) Summary costs - This portion displays the total processing costs, total nonprocessing node costs (if any), total transportation costs and overall system costs.

Following the standard output section, or between the three portions of standard output, the routine may be augmented by the user to provide a display of design oriented data as required. This data would most likely consist of design variables generated during the course of calculating cost coefficients and which could be transferred into REPORT through the use of labelled common block. Alternatively, costs may be broken down into capital and annual components. The user can easily enter any output statements required without having to modify the standart portion of the routine.

Additional routines may also be used to replace or supplement the output provided by REPORT. One such routine involves the display of the network solution in a graphical manner by plotting the nodal points and using linear segments to form the links containing the non-zero solution flowrates. Alternate solution presentation methods are not described in detail in this study but are left as an opportunity for further work.

## 4.3 Network Package Application

A sample application of the program package as outlined in the previous sections is provided in Appendix F. The network considered is a thirteen node water supply network. The network is first defined and the data file is set up for model use. The cost functions are then constructed for inclusion in COST1 and COST2. No additional data is defined through CNSTIN and the standard output for REPORT is not augmented. When the package is applied, few of the available commands in subroutines QINIT and MODIFY are utilized since the example is only intended to provide an illustration of the overall package usage. Fuller usage of the command options is made in Chapter 6 when practical applications of the model are investigated. The output from two interactive runs are included in the Appendix.

## 4.4 Conclusions

In the absence of a universal solution algorithm which is economical, has good convergence properties and uses an acceptable amount of storage, it is very desirable to develop a heuristic model which allows the input of engineering common sense in problem solving. Prior to package development, the model objectives are outlined. The package is developed in modular form to allow the substitution or augmentation of certain portions of the package such as the optimization routine or the subroutine which displays the results. The main routines and their purposes are illustrated in figure 4.1.

The routines have a number of interactive commands (especially QINIT and MODIFY) to facilitate user input during execution. This approach makes time-sharing use of the program very attractive, but makes the preparation of decks, for batch runs more demanding in any but the most straightforward application.

Various checks, such as checks for continuity and feasibility, are made in many routines to detect most data or user errors before they create problems in execution. An attempt has also been made to utilize standard Fortran code in the program to aid in the transportability between computer systems. To date, the package has been used successfully on the following computer systems: CDC 6400, CDC 7600, ICL 1904S and HP 3000 ((for use on the ICL 1904S compiler, certain program changes were required for the interpretation of interactive commands).

The model is applied to a 13 node water supply network as an illustration. The problem has relatively straightforward cost functions which exhibit economies of scale. The model employed uses the ILP algorithm for optimization. Two model applications are made, one without specification of an initial solution and the second with specification. Although convergence to a solution is rapid in both cases, the final solutions differ somewhat. However, the cost difference is small and either would be acceptable.

With a computer model, using a heuristic approach, being successfully developed in this chapter, the next step is to explore the possibility of developing a new solution algorithm which combines the efficiency of the SIMPLEX method with some guarantee of global optimality. This course of action is followed in Chapter 5.

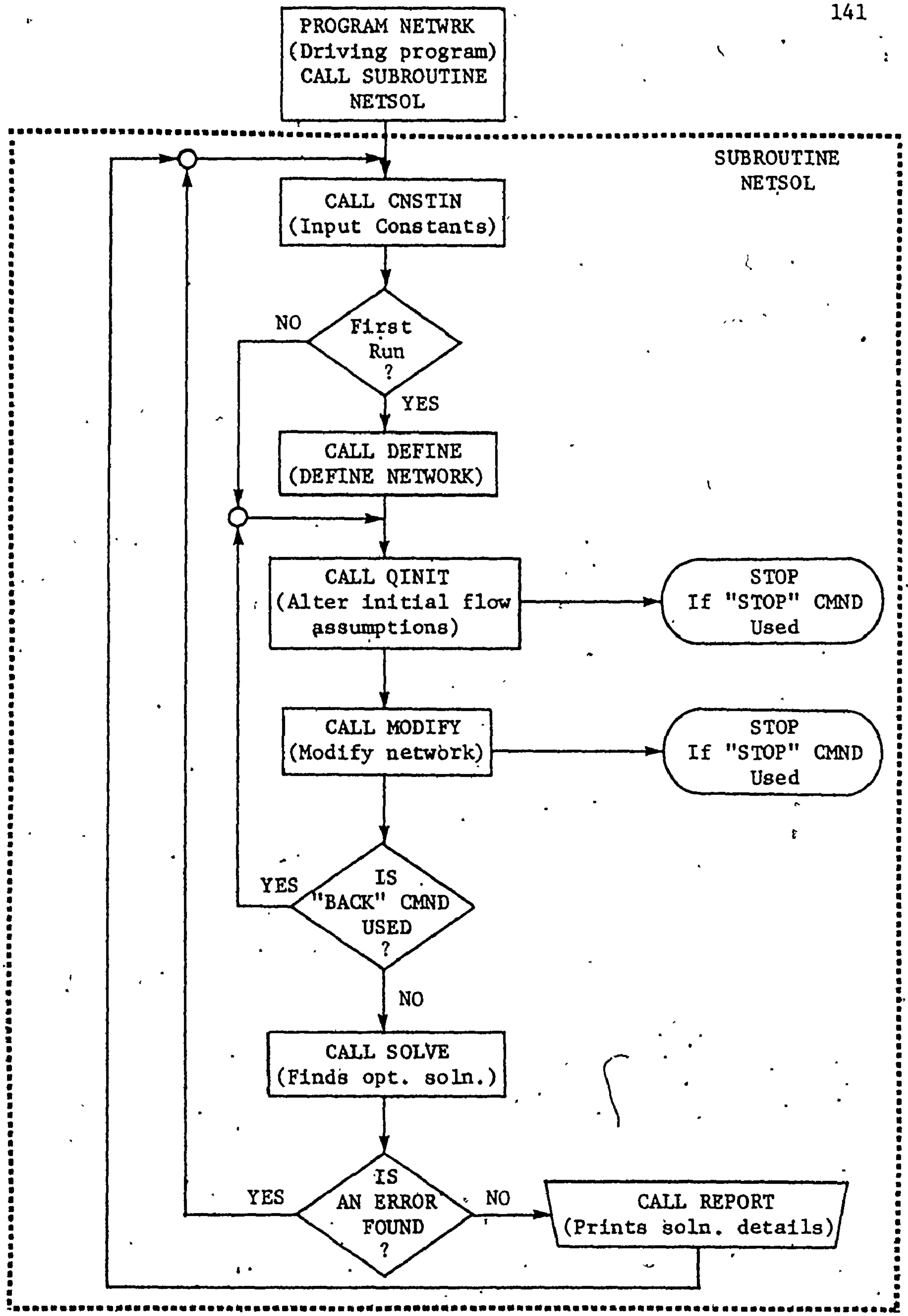


FIGURE 4:1 — OVERALL FLOWCHART OF THE NETSOL PACKAGE

# CHAPTER 5

# A NEW OPTIMIZATION ALGORITHM

## 5.1 Nonlinear Problem Characteristics

In Chapter 3, some of the disadvantages of existing nonlinear algorithms became apparent, namely, large array storage requirements, costly execution times, work involved in modifying the problem formulation to fit the algorithm and unreliability in finding the optimal solution. It therefore appears worthwhile to develop a solution algorithm which is designed to take advantage of the special characteristics of a network problem in an attempt to find an algorithm with good convergence properties and which, hopefully, is also economical.

Prior to the development of a solution algorithm, the characteristics of the problem may be summarized as follows:

1) The objective function is separable, made up of the summation of concave functions of single variables. Thus the objective function is itself strictly concave.

2) The problem constraints are linear equations (converted from inequalities through the use of

slack variables), which form a convex solution set (Appendix A).

3) The matrix of structural coefficients comprises 0, 1 or -1 elements and is normally sparse. This quality implies that matrix manipulation can be quick and efficient in specially designed algorithms.

The first two properties dictate that the optimal solution of the problem is a basic feasible solution (i.e. not more than N non-zero variables in the solution, where N = number of constraint equations). The proof of this is supplied in Appendix B. An interesting aspect of a basic solution is that it represents a network reduced to one or more branching or tree networks, as described in Chapter 2. This form of network has no redundant links. A basic feasible solution also represents a vertex of the feasible solution space (see Appendix A).

With the optimal solution lying on a vertex of the solution set, it appears obvious that the most accurate solution technique would involve using an organized search technique to compare the objective function at the vertices of the convex solution set. With its search being limited to the vertices, a technique of this type would be computationally economical to use. Direct search algorithms, applicable to nonlinear problems, such as the Hooke and Jeeves technique (24) and the Rosen algorithm exist but neither of these limit their search to the vertices. The SIMPLEX technique limits its search to the vertices of a solution set but, of course, cannot be applied to a network problem except through the use of a linear approximation technique such as the ILP method. It should be worthwhile, therefore, to develop a similar vertex-to-vertex searching technique which can be applied directly to network problems.

## 5.2 Vertex-to-Vertex Optimization Technique

The vertex searching strategy must provide for:

a) The determination of an initial basic feasible solution as a starting point, and

b) A means of moving to successive basic feasible solutions in search of an improvement in the objective function.

The initial basic feasible solution can be either program generated or user defined for added versatility. The search of successive vertices should follow a logical pattern to ensure convergence to an optimum and be computationally efficient..

To facilitate movement from one basic feasible solution to another, a "variable swapping" method is used, that is, a variable in the basic is taken out and a nonsolution variable inserted. This method uses a two step approach:

(1) A trial variable from the (NQ - N) variables not in the basic feasible solution must be selected for inclusion in the solution.

(2) A method must be devised to identify which of the variables in the basic should be discarded (i.e. reduced to zero) to form a new basic feasible solution.

In linear problems, the selection of the trial variable which yields an improvement in the objective function may be made through the use of the cost coefficients of the design variables. The SIMPLEX method utilizes a technique of this type. This is possible since the coefficients represent a constant marginal cost for each variable with respect to flowrate. Unfortunately, in nonlinear network problems, the cost coefficients are functions of the design variables and cannot be used in the same manner. Each nonbasic variable must be considered in turn and the basic feasible solution used in the next step of the search pattern is selected on the basis of the objective function values. A method is therefore required which can be utilized to identify the variable to be discarded from the basic feasible solution when a nonbasic variable is introduced, thereby facilitating movement from one basic solution to another. The method adopted is referred to as the "θ Rule" which is also used in the SIMPLEX method (3).

### 5.2.1 Method for Determining Alternate Basic Solutions - The θ Rule

The θ Rule states that, given an initial basic feasible solution, a new basic feasible solution can be formed by bringing in a nonbasic variable if one of the structural coefficients associated with the variable introduced is positive in the reduced structural matrix. The reduced structural matrix is obtained by transforming the structural matrix to its canonical equation form where the unit coefficients for the solution variables appear along a diagonal and all off diagonal elements are zero. The θ Rule is developed, below to facilitate explanation as to how it is implemented in a solution algorithm.

Recall, from Chapter 2, that the constraint equations can be stated generally in matrix form as follows:

$$\bar{A} \cdot \bar{Q} = \bar{B} \qquad \text{where: } \bar{A} = \text{N x NQ matrix of structural coefficients} \qquad (5.1)$$

$$\bar{Q} = \text{Flow variable vector}$$

$$\bar{B} = \text{Stipulation vector}$$

When an initial basic feasible solution is chosen, the variables can be renumbered so that the first N variables are in the basic. A revised matrix formulation can then be written as:

$$\bar{D} \cdot \bar{Q}_N = \bar{B} \tag{5.2}$$

where: $\bar{D} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$

and: $\bar{Q}_N = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ \cdot \\ \cdot \\ Q_N \end{bmatrix}$

where: $\bar{D}$ = The revised matrix of the first N structural coefficients

$\bar{Q}_N$ = Solution variable vector

Using matrix algebra, the solution to equation 5.2 is found through augmentation of the matrices:

$$(\bar{D} \mid \bar{B}) \tag{5.3}$$

Multiplying through by the inverse matrix $\bar{D}^{-1}$ an identity matrix is obtained on the left hand side, i.e.

$$(\bar{D}^{-1} \bar{D} \mid \bar{D}^{-1} \bar{B}) = (\bar{I} \mid \bar{B}') \tag{5.4}$$

$$= \left[\begin{array}{cccc|c} 1 & 0 & \ldots & 0 & b_1' \\ 0 & 1 & \ldots & 0 & b_2' \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ 0 & 0 & \ldots & 1 & b_N' \end{array}\right]$$

Therefore, we obtain:

$$\bar{Q}_N = \bar{B}' = \bar{D}^{-1}\bar{B} \tag{5.5}$$

or

$$\bar{Q}_N = \begin{bmatrix} Q_1 \\ Q_2 \\ \cdot \\ \cdot \\ \cdot \\ Q_N \end{bmatrix} = \begin{bmatrix} b_1' \\ b_2' \\ \\ \\ \\ b_N' \end{bmatrix} \tag{5.6}$$

In order to ensure a feasible solution (i.e. one with nonnegative values), the elements of the modified vector of stipulations, $\bar{B}'$, must all be positive. In practice, this is easily accomplished by requiring that the initial vector, $\bar{B}$, is also free of negative elements, which can be done by multiplying rows by (-1) where necessary.

The nonbasic variable, $Q_K$, which is to be introduced into the basic, is brought in by augmenting the column of the $\bar{A}$ matrix which corresponds to $Q_K$, this column vector is denoted as $\bar{P}_K$:

$$\bar{P}_K = \begin{bmatrix} a_{1K} \\ a_{2K} \\ \cdot \\ \cdot \\ \cdot \\ a_{NK} \end{bmatrix} \tag{5.7}$$

Using the augmentation operation:

$$(\bar{D} \mid \bar{P}_K \mid \bar{B}) \tag{5.8}$$

and using matrix algebra,

$$(\bar{D}^{-1}\bar{D} \mid \bar{D}^{-1}\bar{P}_K \mid \bar{D}^{-1}\bar{B}) = (I \mid \bar{P}_K' \mid \bar{B}') \tag{5.9}$$

$$= \left[\begin{array}{cccc|c|c} 1 & 0 & \dots & 0 & a_{1K}' & b_1' \\ 0 & 1 & \dots & 0 & a_{2K}' & b_2' \\ & & \cdot & & \cdot & \cdot \\ & & \cdot & & \cdot & \cdot \\ & & \cdot & & \cdot & \cdot \\ 0 & 0 & \dots & 1 & a_{NK}' & b_N' \end{array}\right]$$

For simplicity set $Q_K = \theta$ and the new equation set is formed:

$$\begin{aligned} Q_1 + a_{1K}'\,\theta &= b_1' \\ Q_2 + a_{2K}'\,\theta &= b_2' \\ &\cdot \\ &\cdot \\ &\cdot \\ Q_N + a_{NK}'\,\theta &= b_N' \end{aligned} \tag{5.10}$$

where: $\theta = Q_K$

A new solution is obtained by transposing the elements, $a_{1K}'\,\theta$, to the right hand side, yielding the vector arrays.

$$\bar{Q}_{N+1} = \begin{bmatrix} Q_1 \\ Q_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ Q_N \\ Q_K \end{bmatrix} = \begin{bmatrix} b_1' & - a_{1K}' \theta \\ b_2' & - a_{2K}' \theta \\ & \cdot \\ & \cdot \\ & \cdot \\ & \cdot \\ b_N' & - a_{NK}' \theta \\ & \theta \end{bmatrix} \tag{5.11}$$

The new solution $\bar{Q}_{N+1}$ in (5.11) is made basic by choosing $\theta$ so that one of the variables, $Q_i$, $i = 1, N$, becomes zero, hence leaving the solution; while simultaneously ensuring that none of the terms $b_i' - a_{iK} \theta$ become negative. This condition is met by choosing $\theta$ so that:

$$\theta = \min \frac{b_i'}{a_{iK}'} \quad \text{for} \quad a_{iK}' > 0 \tag{5.12}$$

This forms the $\theta$ Rule.

### 5.2.2 Utilizing the θ Rule

For a new solution to be basic, a variable must leave the solution as a nonbasic variable, $Q_K$, enters. It can clearly be seen that if the $\theta$ value is not chosen using the $\theta$ Rule in (5.12), the new basic solution could have negative values, resulting in an infeasible solution. The variable, $Q_K$, can enter the basic solution only when one of $a_{iK}'$ is greater than zero, otherwise the value of a solution variable could not be reduced to zero and the new solution would not be basic. The physical significance of this condition is discussed later in the chapter (see page 157).

When a new basic feasible solution is formed by increasing θ to its full value as defined by the θ Rule, it is effectively the same as moving from a vertex of the convex solution set along the intersection of the hyperplanes formed by the constraints to another vertex. At any point between the two vertices both the variable being introduced to the solution, $Q_K$, and the variable leaving the solution have nonnegative values. As movement towards the new vertex proceeds, the value of $Q_K$ increases to θ and the value of the variable leaving the solution is reduced to zero. By introducing each of the nonsolution variables in turn, thereby forming new basic solutions, all of the vertices joined to the vertex representing the initial solution by the hyperplane intersections are evaluated. This forms a basis for the optimization technique as outlined in the following steps:

1) To provide a starting point, a basic feasible solution is chosen as an initial solution and its corresponding objective function cost calculated.

2) In turn, each nonbasic variable is introduced, if possible, into the initial solution and the θ Rule is used to form a set of trial basic feasible solutions.

3) The nonlinear objective function is re-evaluated for each trial basic feasible solution.

4) If the objective function values for the trial solutions are all greater than the initial solution, then the initial solution is the optimum and the preliminary search ends.

5) If any of the trial solution costs are less than that of the initial solution, the one with the lowest cost is chosen as the new basic feasible solution, replacing the initial solution in step 2.

Steps 2 through 5 are repeated until an optimal solution is found in step 4. At this point, the vertices surrounding the optimum all have higher costs.

To facilitate understanding of the optimization technique described above, an example is provided. The example also illustrates the different types of solutions which can be encountered.

### 5.2.3 Example Using Vertex Searching Method

The network system used for this sample problem is a 5 node distribution network as illustrated in figure 5.1. Nodes 1 and 2 are processing nodes and the remainder are consumer nodes. The processing capacities and consumer demands are as follows:

| Processing Node | Processing Capacity |
|---|---|
| 1 | 15.0 |
| 2 | 10.0 |

| Consumer Node | Consumer Demand |
|---|---|
| 3 | 9.0 |
| 4 | 6.5 |
| 5 | 8.0 |

The data defining the network system is given in table 5.1.

The objective function, for demonstrative purposes, is based upon transportation costs only, using the following concave function for each design variable:

$$\text{Link Cost} = (15 \cdot XL \cdot Q^{\frac{1}{2}}) + (200 \cdot Q(0.004 \cdot XL + STDS - STUS)) \qquad (5.13)$$

where: XL = Link length (ft.)

STDS = Downstream elevation (ft.)

STUS = Upstream elevation (ft.)

A penalty cost coefficient of $10^{10}$ is applied to the artificial slack variables to force them out of the solution. The general form of the objective function is therefore:

$$\min_{Q^*} Z = \sum_{i=1}^{NQ-N} (C'_i Q_i^{\frac{1}{2}} + C''_i Q_i) + \sum_{i=NQ-N+3}^{NQ} 10^{10} Q_i \qquad (5.14)$$

where: $C'_i = 15.0 \cdot XL$

and: $C''_i = 200.0\ (0.004 \cdot XL + STDS - STUS)$

The constraint equations are formed using the network system diagram, figure 5.1, with the signs in the consumer node constraints changed, to meet the nonnegativity restrictions in the θ Rule.

$$\begin{array}{lr} Q_1 - Q_2 - Q_5 + Q_6 - Q_9 + Q_{10} - Q_{17} + Q_{18} + Q_{19} = 15.0 & (5.15) \\ Q_{11} - Q_{12} + Q_{13} - Q_{14} - Q_{15} + Q_{16} + Q_{17} - Q_{18} + Q_{20} = 10.0 & \\ Q_1 - Q_2 - Q_3 + Q_4 + Q_{11} - Q_{12} + Q_{21} = 9.0 & \\ Q_3 - Q_4 - Q_5 + Q_6 - Q_7 + Q_8 + Q_{13} - Q_{14} + Q_{22} = 6.5 & \\ Q_7 - Q_8 - Q_9 + Q_{10} - Q_{15} + Q_{16} + Q_{23} = 8.0 & \end{array}$$

The constraints are shown in matrix form in table 5.2.

To begin the solution method, an initial basic solution is specified. The most straightforward method to use in specifying an initial solution is to equate the slack and artificial slack variables to the stipulations. By using this method, a reduced structural matrix is already established and the search method quickly forces the artificial slack variables out of the solution due to the high penalty costs. It is much more economical, computationally, to choose an initial solution which appears close to the optimal solution since the number of steps required to reach the optimal solution is reduced. The initial solution chosen in this case is:

$$Q_4 = 9.0$$
$$Q_8 = 15.5$$
$$Q_{16} = 23.5$$
$$Q_{18} = 13.5$$
$$Q_{19} = 1.5$$

The structural matrix is reduced by using row operations. Table 5.3 contains the reduced matrix formed by the following row operations, carried out in the order indicated:

(1) Row 4 = Row 3 + Row 4

(2) Row 5 = Row 4 + Row 5

(3) Row 2 = Row 5 - Row 2

(4) Row 1 = Row 1 - Row 2

The circled elements of the reduced structural matrix (table 5.3) correspond to the variables in the solution. The next step in the solution technique is to evaluate the objective function for the initial solution which is, in this case, $Z = 8.037 \times 10^6$. A trial solution is generated by bringing a nonbasic variable into the solution and deleting a solution variable using the θ Rule. The first nonbasic variable used is $Q_1$. Using the nomenclature of equation 5.11, the new solution is developed:

$$\begin{bmatrix} Q_{19} \\ Q_{18} \\ Q_4 \\ Q_8 \\ Q_{16} \\ Q_1 \end{bmatrix} = \begin{bmatrix} 1.5 - 0 \times \theta \\ 13.5 - 1 \times \theta \\ 9.0 - 1 \times \theta \\ 15.5 - 1 \times \theta \\ 23.5 - 1 \times \theta \\ \theta \end{bmatrix} \qquad \theta = \min \frac{b_i^{r}}{a_{i1}^{r}} = 9.0$$

Therefore, $Q_4$ leaves the solution and the new basic solution and corresponding objective function are as follows:

$$\begin{aligned} Q_1 &= 9.0 \\ Q_{19} &= 1.5 \\ Q_{18} &= 4.5 \\ Q_8 &= 6.5 \\ Q_{16} &= 14.5 \\ Z' &= 6.575 \times 10^6 \end{aligned}$$

This procedure is carried out for each remaining nonbasic variable, the next being $Q_2$. A new solution is developed using:

$$\begin{bmatrix} Q_{19} \\ Q_{18} \\ Q_4 \\ Q_8 \\ Q_{16} \\ Q_2 \end{bmatrix} = \begin{bmatrix} 1.5 - 0 \times \theta \\ 13.5 - (-1) \times \theta \\ 9.0 - (-1) \times \theta \\ 15.5 - (-1) \times \theta \\ 23.5 - (-1) \times \theta \\ \theta \end{bmatrix}$$

In this case, all $a'_{i2}$, i = 1, N are less than zero. As mentioned previously, a trial solution cannot be formed when this occurs, since the resulting solution would not be basic. It is interesting to note, however, that if θ is given any positive value a feasible, although nonbasic, solution results. In network problems, this condition results from a closed loop being set up which joins two or more nodes. The flow within the loop can increase without limit and not require additional input to the system. The above case is illustrated in figure 5.2, where $Q_2$ completes the closed loop which joins nodes 1, 2, 5, 4 and 3. It is easily seen that θ can be increased without limit and the resulting solution remains feasible. Since looping solutions do not represent a desirable form of trial solution, the nonbasic variables with negative structural coefficients are not used to form a trial solution. There are at least N possible looping solutions for each basic feasible solution (i.e. one for each flow variable opposite in direction to the solution variables). The number of possible looping solutions greater than N depends upon the configuration of the network. The existence of looping solutions decreases the number of possible trial solutions to something less than NQ - N but also decreases the execution time involved in each step to the optimum.

Each of the remaining nonsolution variables are used to develop trial basic solutions when possible and the objective function evaluated for each. The artificial slack variables are not considered, however, since they should not be in the final solution. The following table summarizes the result of introducing each trial nonsolution variable in turn:

| Variable | Objective function value |
|---|---|
| $Q_1$ | $6.575 \times 10^6$ |
| $Q_2$ | Looping solution |
| $Q_3$ | Looping solution |
| $Q_5$ | Looping solution |
| $Q_6$ | $7.41 \times 10^6$ |
| $Q_7$ | Looping solution |
| $Q_9$ | Looping solution |
| $Q_{10}$ | $8.50 \times 10^6$ |
| $Q_{11}$ | $7.772 \times 10^6$ |
| $Q_{12}$ | Looping solution |
| $Q_{13}$ | $8.42 \times 10^6$ |
| $Q_{14}$ | Looping solution |
| $Q_{15}$ | Looping solution |
| $Q_{17}$ | Looping solution |
| $Q_{20}$ | $8.155 \times 10^6$ |

The functional value obtained when $Q_1$ is brought into the solution (and $Q_4$ deleted) is lower than any other found during this iteration and also lower than the original objective function value. Therefore, this solution is taken as the new basic feasible solution and the other is discarded. Row operations are performed on the structural matrix and stipulations to reduce the matrix to the canonical form containing the new solution. This is the end of the first iteration.

The second and subsequent iterations proceed as the first, introducing the nonsolution variables to form a set of trial solutions and find a less costly solution than the initial solution. The least costly solutions and their objective function values found during the iterations are summarized in the table below.

| Iteration | Least Costly Solution Found | Cost |
|---|---|---|
| 1 | $Q_1 = 9.0$<br>$Q_8 = 6.5$<br>$Q_{16} = 14.5$<br>$Q_{18} = 4.5$<br>$Q_{19} = 1.5$ | $6.575 \times 10^6$ |
| 2 | $Q_1 = 13.5$<br>$Q_3 = 4.5$<br>$Q_8 = 2.0$<br>$Q_{16} = 10.0$<br>$Q_{19} = 1.5$ | $6.011 \times 10^6$ |
| 3 | $Q_1 = 15.0$<br>$Q_3 = 6.0$<br>$Q_8 = 0.5$<br>$Q_{16} = 8.5$<br>$Q_{20} = 1.5$ | $5.819 \times 10^6$ |

| Iteration | Least Costly Solution Found | Cost |
|---|---|---|
| 4 | $Q_1 = 15.5$ | $5.784 \times 10^6$ |
| | $Q_3 = 6.5$ | |
| | $Q_{16} = 8.0$ | |
| | $Q_{17} = 0.5$ | |
| | $Q_{20} = 1.5$ | |
| 5 | All trial solutions are more costly than the initial solution (shown in 4). | |

A solution less costly than the one found in the fourth iteration is not found in the fifth, hence the least costly solution of the fourth iteration is the optimal one. The optimal solution is illustrated graphically in figure 5.3.

In this example, the solution method converged upon the optimal solution in five iterations, implying that it has good convergence properties; however, further testing is required since the initial solution is very close to the optimum. During the five iterations, 24 objective function evaluations are carried out. In larger network problems, this can result in significantly long execution times depending upon the calculations required in evaluating the objective function. Again, further testing is required to find how substantial the time required is. In Chapter 7, methods are discussed whereby the computational effort is minimized (see page 249).

### 5.2.4 Degeneracy

Although not encountered in this example, degeneracy can arise in some cases. A degenerate basic feasible solution is a basic solution with one or more zero valued components. The example in the section above can be used to illustrate degeneracy. If the demand at node 4 is 6.0 instead of 6.5, a feasible solution very similar to the optimal solution is formed by setting $Q_{17}$=0.0 and changing the other solution variables to suit. This is not necessarily the optimal solution to the altered network but it is close to the optimum and would most likely be encountered during the course of the search technique. Since there are only four nonzero elements in the solution ($Q_1$=15.0, $Q_3$=6.0, $Q_{16}$=8.0, $Q_{20}$=2.0), it is a degenerate feasible solution. It is reported (3, 9) that degeneracy can lead to cycling in the SIMPLEX algorithm but can be avoided by applying a perturbation technique (i.e. adding a very small quantity to the stipulations). Degeneracy has not been found to lead to cycling in the algorithm presented in this chapter but the solution mechanism, at times, encounters nonfeasible basic solutions. This occurs when the $K^{th}$ column of the reduced matrix associated with the nonsolution variable $Q_K$ has a positive structural coefficient in the same row as the zero valued stipulation. Therefore, if θ is given a value, the previously zero valued variable in the solution is assigned a new value of (-θ), resulting in a nonfeasible solution. An example of this is shown using the altered network discussed above. Consider the feasible solution:

$$\begin{aligned} Q_1 &= 15.0 \\ Q_3 &= 6.0 \\ Q_{16} &= 8.0 \\ Q_{17} &= 0.0 \\ Q_{20} &= 2.0 \end{aligned}$$

If the K$^{th}$ column ($Q_K$ not in the solution) of the modified structural matrix is:

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

and the stipulation vector and corresponding variables are:

$$\begin{bmatrix} Q_{20} \\ Q_3 \\ Q_1 \\ Q_{17} \\ Q_{16} \end{bmatrix} = \begin{bmatrix} 2.0 \\ 6.0 \\ 15.0 \\ 0.0 \\ 8.0 \end{bmatrix}$$

then the new solution is found:

$$\begin{bmatrix} Q_{20} \\ Q_3 \\ Q_1 \\ Q_{17} \\ Q_{16} \\ Q_K \end{bmatrix} = \begin{bmatrix} 2.0 \\ 6.0 - \theta \\ 15.0 - \theta \\ 0.0 - \theta \\ 8.0 + \theta \\ \theta \end{bmatrix} \qquad \therefore \quad \begin{aligned} \text{Min } \theta &= 6.0 \\ Q_K &= 6.0 \\ Q_{20} &= 2.0 \\ Q_1 &= 9.0 \\ Q_{17} &= -6.0 \\ Q_{16} &= 14.0 \end{aligned}$$

This is clearly an infeasible solution which must be disregarded by the search technique. If θ is allowed to equal zero, the minimum θ value would then be zero. In this case, however, the solution would remain unchanged except for $Q_K$ equaling zero and $Q_{17}$ removed. Clearly, a solution of this type is of no significant difference than the initial solution since the solution cost remains the same.

The degeneracy in the above example is caused by the supply at node 1 equaling the total demand of nodes 3 and 4. At first the only consequence of degeneracy seems to be the appearance of infeasible solutions during the course of the solution algorithm. This does not appear to present a problem since the algorithm can be programmed to disregard any infeasible solutions and the optimal solution can still be found. Upon further consideration, however, it becomes apparent that the existence of degeneracy reduces the number of trial basic feasible solutions found in each iteration. If, in the example above, the demand at node 4 was 6.0 + ε where ε is a small insignificant value, the solution considered again becomes basic. When a new solution is formed $Q_{17}$ equals ε, therefore θ is assigned the value of ε,

$Q_{17}$ leaves the basic solution and $Q_K$ enters with the value of $\varepsilon$. By using this method, the solution formed is basic and feasible, therefore eliminating degeneracy and introducing extra vertices. It appears logical that by increasing the number of vertices on the convex solution set, the solution mechanism has an increased chance of finding a path to the optimal solution, although execution may be more expensive through an increase in the number of vertices evaluated.

Degeneracy can also occur through the inclusion of one or more null nodes in the network. A further example is useful in demonstrating this situation. Figure 5.4 (a) illustrates a hypothetical distribution network with 6 nodes and 7 links; 2 of the 6 nodes are supply nodes, 3 demand nodes and 1 null node. If the initial flows specified correspond to the directed links shown in figure 5.4 (b), the algorithm will not find an alternate solution whereby node 5 is supplied by processing nodes 1 or 2 since the algorithm only introduces one solution variable into a basic solution at any time. The introduction of $Q_1$, $Q_5$ or $Q_{12}$ into the solution has no effect on the objective function since the flowrates are zero, therefore the algorithm does not attempt to bring them in. Two variables need to be introduced to supply node 5 through node 1 or 2. This problem is easily alleviated by assigning a small demand stipulation, $\varepsilon$, to the null node. With the new demand value, the solution mechanism introduces either $Q_1$, $Q_5$ or $Q_{12}$ during one iteration and then $Q_9$ during the next.

A simple perturbation technique can be used to avoid degeneracy regardless of the cause. In the case of null nodes, a stipulation value of ε should be used to denote a small demand in the case of distribution networks and a small supply in the case of collection networks. In a distribution network, a small perturbation could be used to unbalance balanced supply and demand values. The perturbation value in either case should be small enough so that it does not affect the solution costs appreciably. The perturbation technique is built into the program package in subroutine DEFINE to replace zero valued stipulations with a program defined value, EPS.

The theory behind the θ Rule assumes that degeneracy is not encountered in the problem being solved. The perturbation technique discussed above should avoid degeneracy and ensure the validity of the search technique. The perturbation technique is needed only to avoid degeneracy due to null nodes since, as discussed earlier, they can cause difficulties in the search technique, whereas the only consequence of degeneracy due to a balanced network is that it decreases the number of trial solutions in each iteration. Testing of the solution technique has shown that if the algorithm is coded to avoid infeasible solutions, convergence to an optimum is still possible if degeneracy is encountered.

The SIMPLEX algorithm, which is used in the ILP algorithm, is similar to the search algorithm discussed here in that it uses the θ Rule in its computations. Although it is possible for degeneracy to cause cycling within SIMPLEX, it has generally been found to be quite stable in practical applications (9). Cycling has not been encountered in application to network problems. An example is provided in Appendix G which is solved using the SIMPLEX algorithm. Degeneracy is encountered during application but convergence to the optimum is still attained. The example further illustrates that null nodes do not cause a problem in the search pattern of SIMPLEX as in the vertex-to-vertex search technique presented in this chapter. The SIMPLEX method therefore does not require a perturbation technique to avoid degeneracy.

## 5.3 Global and Local Optima

By definition, if the solution, $\bar{Q}^*$, produces the minimal value $Z(\bar{Q}^*)$, where $Z(Q)$ is the objective function, in the entire feasible region R (R is closed) then it is the global optimal or minimal policy. A feasible neighborhood is known as a subset of R bordered by the solutions at a radius $\delta$ around the solution $\bar{Q}^*$. Effectively, $\delta$ is the radius of a multidimensional, spherical, open region centered at $\bar{Q}^*$. Let the feasible neighborhood be denoted by R'. If $Z(\bar{Q}^*) < Z(\bar{Q})$ for $\bar{Q} \in R'$, then $Z(\bar{Q}^*)$ is a local minimum and $\bar{Q}^*$ is a locally minimizing policy. Therefore, by strict definition, the minimum found by the search technique presented in this chapter is a local minimum since in the final iteration the algorithm only checks the values of the vertices immediately surrounding the optimal policy $\bar{Q}^*$ to ensure that the solution is optimal within that neighborhood. This technique is sufficient to ensure that the policy found is also a global optimum within the feasible region R if $Z(\bar{Q})$ is unimodal in that region.

Testing of the algorithm shows that, depending upon the objective function used, $\mathfrak{Z}(\bar{Q})$ may not be unimodal in some cases and local optima can exist. A three-dimensional distribution network can be,used to demonstrate the existence of local optima. Figure 5.5 (a) illustrates the 4 node, 3 variable network used for the example. An arbitrary concave cost function, which exhibits the required properties, is chosen. The general problem statement is:

$$\min_{Q^*} \mathfrak{Z} = 10.0 \cdot Q_1 - 0.1 \cdot Q_1^2 + 14.5 \cdot Q_2 - 0.5 \cdot Q_2^2 \qquad (5.16)$$

$$+ 11.0 \cdot Q_3 - 0.2 \cdot Q_3^2$$

$$\text{subject to:} \qquad \begin{aligned} Q_1 &\le 10.0 \\ Q_2 &\le 10.0 \\ Q_3 &\le 5.0 \\ Q_1 + Q_2 + Q_3 &= 15.0 \end{aligned}$$

Slack or artificial slack variables are not required in this illustrative example. The solution regime is shown graphically as the shaded region in figure 5.5 (b). The vertices where the optimal solution could lie are numbered 1 to 4. The objective function values for the corresponding solutions are:

| Vertex | Functional Value |
|---|---|
| 1 | 142.5 |
| 2 | 150.0 |
| 3 | 140.0 Optimum |
| 4 | 145.0 |

By inspection, the optimal policy lies at vertex 3. If, however, the solution represented by vertex 1 is specified as an initial solution for the nonlinear algorithm presented in this chapter, the vertices surrounding it are checked for a less costly solution, namely, vertex number 2 and 4. Since the costs at vertex 2 and 3 are higher than at 1, vertex 1 is assumed to be the optimal solution. This problem therefore has a global minimum at node 3 and a local minimum at node 1.

Depending upon the curvature of the objective function and the implied curvature as formed by the vertices of the convex solution set, local optimal solutions can exist in larger, more practical network problems. The vertex-to-vertex search algorithm must be extended to try and ensure that the optimal solution found is a global one. There are certain random search techniques available, similar to "shotgun search" algorithms, which could possibly be used in conjunction with this algorithm to search for a less costly solution after an optimum is found. However, these methods tend to be quite expensive to use and location of a global optimum is not guaranteed. Consequently, another method is chosen which uses the vertex-to-vertex search technique. The method begins after the optimal solution is found by continuing the vertex-to-vertex search along a path of "lowest cost" vertices, until either a vertex with a lower cost than the optimum is found or the number of vertices visited meets a specified limit. The criterion used to choose from among the trial solutions generated in each iteration for the initial solution for the next iteration is similar to the first search technique (i.e. the lowest cost solution) except that solutions used previously as an initial solution in this part of the solution are not used again. The additional restriction ensures that cycling does not occur in the search.

The search for a less costly solution is analogous to exploring the lowest valleys in a range of mountains in order to seek out a lower valley or possible "improved" position. If the algorithm finds a solution, less costly than the optimum, the original search technique is again used to converge to a new optimal solution using the less costly solution as a starting point. The search for a less costly solution is then started again. When the search for a less costly solution reaches the specified limit in the number of vertices visited, the algorithm assumes that the last optimal solution found is global and terminates execution.

The limit on the number of vertices visited in the search technique is a very important factor as to whether the optimal solution found is global; a high number increases the chances of finding the optimal solution but also increases the execution time in applying the algorithm. Testing shows, however, that the required number of vertices searched to find the global optimum is normally less than or equal to the number of constraint equations. This is the value which is recommended for use in the algorithm but the variable is included in the subroutine parameter list to allow specification by the user.

## 5.4 Vertex-to-Vertex Optimization Technique in Subroutine Form

The optimization technique described in the preceding sections is coded in a series of routines for use in the econometric model package. The main computational routine, HYVRST, contains the vertex-to-vertex search technique. The name is derived from the method utilized, Hyperspace Vertex Searching Technique. Other subroutines required are:

| | |
|---|---|
| SOLVE: | to define parameters used in HYVRST |
| INTFES: | to establish an initial feasible solution for a starting point in HYVRST |
| SOLCST: | to evaluate the objective function for any specified solution |

The purposes and computational procedures of each are described in more detail in the following sections. Listings and flowcharts for the routines are provided in Appendices H and I respectively.

### 5.4.1 Subroutine SOLVE

This routine is used to interface the HYVRST algorithm with the econometric package. The subroutine statement is set up in a form which can be accessed by the routines in the package. Parameters are defined, which are required in HYVRST but are not supplied by the model. These parameters include the number of variables used in the search technique (i.e. set equal to the total number of variables minus the number of slack variables), the cycle limits on the iterative procedures in the search technique and the flag variable controlling intermediate printout during execution. The routine then calls subroutine HYVRST.

### 5.4.2 Subroutine HYVRST

Subroutine HYVRST finds the optimal solution using the searching technique described in the previous sections. The initial basic feasible solution used for a starting point is obtained through a call of subroutine INTFES. The cost for this and any other solution is determined through a call of subroutine SOLCST. The search for an optimal solution is first carried out. When found, the search is continued for a solution with a cost lower than that of the optimal solution. This search is continued for a number of iterations specified by the user. Both search methods are performed as described in previous sections. During each iteration in the search for a less costly solution, the trial solution with the lowest cost is used as the new initial solution for the next iteration. To ensure that the same solution is not used more than once for an initial solution in any iteration, a two dimensional array is used to store the numbers of the flow variables in each initial solution used. Any given set of variables in a basic solution yields a solution which is unique. The two dimensional array of solution variables is used as a base for comparison before a trial solution is assigned as an initial solution for the following iteration. If used previously, the trial solutions are checked in order of increasing costs until one which was not used previously is located. The first unused solution found is used for an initial solution. If a solution with a lower cost than the optimum is found in the search for a less costly solution, execution passes to the first search technique and the less costly solution is used as an initial solution. If a less costly solution is not found before the specified limit is met, the optimal solution found is assumed to be global and execution returns with the optimal solution stored.

During each iteration of the search techniques, checks are performed on the equations or solutions. The equations are checked to ensure that they are not linearly dependent. This could be caused by an input error and may result in failure of the algorithm unless detected. Prior to calculating a trial solution in the search techniques, checks are made to ensure that the solution will not be a looping or infeasible one, if so, the solution is not formed.

The number of iterations carried out in the search techniques have limits to maintain control of the execution time of the algorithm and to prevent cycling. The limits are user defined (in SOLVE) and are:

1) MITER1 = The maximum number of iterations to be carried out to find the optimal solution.

2) MITER2 = The maximum number of iterations used in the search for a solution less costly than the optimum (should be less than or equal to N).

3) MITER3 = The maximum number of times that the search can be carried out for a solution less costly than the optimum.

If MITER1 or MITER3 are reached, the routine prints out a message warning the user, sets an error flag and diverts execution out of HYVRST.

The user has the option of obtaining intermediate printout during execution by setting the appropriate flag in SOLVE. If requested, HYVRST prints out the initial solution and corresponding cost for each iteration. Upon finding the optimum and searching for a less costly solution, the routine prints a message to that effect. When a less costly solution is not found and the optimum is assumed global, a final message is printed telling the user this before execution returns.

### 5.4.3 Subroutine INTFES

The purpose of subroutine INTFES is to set up the structural matrix in its reduced matrix form to correspond to the initial solution specified by subroutine DEFINE or QINIT. INTFES is designed to work in conjunction with HYVRST. Initially, the routine checks that the stipulations are nonnegative, if not, it reverses the signs of the negative stipulations as well as the nonzero elements of the corresponding rows of the structural matrix. This is required for the HYVRST routine, to produce feasible solutions.

The INTFES routine checks as to whether or not an initial solution is specified by searching the flow array. If an initial solution is not specified (i.e. no nonzero elements in the flow array), an initial solution is assigned by equating the slack and artificial slack variables to their respective stipulations. When this is done, the structural matrix is already in its reduced form and execution reverts to subroutine HYVRST.

If an initial solution is specified, execution moves to another portion of the subroutine. The specified solution is checked to ensure that it is feasible, continuous and basic. If these requirements are not met, an appropriate diagnostic is printed out, an error flag set and execution passes out of INTFES.

If the specified solution is basic, feasible and continuous, the routine reduces the structural matrix to correspond to the initial solution using an iterative technique. The algorithm proceeds by comparing each of the nonzero variables with the stipulations, in turn, until it finds two which are equal (say $Q_K$ and $S_L$). The structural matrix element $a_{LK}$ is then known as the pivotal element. If the pivotal element is equal to 1, then the remaining nonzero elements in the same column are reduced to zero by performing row operations on the respective rows of the structural and stipulation matrices. On the other hand, if the pivotal element is not 1, the search for equal flow variable and stipulation values continues, beginning with the next nonzero variable until a 1 valued pivotal element is found. The first iteration ends when a 1 valued pivotal element is found and its column reduced.

Subsequent iterations proceed in the same manner as the first. In subsequent iterations, however, the variable corresponding to a column which has been reduced in a previous iteration is not used again for comparison with the stipulations since it is already contained in the reduced matrix. An exception to this rule lies in slack variables, they can be used in any iteration for a comparison. When all of the variables have been worked into the matrix, the reduced form is complete. If, however, during any iteration, equal variable/stipulation pairs cannot be found with a corresponding pivotal value of 1, the reduced matrix cannot be formed using this method and the routine ends, printing a diagnostic and setting an error flag.

The development of this technique is intended to allow the user to specify an initial basic solution which is considered to be close to the optimal policy so that a saving in computer execution time is made. Testing shows that the method works for most basic solutions specified, especially branching networks. Basic solutions produce a branching network system due to the low number of variables in the solution.

### 5.4.4 Subroutine SOLCST

Subroutine SOLCST is used to determine the cost for a specified solution using subroutines LNKCST and CENCST as described earlier in Chapter 3. This routine is required by HYVRST to evaluate the objective function for each trial basic solution generated.

## 5.5 Use of HYVRST

The array storage required in using the HYVRST optimization algorithm is given by:

$$\text{Array storage} = ((2NQ + N + 7) \times N) + NQ$$

Therefore, the storage required for different problem sizes as defined in Chapter 3 is:

| N | 10 | 20 | 40 | 80 | 100 |
|---|---|---|---|---|---|
| Array Storage | 1010 | 3820 | 14840 | 58480 | 91100 |

When compared to the storage requirements of the iterative linear programming technique and Rosen algorithm given in Chapter 3, HYVRST requires approximately twice that of each. This implies that the HYVRST algorithm could possibly cause storage problems in larger network problems, depending upon the capacity of the computing facilities.

A model run using the HYVRST algorithm and a network problem similar to that used in Appendix F is presented in Appendix J. When including the HYVRST algorithm in the model, changes must be made in the dimension statements of the driving program and subroutine NETSOL to include the extra storage required by HYVRST. Changes such as this are most likely required whenever different optimization routines are implemented.

## 5.6 Conclusions

This chapter covered the presentation of a new solution algorithm, HYVRST, which uses a vertex-to-vertex search strategy. HYVRST is accurate, stable, efficient and converges rapidly to a minimum. In addition, further explorations are automatically carried out using the same basic strategy in an attempt to locate an improved minimum. In this way, the final solution obtained has a very high probability of being a global minimum. This two stage search strategy is, as far as is known, unique in the solution of linearly constrained concave programming problems.

The penalty for the improved confidence in the solution is in the storage required and execution time. However, the initial starting condition may be set by the user so that subsequent runs may be initialized where the previous run terminated if a time limit was met.

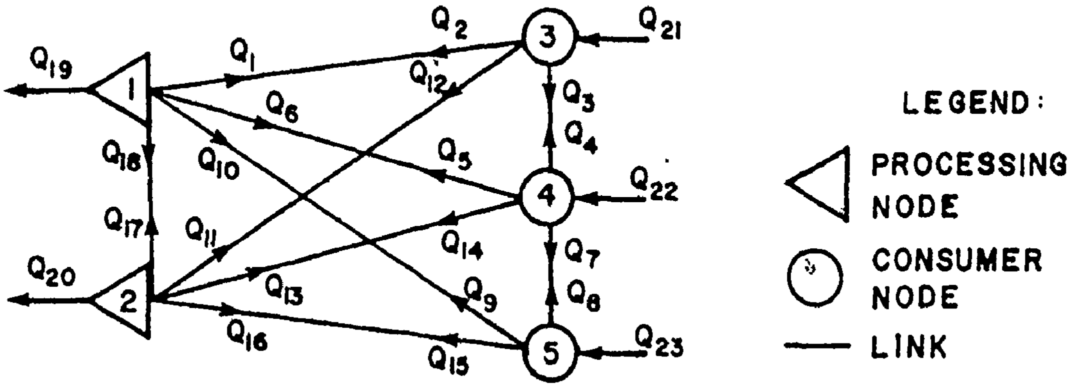


FIGURE 5·1 — 5 NODE DISTRIBUTION NETWORK FOR SAMPLE PROBLEM

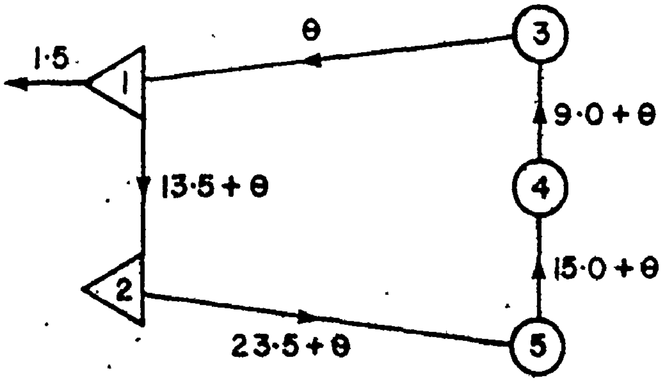


FIGURE 5·2 — FEASIBLE SOLUTION WITH A CLOSED LOOP

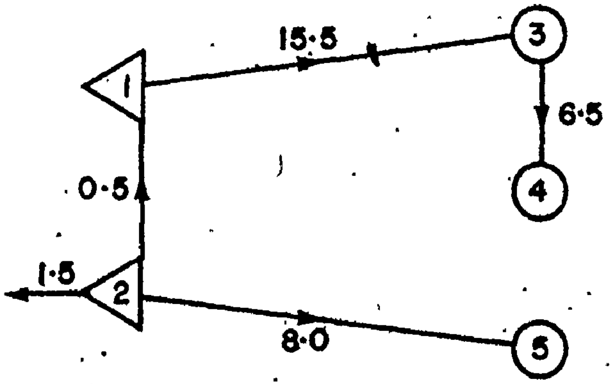


FIGURE 5·3 — OPTIMAL SOLUTION FOUND BY THE HYVRST ALGORITHM

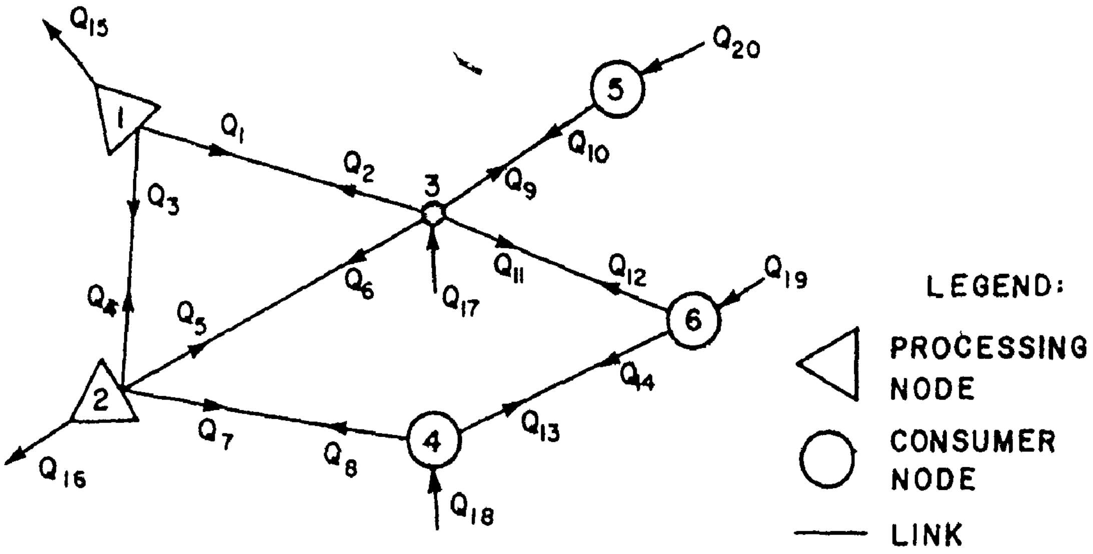


FIGURE 5·4(a) — 6 NODE DISTRIBUTION NETWORK WITH A NULL NODE

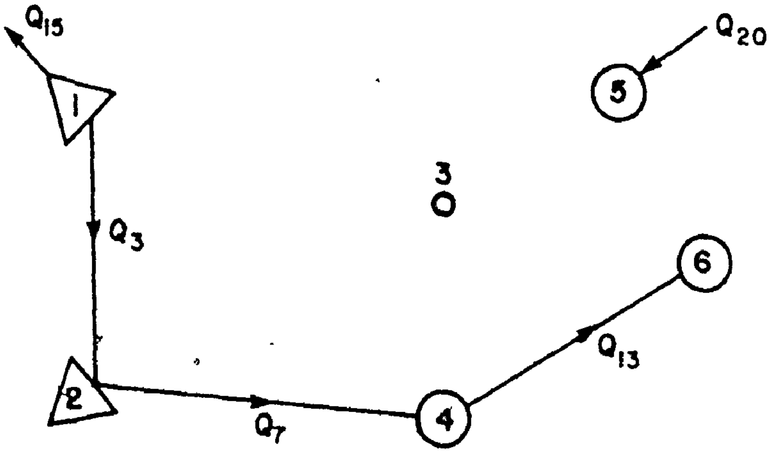


FIGURE 5·4(b) — INITIAL SOLUTION WHICH RESULTS IN THE CONCEALMENT OF THE OPTIMAL SOLUTION

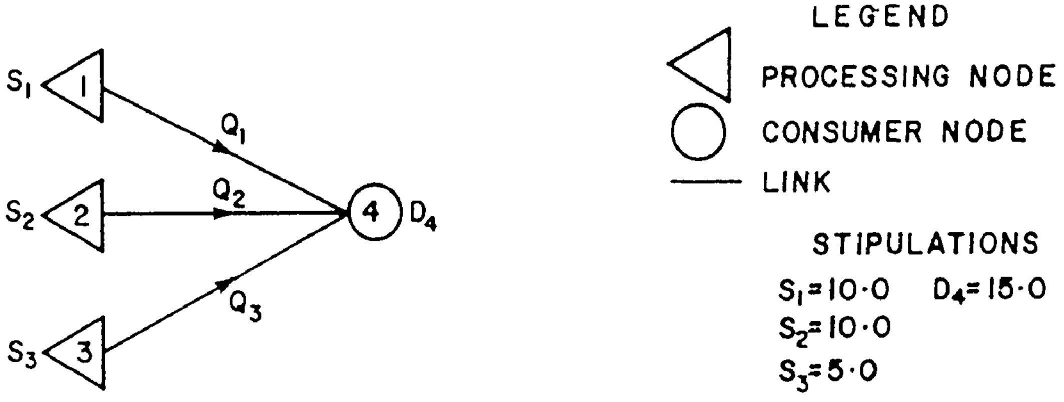


FIGURE 5·5(a) – SAMPLE 4 NODE DISTRIBUTION NEWORK

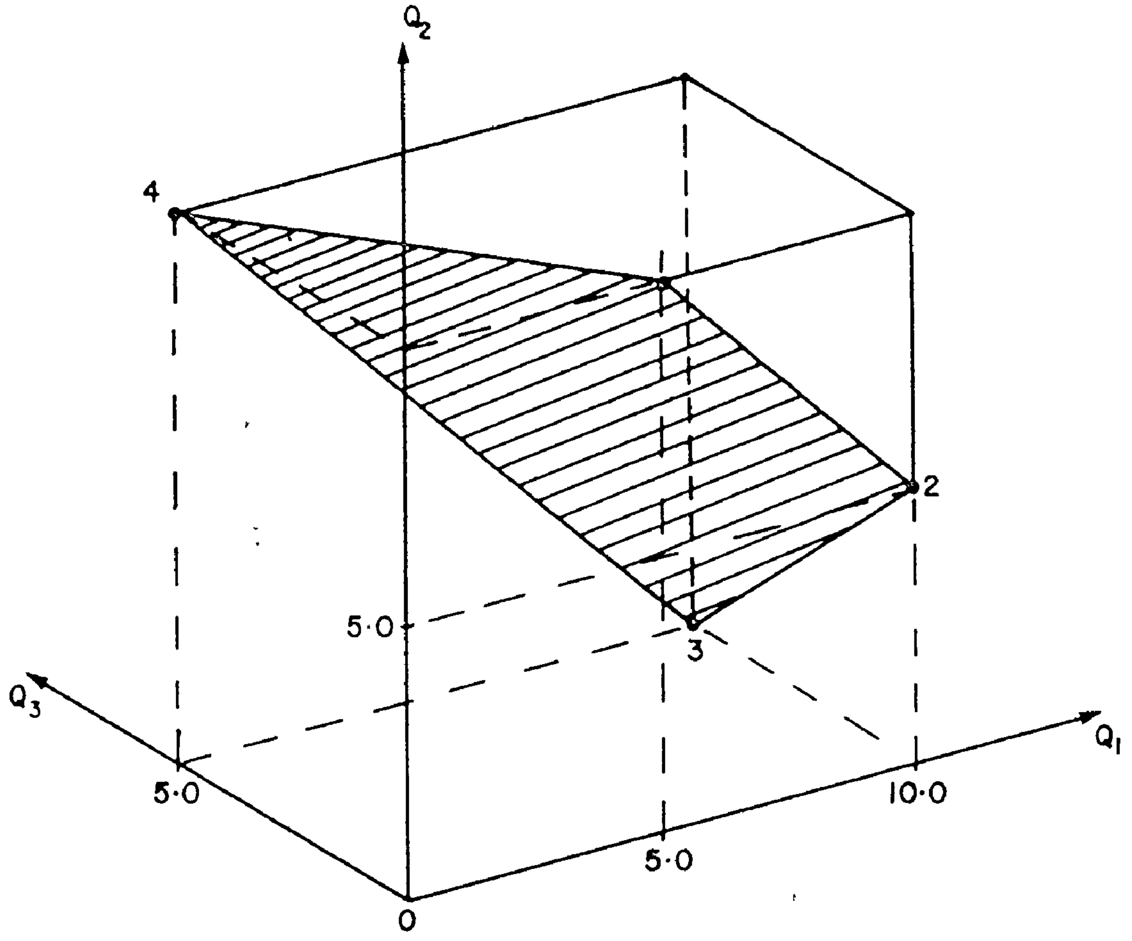


FIGURE 5·5(b) – SOLUTION SPACE FOR NETWORK OF FIGURE 5·5(a)

## TABLE 5·1 – DATA FOR 5 NODE DISTRIBUTION NETWORK

| UPSTREAM NODE | DOWNSTREAM NODE | UPSTREAM ELEVATION (feet) | DOWNSTREAM ELEVATION (feet) | LINK LENGTH (miles) |
|---|---|---|---|---|
| 1 | 2 | 150 | 250 | 4.0 |
| 1 | 3 | 150 | 175 | 7.0 |
| 1 | 4 | 150 | 200 | 10.0 |
| 1 | 5 | 150 | 225 | 9.5 |
| 2 | 3 | 250 | 175 | 9.0 |
| 2 | 4 | 250 | 200 | 10.5 |
| 2 | 5 | 250 | 225 | 7.5 |
| 3 | 4 | 175 | 200 | 4.0 |
| 4 | 5 | 200 | 225 | 5.0 |

## TABLE 5·2 – STRUCTURAL MATRIX DEPICTING THE CONSTRAINTS FOR A 5 NODE DISTRIBUTION NETWORK

|  | FLOW VARIABLE NUMBER |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | SLACK AND ARTIFICIAL SLACK NUMBER |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | STIPULATION |
| 1 | 1 | -1 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 1 | 0 | 0 | 0 | 0 | 15 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | -1 | -1 | 1 | 1 | -1 | 0 | 1 | 0 | 0 | 0 | 10 |
| 3 | 1 | -1 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 9 |
| 4 | 0 | 0 | 1 | -1 | -1 | 1 | -1 | 1 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 6.5 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 8 |

# TABLE 5·3 – REDUCED STRUCTURAL MATRIX DEPICTING A SPECIFIED INITIAL SOLUTION

| | FLOW VARIABLE NUMBER | | | | | | | | | | | | | | | | | | SLACK AND ARTIFICIAL SLACK NUMBER | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | STIPULATION |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ① | 1 | -1 | -1 | -1 | 1.5 |
| 2 | 1 | -1 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | ① | 0 | -1 | 1 | 1 | 1 | 13.5 |
| 3 | 1 | -1 | -1 | ① | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 9.0 |
| 4 | 1 | -1 | 0 | 0 | -1 | 1 | -1 | ① | 0 | 0 | 1 | -1 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 15.5 |
| 5 | 1 | -1 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 1 | 1 | -1 | 1 | -1 | -1 | ① | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 23.5 |

# CHAPTER 6

# APPLICATIONS OF THE NETSOL PACKAGE

## 6.1 Introduction

This chapter describes the application of the NETSOL package to two network problems, a water supply network (distribution) and a solid waste network (collection). In both examples the network is set up by choosing all feasible processing nodes, nonprocessing nodes and transportation links. The computer model of the network is then set up by constructing the cost functions necessary to define total system costs and making any required additions to subroutines CNSTIN and REPORT to augment the standard input and output provided by the package. The HYVRST optimization routine is used in one model and the iterative linear programming (ILP) routine is used in the other. The ILP algorithm is used in the water supply network model since this network is larger than that for solid waste (41 nodes, 54 links as compared to 14 nodes, 26 links) and some savings are realized in execution times. The models developed are used to find the optimal network for each problem and the interactive commands in the package are utilized in investigating the network systems further to answer typical questions which arise.

## 6.2 Water Supply Network

The water supply network analysed in this case is the system proposed for the new capital city of Dodoma, Tanzania. The example is one of a number of cases which were studied for this water distribution network (35). The objective of the study is to locate the most economical conveyance network which can be used to connect the two supply points to the various districts for a design life which spans to the year 2000.

### 6.2.1 Network Definition

A schematic diagram of the network with its supply points, consumer nodes and feasible conveyance links is given in figure 6.1. The system analysed in this example is a reduced version of the network in the original study (35). The total number of nodes is reduced from 64 to 41 by lumping the nodal points for some outlying areas to single nodes and maintaining the same demand stipulation for each area. Some balancing reservoirs were also deleted using the same lumping approach. The network size was reduced to 41 nodes to "fit" the model into the interactive computer facilities available at the time (the full-sized system was analysed using the ILP algorithm on a CDC 7600 computer). Of the 7 supply points shown in figure 6.1, numbers 1 and 2 are the two main supply points with the remainder being balancing reservoirs. The analysis carried out in this example is based upon the maximum day demands, in which case, the balancing reservoirs are being recharged and thus have a demand associated with them. The supply and demand stipulations are shown in the data file provided in table 6.1. The supply point yields are positive and demand values are negative to conform to the sign convention used in the program.

Each demand node is characterised by a number and descriptive name which relates to the area being served by that node. The boundaries of the area served by each node are dictated by either a physical constraint or pressure zone boundary. The state, or pressure elevation, at each node is defined by the location of a primary or secondary (storage tank) source for supply points or reservoir nodes, or, by topographical levels with a minimum service pressure of 30 psi (21 meters) for consumer nodes. The maximum allowable service pressure in the system is 85 psi (60 meters) which is not used at this point but is used later when discussing the consolidation of pumps. Table 6.1 shows the state values assigned in meters $H_2O$.

As shown in the schematic of figure 6.1, the nodal points are interconnected by 54 feasible links. The link lengths, in meters, and connectivity are given in the input data file of table 6.1. The nodal order does not imply any preconceived flow direction. In this case, each link is the same length in both directions and, thus, only one link length is supplied for each link in the data file, as specified in the data file requirements described in Appendix E. Similarly, no initial flowrates are specified and the corresponding fields are left empty in the last record type in the data file.

### 6.2.2 Cost Algorithms

In this analysis, processing costs are not evaluated since both supply points are to be used in the final system and thus no choice has to be made on the basis of economic considerations. The balancing reservoirs also represent a fixed charge on the system since they are dependent on the water consumption in the area and not upon the supply route selected. The system costs therefore depend only upon the pipeline and pumping costs within the links themselves.

Approximate pipeline costs for the Dodoma area can be represented by equations 6.1 and 6.2 given below. Two functions are required, one for diameters less than or equal to 0.5663 meters and one for larger diameters.

$$\begin{aligned} C = {} & 1458.0 \,.\, DIA - 3310.6 \,.\, DIA^2 \\ & + 10450.8 \,.\, DIA^3 + 1435.9 \,.\, DIA^4 \qquad 0 < DIA \leq 0.5663 \\ & - 33661.4 \,.\, DIA^5 + 41885.4 \,.\, DIA^6 \end{aligned} \tag{6.1}$$

$$C = 252.0 \times 10^{1.3317 \,.\, DIA} - 200.0 \qquad 0.5663 < DIA \tag{6.2}$$

where: C = Installed pipeline cost (shs/meter)

DIA = Pipeline diameter (meters)

The capital costs for pumping stations within a pipeline are given by the following logarithmic relation:

$$C = 10^{(3.158 - 0.318 \log_{10} (DHP))} \qquad (6.3)$$

where: C = Capital pump station (shs/HP)

DHP = Design pump horsepower

The installed horsepower is increased over the design value by an amount depending upon the magnitude of the horsepower. This factor is variable between the limit of 2.0 (i.e. duplicate standby power) for very small stations, and 1.4 for very large stations comprising several pump sets. The following exponential relation is used.

$$IHP = (1.4 + 0.6\ e^{(-DHP/200.0)})\ DHP \qquad (6.4)$$

where: IHP = Installed horsepower

DHP = Design horsepower

The power costs are taken to be set at a constant rate of 850 shs/HP/ annum.

Costs are also included to allow for the replacement of pumps and other related machinery. It is assumed that 30% of the capital expenditure represents equipment with a design life of only 15 years. Replacement is provided for by a sinking fund annual cost based on 4.5% inflation and the standard discounting interest rate for the model.

The cost functions are developed for the model using the costs given above for each of the components. All costs are converted to a present worth value for comparison. In doing this, an interest rate of 8.0% and inflation rate of 4.5% are used. The economic life of the system components is taken as 15 years for pumps and 30 years for any pipes and buildings. For pipe size computations, the Colebrook-White formula is used with a roughness height of 0.00025 meters.

As discussed earlier (section 3.3, page 81), in water supply systems for a given pipeline, it may be more economical to use a smaller pipe size and introduce a pump in the link to maintain the head requirements. The question then arises as to what is the most economical trade-off between pump and pipe sizes. A typical plot of total costs versus pipe friction losses for a pump-pipe setup is given in figure 3.4. To find the optimal configuration, a sub-optimization algorithm is included in the cost routines which uses a Fibonacci type search method (24) to select the optimum head loss. The cost function to evaluate the present value cost of the pipe-pump link as a function of head loss comprises the following steps:

1) Select an arbitrary headloss, $h_f$, for the flow variable $Q_{ij}$.

2) Design the pipe diameter DIA = f($h_f$, $Q_{ij}$, Pipe length, Resistance law).

3) Calculate pump life requirement.

   $h_p = h_f$ + (state at D/S node) - (state at U/S node).

4) If $h_p \leq 0$ set $C_1 = C_2 = 0$ and go to step 8; otherwise, calculate the required pump power $DHP = \frac{(\gamma\ Q_{ij}\ h_p)}{n}$ where $\gamma$ = specific weight of water.

5) Increase required pump power to the installed value IHP.

6) Compute capital cost for pump station $C_1$.

7) Compute operating, maintenance and sinking fund costs for pump and discount to present value $C_2$.

8) Compute capital cost for pipeline $C_3$.

9) Calculate maintenance cost for pipeline and discount to present value $C_4$.

10) Obtain total present value cost $C = C_1 + C_2 + C_3 + C_4$ .

The head loss is adjusted in accordance with an initial course search, followed by a Fibonacci search and cost evaluated by steps 1 to 10 for each iteration. Convergence to a minimum cost and optimal headloss $h_f$* is assumed when the incremental change in $h_f$ reduces to a value less than a specified tolerance. Upon determination of the optimal headloss this, plus the corresponding values of pipe diameter, pump horsepower, capital and operating costs are stored in ad hoc common block arrays for subsequent transfer back to subroutine REPORT.

It may not be economical to have a pump-pipe configuration in each conveyance link but the sub-optimization algorithm is applied to each link in case it is more economical to include a pump. In order to ensure that the cost curves are concave with respect to flow, a continuous variable such as $h_f$ must be used. Using a discrete pump or pipe size in cost calculations can lead to instabilities and must be avoided. It is a simple matter to adjust the final solution slightly to suit commercially available sizes.

Since a pump-pipe combination can be assigned to any link, the accuracy of the optimal solution depends heavily upon the reliability of the pipe and pump cost functions. If both functions are accurate, the optimal solution determined should yield the proper distribution between pump lift and gravitational head, and an accurate estimate of capital and operating costs. If both cost functions are in error but are nonetheless compatible, then the system costs will be incorrect but the optimal policy defining pump and pipe sizes should be correct. If only one function is in error, then both the system costs and reliability of the solution (i.e. chosen policy of conveying the water) will be in error.

Testing of the cost functions shows that the costs for pipelines are concave with respect to flow for pipe diameters less than approximately 0.8 meters and convex above that. This is not surprising since the accuracy of the pipe cost function for pipe sizes above 0.6 meters (equation 6.2) is suspect but unfortunately no further information is available. It should be stressed that cost curves with a transition point as described here are not typical for pipeline costs and are generally concave throughout their feasible range. Fortunately, in this study, the results show that most pipe sizes chosen in the final solution are less than 0.8 meters in diameter and the ones larger than this size are the mains connecting the supplv to the system. Since both of the available supply points must be connected to the system, and there are only two available connecting links, the size of the mains are dictated by established flows rather than economics and the shape of the cost function does not have a bearing on the sizing of these mains.

### 6.2.3 Results

The first analysis is carried out to determine the optimal system for the defined network. The summary printouts for this analysis are presented in figure 6.2. This represents the optimal solution for the system as defined by the data file in table 6.1. The solution is obtained after 3 iterations of the ILP algorithm and 66 CPU seconds of execution time on a CDC 6400. Figure 6.2 contains the printout provided by NETSOL which is augmented to output further design information. The table presents design and cost details for each nonzero flow variable. For each flow, the appropriate conveyance link is defined by the number and name of the upstream and downstream nodes. The flow in the pipe, required diameter, overall headloss for the link, and pump horsepower (if any) make up the design information. The costs are broken up into capital costs, operating or maintenance costs and total costs. All costs are in millions of Tanzanian shillings.

The results show that for the specified pressure states, pumps are required at 11 different nodes in the optimal network. It may be more economical, however, to have a fewer number of pumps with a larger head capacity and increased pressures at certain nodes due to the economies of scale involved with increased size of pump installations. This is a property particular to water distribution networks since the state at each node can vary between a minimum and maximum allowable pressure. The problem is therefore strictly a non-linear problem in terms of both flow and nodal pressure. The solution to such a problem by standard methods such as hill climbing or pattern search would most likely be prohibitively expensive for problems of other than trivial size. The NETSOL package however gives the user the option of changing the nodal states specified which can be useful to consolidate pumping locations.

A second analysis is performed to investigate the consolidation of pumps. The following nodal points are chosen for likely pump locations using, as a guide, the results of the first analysis.

| | | |
|---|---|---|
| Node 1 | - | Bubu River (source) |
| Node 2 | - | Makutapora Wells (source) |
| Node 10 | - | Pumps (PD 2) |
| Node 13 | - | Pumps (PD 3 - West) |
| Node 24 | - | Junction G-2/Southeast |
| Node 36 | - | Junction I-3/West |
| Node 38 | - | District D-3/West |

Pumping stations are not necessary at each selected location but the above nodal points were chosen as the most desirable locations for pumping facilities in the original study.

As discussed previously, each link is designed on the basis of minimizing the cost using the relationship between pump and pipe costs and the head loss along the link. To overcome the need for a pump in a given link, the optimal head loss determined through sub-optimization in the first analysis is added to the state of the downstream node, yielding the new upstream state. The nodal state is increased at each node further upstream along connecting links until a connecting link is reached where a pump is allowed. This procedure is performed along each series of connecting links where pumps are to be omitted. Although not used in this example, the original study utilized an algorithm which facilitated the consolidation of pumps to specified nodes allowing the user to easily obtain a consolidation run directly following the initial optimization. In the example, the STATE command in subroutine MODIFY is utilized for the consolidation of pumps.

The method of consolidating pumps described above assumes that the optimal strategy does not change after changing the nodal states. This of course depends upon many factors such as the network configuration, cost functions and number of state changes made. It is recommended that a subsequent optimization attempt be carried out after any state changes which gives the user some insight as to the sensitivity of the model. If it is anticipated that solution changes will be negligible, then solution costs for a previously obtained strategy and new states can easily be obtained through the use of the "SOLVE" command in the program.

The changes made to the nodal states for the second analysis are summarized in table 6.2. The old state values shown in the table represent a service pressure of 30 psi (21 meters) at the corresponding nodal points. As mentioned previously, the maximum service pressure allowable is 85 psi (60 meters) which represents a working range of 39 meters. In the table, however, it can be seen that in some cases the maximum pressure is exceeded. Being a preliminary analysis, this condition is allowed but changes would have to be made in the final design to compensate for the high pressure values.

Following modification of the nodal states, a second optimization is carried out to detect any change in the optimal network selected. The results of this analysis are shown in figure 6.3. The optimal policy determined by the model is essentially the same as the one found except for the reduction in the number of pumping installations from the original 11 to 6. Due to the economies of scale realized with the larger pump installations, there is a net saving of 1.02 million shillings. However, this saving is not as significant as it first appears, since the reduction is only 1.0% when compared to the original cost obtained. It appears, therefore, that no tangible benefit is obtained by obtaining a second optimization after consolidating the pumps in this particular case.

## 6.3 Solid Waste Network

In this example, the solid waste network analysed is for a solid refuse collection system servicing the Hamilton-Wentworth Region in southern Ontario. Some of the data used in this example is extracted from an original study (31) which was undertaken to propose a regional disposal facility for the area. The objective of this example is to determine the most economical disposal facility from a number of alternatives and find the optimal conveyance network for transporting the wastes to the disposal site. A 20 year span is used as a design life period for the system. Upon finding the optimal system, modifications are made to determine the difference in cost for slightly different solutions and to quantify intangible benefits.

### 6.3.1 Network Definition

The boundaries of the region included in this study are shown in figure 6.4. The nodal points representing waste generating areas and the feasible disposal facilities are also shown in the figure. Each waste generating centre is located near the centroid or at a disposal centre of the waste generating area which it represents. If the volume of waste generated in the area is large enough, the nodal point could represent a transfer facility for transferring waste from a collection vehicle to a larger, long haul type vehicle. No attempt is made to optimize the location of the transfer facility or assign a cost of local collection. The costs generated by the model are for transporting the wastes to the disposal site and subsequent disposal costs.

Three disposal facilities are considered as being feasible in this study:

1) SWARU -

This is an existing reclamation-incineration facility which presently handles a portion of the area's wastes. In the past, the unit has been operating well below its design capacity primarily due to this being a relatively new concept for a large scale disposal facility. Incineration is an expensive type of disposal practice, but it has definite aesthetic advantages over alternate methods such as landfill. For the purposes of this study, it is assumed that SWARU can be easily expanded to handle all of the region's wastes over the design life chosen.

2) Glanbrook Landfill Site -

A sanitary landfill site in Glanbrook, being near the border of the region, has the advantage of being relatively inexpensive if transportation costs do not make up a major portion of the total costs. Although this type of disposal is more "offensive" than incineration, its impact is lessened by removal from any large residential areas.

3) Hagersville Landfill Site -

This possible disposal site is out of the region and quite distant from the waste generating areas. However, it has the advantage of inexpensive land and connection to the collection network by relatively inexpensive rail transport.

No effort has been made to establish the feasibility of each site with respect to being able to acquire the land required for the sites or use the existing rail lines for transporting the waste. This study is undertaken assuming that each component of the system proposed is feasible.

The transportation corridors chosen for the conveyance network are shown in figure 6.4. Most links utilize the road systems with the exception of a single rail line connecting the Hagersville landfill site to the network. Major highway systems are chosen where possible to decrease travel times. The links are also made as direct as possible to keep the distances low. The network is planned to provide a number of alternate routes for the waste to be transported to the optimal disposal site, allowing the model to choose the least costly conveyance system.

The network system is represented by the schematic shown in figure 6.5. Each node is assigned a number with its corresponding name shown in the program data file in table 6.3. The stipulations shown in the data file opposite each nodal description represent the anticipated annual waste production, in millions of tons, for the year 1997. These future production values are determined through the use of projected population figures (31). The disposal sites, nodes 1, 2 and 3, have no stipulation except for node 1, the SWARU site, which carries the waste production for the Hamilton proper area. The links are also defined in the data file in terms of the two end points of each link with its corresponding length in miles. Unlike the manner of conveyance in a water supply network, the flow through a link in solid waste is not continuous but involves a discrete movement of material from one point to another. Each link can comprise different classes of roads (i.e. freeway, rural, suburban or central) and an average velocity is assigned to each link. Both the average velocity and link length are required in defining the conveyance. The two terms are used to calculate travel times and fleet size which are in turn required in the calculation of conveyance costs. The input of velocity data is provided for by subroutine CNSTIN in the form of an ad hoc block data statement. The velocity values included in the block data statement are shown in brackets in table 6.3.

### 6.3.2 Cost Algorithms

In this example, costs arise from both processing (i.e. disposal of waste) and transportation. The costs comprise both capital and annual operating and maintenance costs. A variety of literature is available to facilitate the definition of costs for different components of a solid waste system.

To define cost functions for this example, cost data is obtained from references which displayed the variation in component cost with the throughput, which is then scaled to represent any available costs for the study area. Curve fitting techniques are used to facilitate the definition of readily usable cost functions.

For the SWARU installation, functions are derived which represent concave cost curves for capital and operating costs for similar type incineration facilities (22). The resulting cost function for capital costs follows.

$$C = 971950.6 + 4289.51 \,.\, QDES - 0.34625 \,.\, QDES^{2} \qquad (6.5)$$

$$+ 1.25908 \times 10^{-5} \,.\, QDES^{3} - 1.66771 \times 10^{-10} \,.\, QDES^{4}$$

where: C = Capital expenditure (dollars)

QDES = Design flowrate (tons/day)

$= \dfrac{\text{Actual flowrate (tons/day)}}{\text{Underutilization ratio}}$

The underutilization ratio (i.e. ratio of actual processing flowrate to design flowrate) of an incineration facility can vary widely from installation to installation depending upon a number of factors such as the experience of the operators, suitability of the waste to the type of machinery, etc. The cost function is therefore formulated so that the design flowrate can be altered to reflect the underutilization expected. The operating costs for incineration are given by the following relation:

$$C = 4.0 + 16.59024 \; . \; e^{(- 0.08308 \; . \; Q^{.5})} \qquad (6.6)$$

where: C = Operating costs (dollars/ton)

Q = Operating rate (tons/day)

To allow for underutilization, the operating costs are modified:

$$CM = 0.97 \; (Eff^{\; - 0.66}) \; . \; C \qquad (6.7)$$

where: CM = Modified operating cost (dollars/ton)

EFF = Underutilization ratio

C = Operating cost from (6.6)

To compute the capital costs for a sanitary landfill site, the volume of waste generated for a 20 year period of operation is calculated. Based upon a projected population growth, the volume of waste for a 20 year period is 10.46 times the yearly quantity in the 20th year. The landfill site costs are then calculated assuming a waste density of 1800 lbs/cubic yd. after baling, final waste depth of 30 feet and prepared land costs of 2200 and 2000 dollars per acre for Glanbrook and Hagersville respectively.

The operating costs for landfill (4) are computed through the following relation:

$$C = 0.8 + 5.477293 \cdot e^{(-\ 0.186507\ \cdot\ Q^{0.4})} \qquad (6.8)$$

where: $C$ = Landfill operating costs (dollars/ ton)

$Q$ = Operating rate (tons/day)

This relation is used to compute operating costs for both landfill sites.

To achieve a solid waste density of 1800 lbs/cubic yard in the landfill site, a baling facility is required. Two feasible sites are chosen for a baling station. Due to the relatively short haul distances in the network, it is assumed that one would be located in the Glanbrook site for the waste deposited there. For the waste at the Hagersville site, a baling station would be located at the loading end of the rail link joining Hagersville to the network, thus decreasing the rail shipping costs. Baling station costs are therefore lumped with processing costs for the Glanbrook site and included with the link costs of link 13-3 for Hagersville.

The capital costs of a baling station (31) are provided by the relation:

$$C = 1401330.0 + 1770.0 \cdot Q - 0.185723 \cdot Q^2 + 3.85975 \times 10^{-6} \cdot Q^3 \qquad (6.9)$$

where: $C$ = Capital cost for baling (dollars)

$Q$ = Operating rate (tons/day)

and the operating costs are given by:

$$C = 670.0 \cdot Q - 0.025 \cdot Q^2 \qquad (6.10)$$

where: $C$ = Operating costs (dollars/year)

$Q$ = Operating rate (tons/day)

The transportation cost for the rail link joining nodes 13 and 3 comprise the sum of the baling costs as computed through equations 6.9 and 6.10 and rail shipment costs. The rail costs (4) are computed assuming 1 trip per week using the relation:

$$C = 0.35 + 444.32 \cdot Q^{(-0.89485)} + 9.0657 \cdot XL \cdot Q^{(-0.977387)} + .003 \cdot XL \qquad (6.11)$$

where: $C$ = Rail shipment costs (dollars/ton)

$Q$ = Tonnage per haul (tons/week)

$XL$ = Haul distance (miles)

In calculating the truck transportation costs, the fleet tonnage and mileage covered per year are calculated assuming 2496 hours in a working year (i.e. 8 hrs/day, 6 days/week, 52 weeks/year) and using the average link velocities provided.

$$\text{MILEAGE} = 2496 \,.\, \text{VEL} \qquad (6.12)$$

$$\text{TONNAGE} = 2 \,.\, \frac{\text{QI} \,.\, \text{XL}}{\text{MILEAGE}}$$

where: MILEAGE = Vehicle miles travelled per year (miles/year)

TONNAGE = Fleet tonnage (tons)

QI = Yearly operating rate (tons/year)

XL = Link length (miles)

VEL = Link velocity (miles/hour)

The capital costs are then computed through the relation:

$$C = 5193.0 + 4066.0 \,.\, \text{LOG}_e\ (\text{TONNAGE}) \qquad (6.13)$$

where: C = Capital costs (dollars)

The operating costs are calculated through the relation:

$$C = 15724.8 + (0.28 + 0.24 \,.\, \text{LOG}_e\ (\text{TONNAGE}))\ .\ \text{MILEAGE} \qquad (6.14)$$

where: C = Operating cost (dollars/year)

In calculating the final costs for comparison, present worth values are used assuming an effective interest rate of 4.5% (e.g. 10% interest and 5.5% inflation). The total system costs are evaluated, that is, the capital costs plus operating costs over the design life (20 years). In calculating the capital costs for the processing equipment (i.e. incinerator and baling station), it is assumed that construction is carried out in stages. The construction of 80% of the equipment required to process the increased quantity of waste generated beyond 15 years is deferred until the fifteenth year. For the sanitary landfill sites, it is assumed that all of the required land is purchased in the first year.

Subroutine CNSTIN is written to facilitate user oriented changes to the effective interest rate and efficiency of the incineration equipment. Variables of this type are most likely to be subject to uncertainty and hence the ones which the user may wish to change for sensitivity tests. The standard output of subroutine REPORT is augmented to supply additional data on the system design. The total tonnage required for the trucking in each link is printed out with the transportation costs. The acreage required for the sanitary landfill sites is output before the cost summary if either sanitary landfill site is included in the solution. If the Hagersville site is included, the flowrate along the rail line is output in tons per week.

### 6.3.3 Results

A number of program runs are executed in this example to obtain, firstly, the optimal system and then determine costs for other, more aesthetically desirable solutions. An analysis is also carried out to determine the effects of the incinerator efficiency on the solution costs.

The first analysis is directed towards finding the optimal solution using the HYVRST optimization routine. The results of the analysis are shown in the summary printout in figure 6.6. The output follows the same general format as outlined in the previous example. The solution is obtained after 67 CPU seconds on an HP 3000 system.

The flowrate values shown in the summary output are in the same units as those in the data file (i.e. millions of tons per year for the final design year). The costs shown are costs for the total design life in millions of dollars.

To test the optimality of the solution, certain links specified in the optimal solution are deleted using the command options in subroutine MODIFY (i.e. links 4 to 14 and 14 to 1) which appear as though they may make the system expensive and a new optimal solution determined. The output from this analysis is shown in figure 6.7. With the above stated nodes deleted, the system costs increase by approximately one million dollars using alternate conveyance links.

In the next analysis, the optimal solution is obtained with the Glanbrook landfill site removed from the network. The output for this analysis is shown in figure 6.8. In this system, the

Hagersville landfill site is chosen resulting in a cost of some 18 million dollars over that of the system in the previous analysis. The transportation costs form the bulk of the system costs in this solution whereas in the previous case the processing costs were higher. The rail costs appear to be especially high but this cost includes the cost of baling prior to rail shipment. This analysis shows that the costs of a landfill site outside the region are prohibitively high due to the increased transportation costs.

To determine the optimal transportation network using the SWARU incineration facility, the Hagersville site is next removed from the network with the Glanbrook site still deleted and a new optimal solution is obtained. The results of this analysis are also shown in figure 6.8. The system costs using the SWARU facility is much more expensive (i.e. approximately 4 times) than using the Glanbrook landfill site. The difference in costs is partially due to the underutilization ratio used for the incineration facility (i.e. 22%). To illustrate the effect of underutilization on the system costs, further analyses are shown in figure 6.8 for values of 75% and 100% respectively. At a value of 100%, the total system cost is reduced by more than half. Using a more reasonable value of 75%, the total costs are reduced by less than half but the costs of the system are now less than twice that for the optimal system using the Glanbrook site.

The total system costs are significantly greater when using an incineration facility rather than sanitary landfill, making it obvious why sanitary landfill sites are generally utilized wherever possible. As feasible landfill sites within the region are depleted, however, sites farther away have to be used if they can be acquired, thereby increasing the transportation costs. The differences in system costs are then decreased. Another factor involved is the aesthetic advantage of an incineration facility. The difficult question to answer, however, is whether the aesthetic advantage of a central incineration facility is worth approximately 90 million dollars over a 20 year period as faced in this example.

The Glanbrook landfill site should be used as the major disposal centre for most of the design life (i.e. 20 years) if not all of it due to the savings involved over incineration. As more experience is gathered in incineration and reclamation of waste material, the cost difference between this and landfill should decrease. The difference would be further reduced by the higher cost of using landfill sites outside the region in areas such as Hagersville. At some point it may be reasonable to assume the extra cost of incineration for the aesthetic improvement and incineration and/or material reclamation facilities should be incorporated.

## 6.4 Conclusions

The examples in this chapter illustrate the application of the NETSOL package to two network problems, water distribution and solid waste collection. Both examples are taken from actual studies performed and are used to answer some of the questions which arise in such studies. The work done in each example, by no means, constitutes complete studies but is sufficient for illustrative purposes. The package can, of course, be applied to other types of distribution and collection networks but the two presented here serve to demonstrate a number of the capabilities of NETSOL. The two optimization routines adopted for use in this study are applied in the network problems. The ILP algorithm which requires relatively small computer storage is used in the larger water supply problem. The HYVRST algorithm which requires more computer storage than ILP but is also more accurate is used in the solid waste problem. Both routines work well in the sample problem that they are applied to and no difficulties are encountered in their use. The solutions converged upon by the routines appear to be global optima.

It is of interest to note that although each network problem is basically the same in theory, there are many special properties existing in each case. The differences do not affect the optimum finding strategy of NETSOL but make the modular approach to model construction very important. An example of this importance can be seen in the need to input special design related information such as interest rates, pipe resistance coefficients and average transportation link velocities. It is beneficial to have the capability of modifying some of these variables (i.e. through CNSTIN) and all of it must be transferred to the proper cost/design routines. As well as augmentation of input data, augmentation of output data is also important. The output of design related information in subroutine REPORT can be very beneficial in interpreting the results of a solution (e.g. pipe diameter and pump sizes in a water distribution example). Modularization of the model makes changing of the input or output facilities much easier.

Other facilities of the NETSOL package also prove themselves useful in the model applications. The commands available in subroutines QINIT and MODIFY are valuable in answering questions arising in these studies. The STATE command is useful to investigate the effects of pump consolidation on the solution to water distribution networks. The NODES and LINKS commands are useful in testing the optimality of the optimal solution found and also quantifying the effects of deleting marginal communities or choosing certain processing centres for intangible reasons. A major advantage in using the package is the ease in which an optimal solution or cost of a specified solution can be obtained once the model is set up.

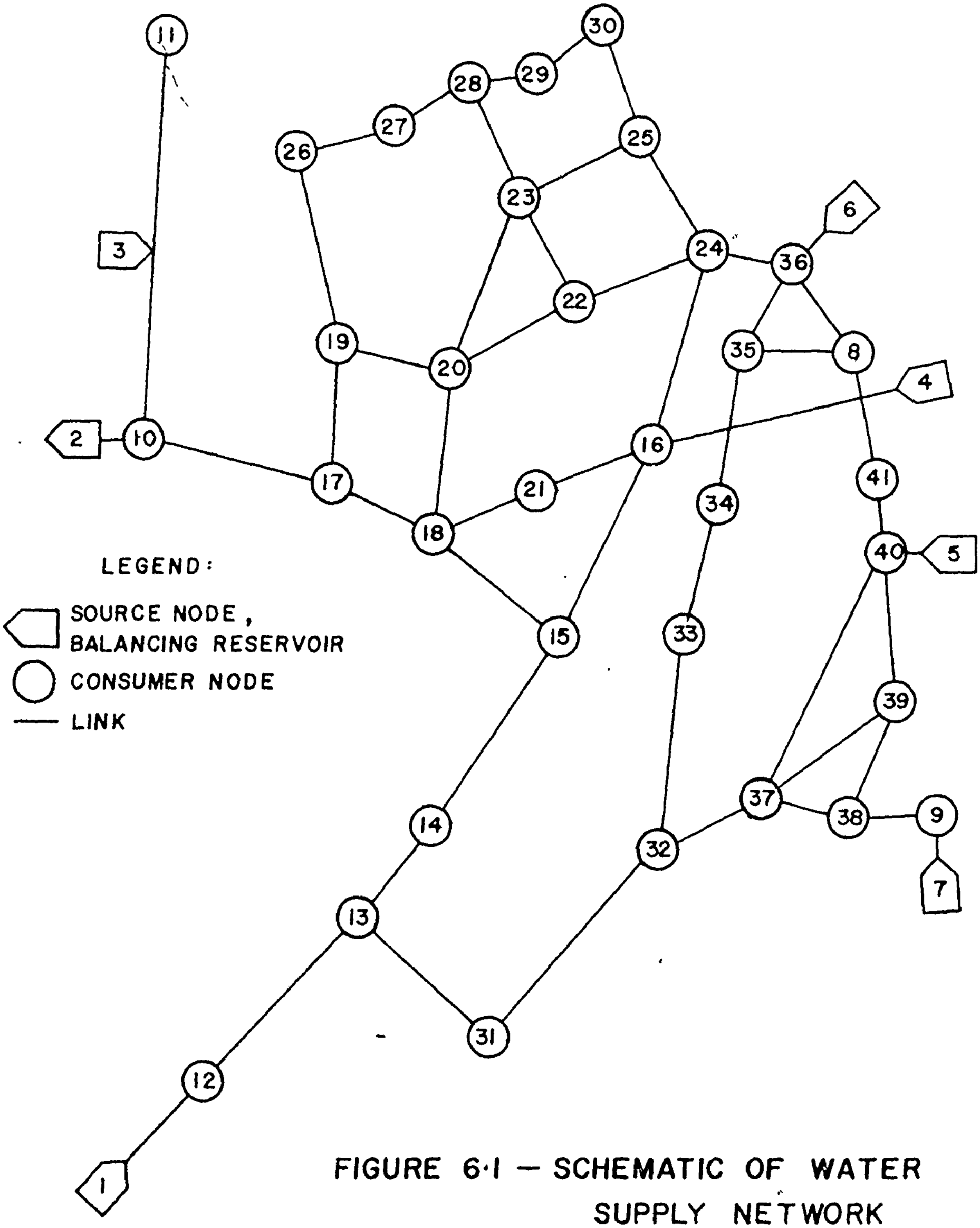


FIGURE 6·1 — SCHEMATIC OF WATER SUPPLY NETWORK

# FIGURE 6·2 – WATER SUPPLY NETWORK SUMMARY OUTPUT

```
                    PROBLEM SOLUTION
 PROCESSING CENTER COSTS
    CENTER                         FLOW          COST (MIL
                                   (CMS)          SHS-M

BUBU RIVER                         1.58            0.00
MAKUTAPORA WELLS                    .39            0.00


 TRANSPORTATION COSTS
NODE           ROUTE                  FLOW     DIA     HEAD    PUMP     COST (SHS-M)
                                      (CMS)    (M)    LOSS M    HP   TOTAL  CAPITAL  OPER.

  1  BUBU RIVER               TO
  12 DIST. B-2                      1.5792    .379    17.32   2776.  32.733 16.453  1.455
 12  DIST. B-2                TO
  13 PUMPS (PD3-WEST)               1.2418    .357    12.85    247.  11.289  9.080   .206
 13  PUMPS (PD3-WEST)         TO
  14 DIST. D-2(S.W.)                 .8441    .746     3.20      0.   1.949  1.907   .013
 14  DIST. D-2(S.W.)          TO
  15 DIST. F-2(S.W.)                 .8349    .807     4.90      0.   4.739  4.562   .046
 15  DIST. F-2(S.W.)          TO
  16 DIST. F-2(S.E.)                 .7491    .711     7.20      0.   3.831  3.577   .036
  2  MAKUTAPORA WELLS         TO
  10 PUMPS (PD2)                     .3939    .568     6.10      0.   1.962  1.362   .019
 10  PUMPS (PD2)              TO
   3 MLIMWA RES.(PD1)                .3027    .482     6.85    422.   5.371  3.250   .233
  3  MLIMWA RES.(PD1)         TO
  11 DIST. A-1                       .2770    .365    45.40      0.   1.405  1.268   .013
 10  PUMPS (PD2)              TO
  17 DIST. E-2(WEST)                 .0912    .320     6.24    182.   2.999  1.742   .103
 15  DIST. F-2(S.W.)          TO
  18 DIST. F-2(N.W.)                 .0756    .196    73.50      0.    .240   .216   .002
 17  DIST. E-2(WEST)          TO
  19 DIST. E-2(CEN)                  .0754    .159    36.90      0.    .217   .195   .002
 19  DIST. E-2(CEN)           TO
  26 JNCTN-(H-2 N)                   .0497    .256     4.30      0.    .646   .603   .006
 18  DIST. F-2(N.W.)          TO
  20 DIST. G-2(N.W.)                 .0493    .321     2.10      0.    .549   .542   .005
 20  DIST. G-2(N.W.)          TO
  19 DIST. E-2(CEN)                  .0202    .151     7.60      0.    .127   .125   .001
 28  JNCTN-(H-2 S)            TO
  27 DIST. H-2                       .0257    .164     6.10      0.    .112   .102   .001
 29  DIST. I-2                TO
  23 JNCTN-(H-2 S)                   .0257    .161     6.70      0.    .109   .099   .001
 30  JNCTN-(I-2 S)            TO
  29 DIST. I-2                       .0644    .234     5.20      0.    .138   .127   .001
 16  DIST. F-2(S.E.)          TO
  21 DIST. F-2(CEN)                  .0167    .122    22.60      0.    .153   .151   .002
 20  DIST. G-2(N.W.)          TO
  22 DIST. G-2(CEN)                  .0201    .195     2.24      0.    .312   .253   .005
 16  DIST. F-2(S.E.)          TO
  24 JNCTN-(G-2 SE)                  .6202    .451    24.30      0.    .666   .600   .006
 24  JNCTN-(G-2 SE)           TO
  25 DIST. I-2(S.W.)                 .2914    .510     4.00    122.   2.666  1.253   .076
```

# FIGURE 6·2 — (contd)

```
25  DIST. I-2(S.W.)        TO
 30 JNCTN-(I-2 S)                .2515   .539    1.00     0.    .601   .540   .005
13  PUMPS (PD3-WEST)       TO
 31 DIST. A-3                    .4476   .531    3.62   458.   5.693  2.914   .247
31  DIST. A-3              TO
 32 DIST. B-3                    .4012   .390   42.20     0.   1.125  1.015   .010
32  DIST. B-3              TO
 33 DIST. C-3                    .0691   .250   12.80     0.    .642   .553   .006
33  DIST. C-3              TO
 34 DIST. E-3                    .0289   .206    3.24    21.    .653   .467   .017
36  JNCTN (I-3/W)          TO
 35 DIST. G-3                    .0276   .233    1.00     0.    .156   .140   .001
32  DIST. B-3              TO
 37 JNCTN (B-3/S)                .3115   .506    6.10     0.   1.276  1.201   .012
37  JNCTN (B-3 S)          TO
 38 DIST. D-3(WEST)              .2344   .447    6.47   218.   3.818  2.406   .125
40  DIST. D-3(EAST)        TO
 41 DIST. F-3(SOUTH)             .0059   .091   27.20     0.    .056   .056   .001
38  DIST. D-3(WEST)        TO
  9 DIST. B-4                    .2126   .423    6.47   359.   4.947  2.781   .197
40  DIST. D-3(EAST)        TO
  5 IMAGI RES.(PD3)              .0334   .170   10.80     0.    .156   .141   .001
16  DIST. F-2(S.E.)        TO
  4 CENTRAL RES.(PD2)            .0757   .309   .6.30     0.    .732   .676   .007
37  JNCTN (B-3 S)          TO
 39 DIST. D-3(CEN)               .0219   .131    4.62    22.    .749   .546   .013
35  DIST. G-3              TO
  8 DIST. H-3                    .0114   .138    3.62    10.    .340   .232   .004
24  JNCTN-(G-2 SE)         TO
 36 JNCTN (I-3/W)                .3298   .500    2.24   197.   2.919  1.695   .109
9   DIST. B-4              TO
  7 MIGANGA RES(PD4/WST)         .0174   .046    4.60     0.    .000   .000   .000
36  JNCTN (I-3 W)          TO
  6 NDIUKA RES(PD4EAST)          .1233   .126    1.00   290.    .397  1.620   .153
37  JNCTN (B-3 S)          TO
 40 DIST. D-3(EAST)              .0552   .263    3.09    93.   2.122  1.411   .141

        PROCESS COST...... (MILLION)      0.000
        TRANSPN COST...... (MILLION)    102.005
        TOTAL SYSTEM ..... (MILLION)    102.005
END OF REPORT
```

# FIGURE 6·3 — WATER SUPPLY NETWORK SOLUTION WITH PUMP CONSOLIDATION

```
                   PROBLEM SOLUTION
 PROCESSING CENTER COSTS
     CENTER                          FLOW        COST (MIL)
                                     (CMS)          SHS-M

BUBU RIVER                            1.58          0.00
MAKUTAPORA WELLS                      .39           0.00


 TRANSPORTATION COSTS
NODE          ROUTE                   FLOW      DIA      HEAD    PUMP       COST (SHS-M)
                                      (CMS)     (M)     LOSS M    HP   TOTAL  CAPITAL   OPER.

  1  BUBU RIVER              TO
  12 DIST. B-2                      1.5783     .879     17.33   3120.  35.177 17.108   1.615
 12  DIST. B-2               TO
  13 PUMPS (PD3-WEST)               1.2918     .831     12.90      0.   7.549  6.960    .070
 13  PUMPS (PD3-WEST)        TO
  14 DIST. D-2(S.W.)                 .8441     .714      3.62    107.   3.297  2.468    .076
 14  DIST. D-2(S.W.)         TO
  15 DIST. F-2(S.W.)                 .8349     .807      4.90      0.   4.739  4.562    .046
 15  DIST. F-2(S.W.)         TO
  16 DIST. F-2(S.E.)                 .7491     .656     11.50      0.   3.224  2.967    .030
  2  MAKUTAPORA WELLS        TO
  10 PUMPS (PD2)                     .3939     .568      6.10      0.   1.962  1.962    .019
 10  PUMPS (PD2)             TO
   3 MLIMWA RES.(PD1)                .3027     .482      6.95    422.   5.371  3.250    .233
  3  MLIMWA RES.(PD1)        TO
  11 DIST. A-1                       .2770     .365     45.40      0.   1.405  1.268    .013
 10  PUMPS (PD2)             TO
  17 DIST. E-2(WEST)                 .0912     .320      6.24    182.   2.889  1.742    .102
 15  DIST. F-2(S.W.)         TO
  18 DIST. F-2(N.W.)                 .0756     .136     33.50      0.    .240   .216    .002
 17  DIST. E-2(WEST)         TO
  19 DIST. E-2(CEN)                  .0754     .139     36.90      0.    .217   .195    .002
 19  DIST. E-2(CEN)          TO
  26 JNCTN-(H-2/N)                   .0497     .286      4.30      0.    .646   .602    .006
 18  DIST. F-2(N.W.)         TO
  20 DIST. G-2(N.W.)                 .0443     .323      2.10      0.    .549   .542    .005
 20  DIST. G-2(N.W.)         TO
  19 DIST. E-2(CEN)                  .0202     .139     11.30      0.    .122   .116    .001.
 23  JNCTN-(H-2 S)           TO
  27 DIST. H-2                       .0257     .164      6.10      0.    .112   .102    .001
 29  DIST. I-2               TO
  28 JNCTN-(H-2/S)                   .0257     .161      6.70      0.    .109   .099    .001
 30  JNCTN-(I-2/S)           TO
  29 DIST. I-2                       .0644     .234      5.20      0.    .139   .127    .001
 16  DIST. F-2(S.E.)         TO
  21 DIST. F-2(CEN)                  .0167     .122     22.60      0.    .169   .151    .002
 20  DIST. G-2(N.W.)         TO
  22 DIST. G-2(CEN)                  .0201     .209      2.20      0.    .213   .205    .002
 16  DIST. F-2(S.E.)         TO
  24 JNCTN-(G-2 SE)                  .6202     .451     24.80      0.    .662   .600    .006
 24  JNCTN-(G-2 SE)          TO
  25 DIST. I-2(S.W.)                 .2914     .530      4.00    122.   2.662  1.357    .076
```

# FIGURE 6·3 — (contd)

```
 25  DIST. I-2(S.W.)          TO
  30 JNCTN-(I-2/S)                .2515   .539    1.00      0.    .601   .540    .005
 13  PUMPS (PD3-WEST)         TO
  31 DIST. A-3                    .4476   .577    4.00   1013. 10.704  4.704    .536
 31  DIST. A-3                TO
  32 DIST. B-3                    .4012   .390   48.20      0.   1.125  1.015    .010
 32  DIST. B-3                TO
  33 DIST. C-3                    .0601   .193   46.70      0.    .488   .439    .004
 33  DIST. C-3                TO
  34 DIST. E-3                    .0289   .215    3.20      0.    .213   .200    .002
 36  JNCTN (I-3/W)            TO
  35 DIST. G-3                    .0276   .219    2.00     23.    .617   .426    .017
 32  DIST. B-3                TO
  37 JNCTN (B-3/S)                .3115   .506    6.10      0.   1.276  1.201    .012
 37  JNCTN (B-3/S)            TO
  38 DIST. D-3(WEST)              .2344   .334   30.10      0.    .702   .634    .006
 40  DIST. D-3(EAST)          TO
  41 DIST. F-3(SOUTH)             .0059   .091   27.20      0.    .056   .036    .001
 38  DIST. D-3(WEST)          TO
   9 DIST. B-4                    .2126   .423    6.47    359.   4.997  2.781    .197
 40  DIST. D-3(EAST)          TO
   5 IMAGI RES.(PD3)              .0334   .170   10.80      0.    .156   .141    .001
 16  DIST. F-2(S.E.)          TO
   4 CENTRAL RES.(PD2)            .0757   .309    6.30      0.    .732   .676    .007
 37  JNCTN (B-3/S)            TO
  39 DIST. D-3(CEN)               .0219   .130   31.00      0.    .208   .198    .002
 35  DIST. G-3                TO
   8 DIST. H-3                    .0114   .146    3.60      0.    .140   .130    .001
 24  JNCTN-(G-2 SE)           TO
  36 JNCTN (I-3/W)                .3289   .500    2.24    197.   2.918  1.695    .109
  9  DIST. B-4                TO
   7 MIGANGA RES(PD4 WST)         .0174   .046    4.60      0.    .000   .000    .000
 36  JNCTN (I-3/W)            TO
   6 NDIUKA RES(PD4EAST)          .1293   .126    1.00    250.   3.347  1.650    .153
 37  JNCTN (E-3 S)            TO
  40 DIST. D-3(EAST)              .0552   .267    5.10      0.    .700   .641    .006

         PROCESS COST...... (MILLION)     0.000
         TRANSPN COST...... (MILLION)   100.985
         TOTAL SYSTEM ..... (MILLION)   100.985
END OF REPORT.
```

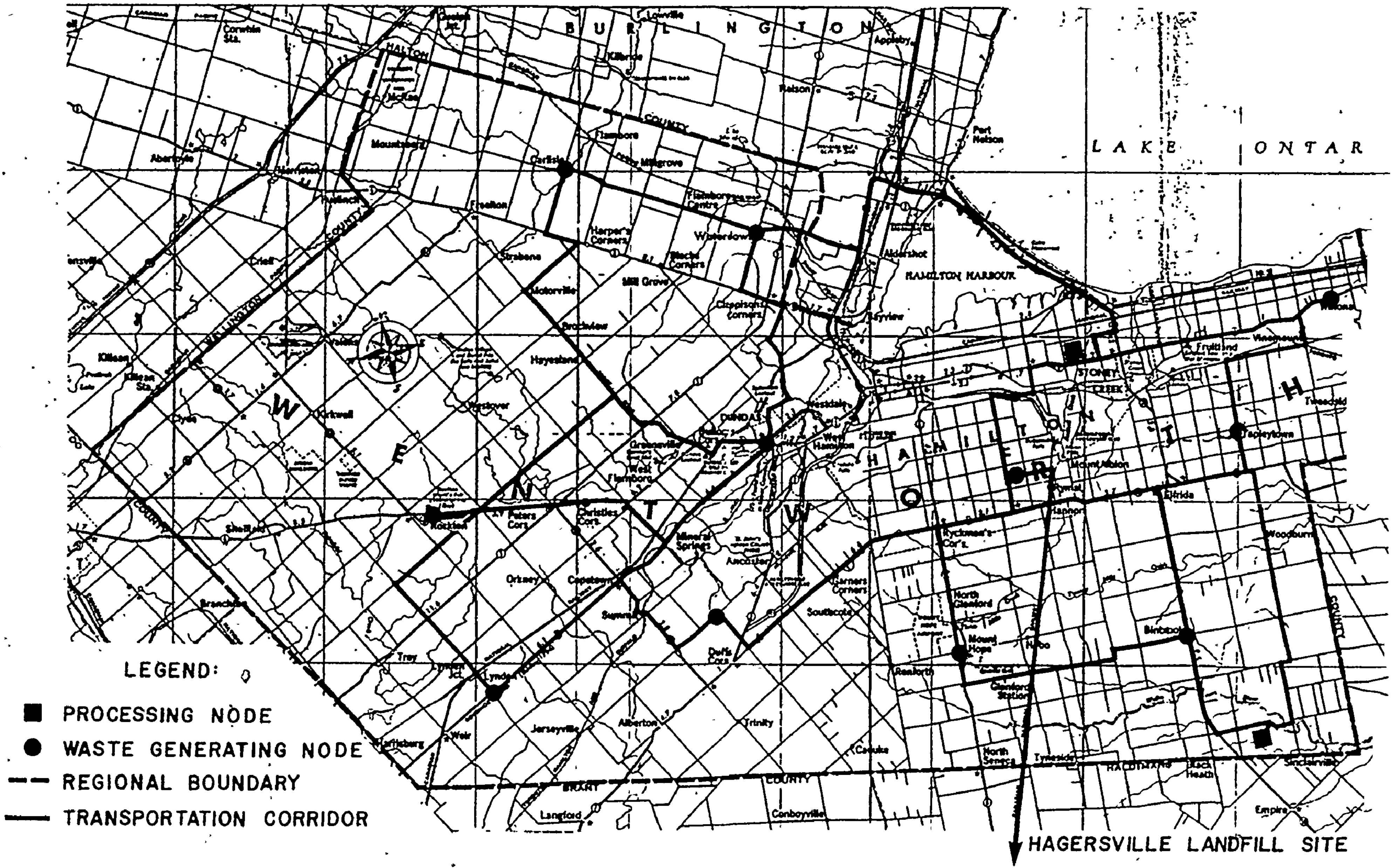


FIGURE 6·4 — SOLID WASTE SYSTEM

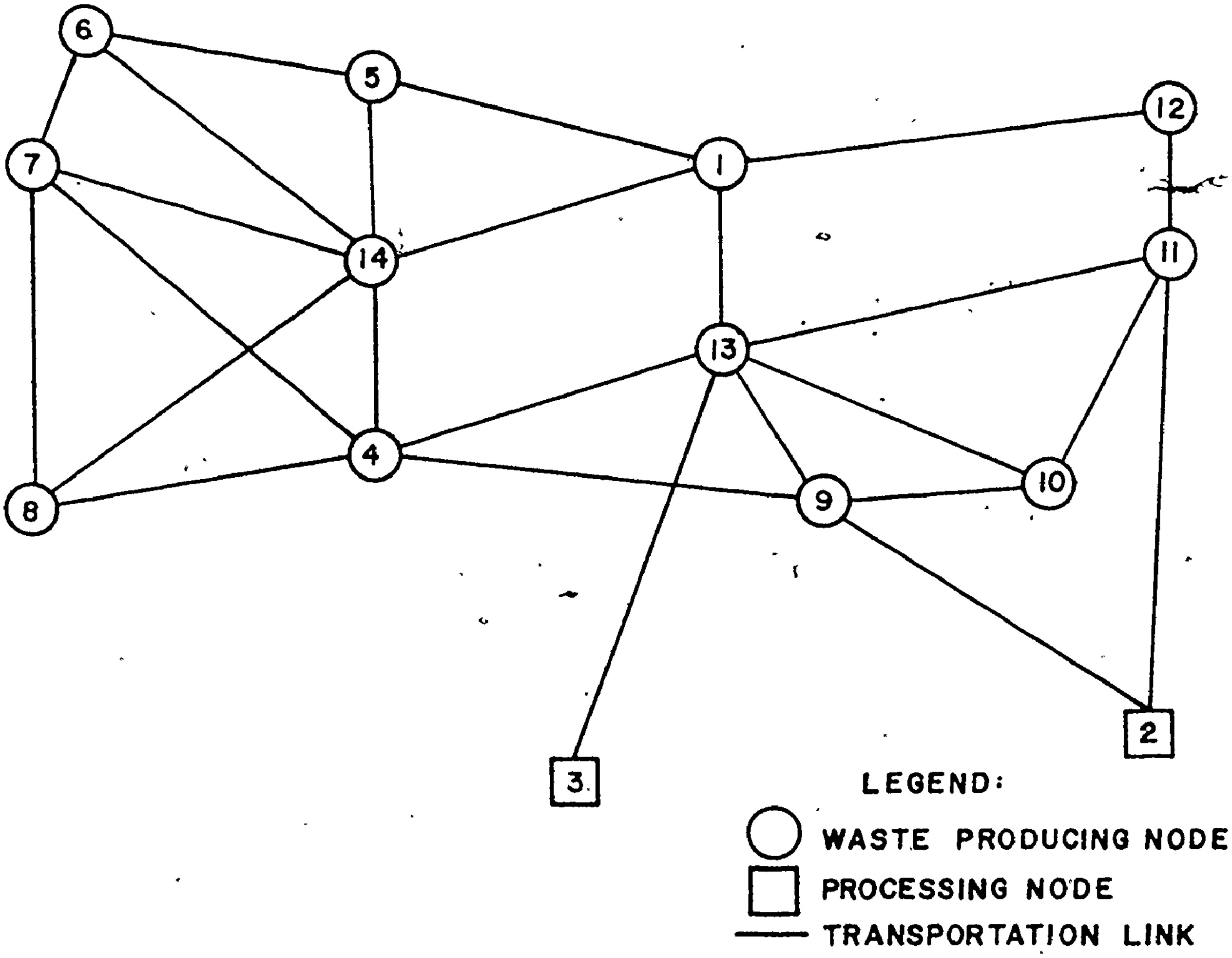
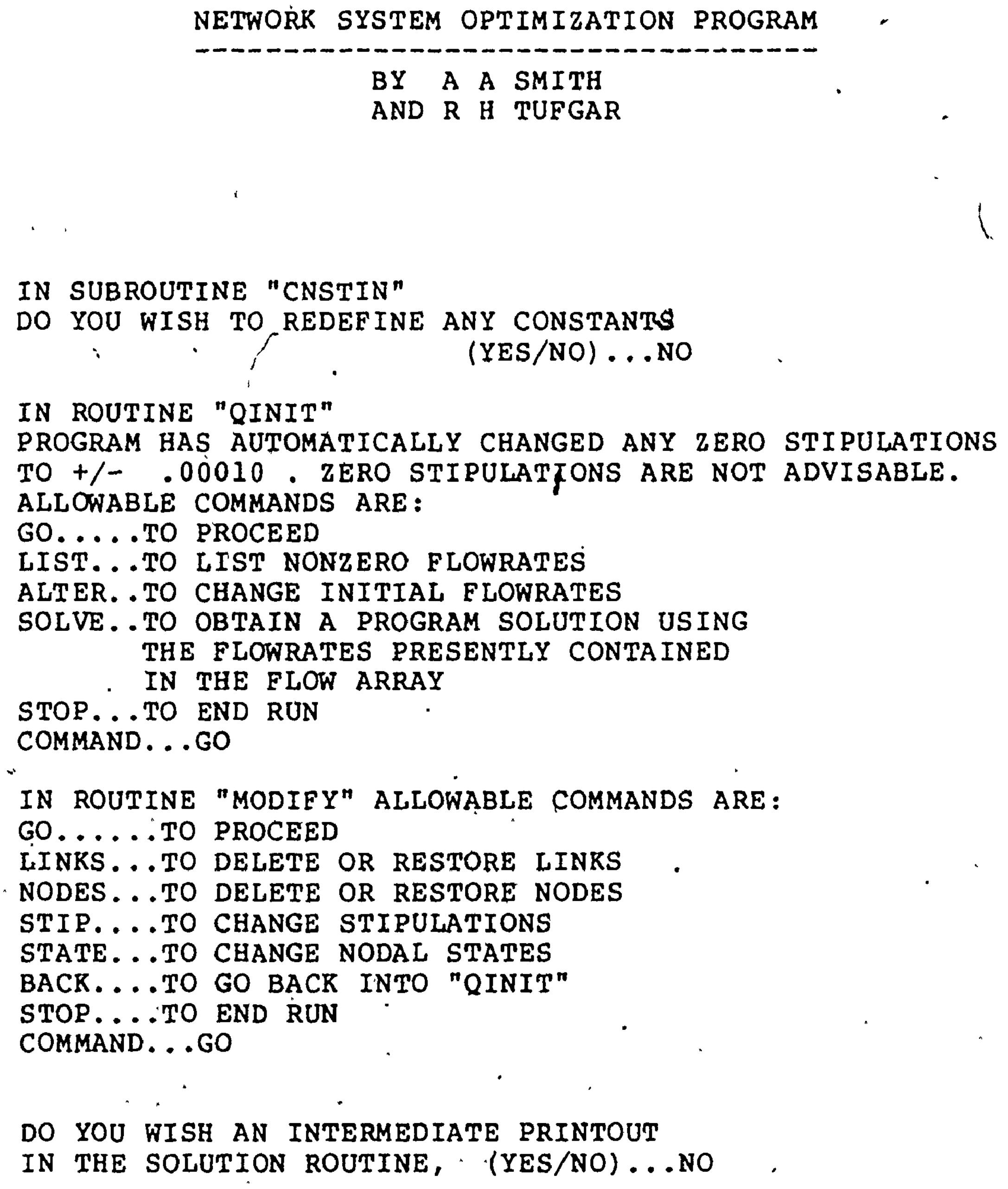


FIGURE 6·5 — SCHEMATIC OF SOLID WASTE DISPOSAL NETWORK

FIGURE 6·6 — SOLID WASTE NETWORK
MINIMUM COST SOLUTION

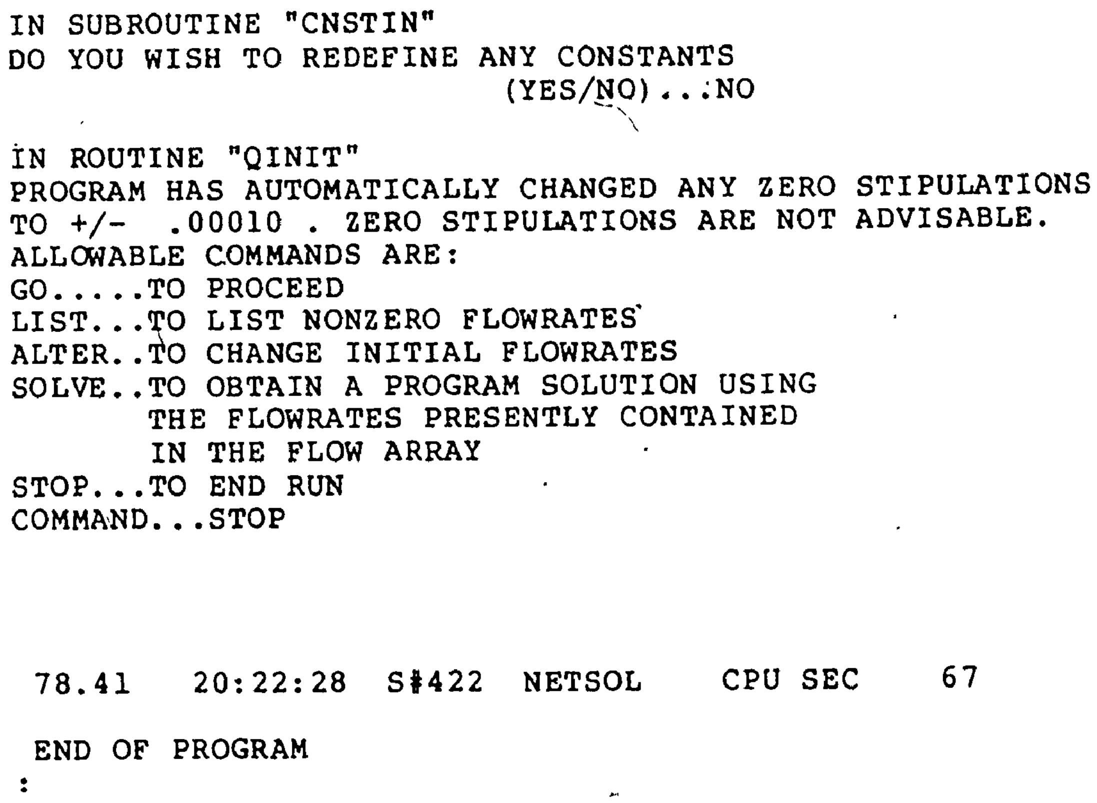


```
               NETWORK SYSTEM OPTIMIZATION PROGRAM
               -----------------------------------
                          BY  A A SMITH
                          AND R H TUFGAR




IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                           (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

## FIGURE 6·6 — (contd)

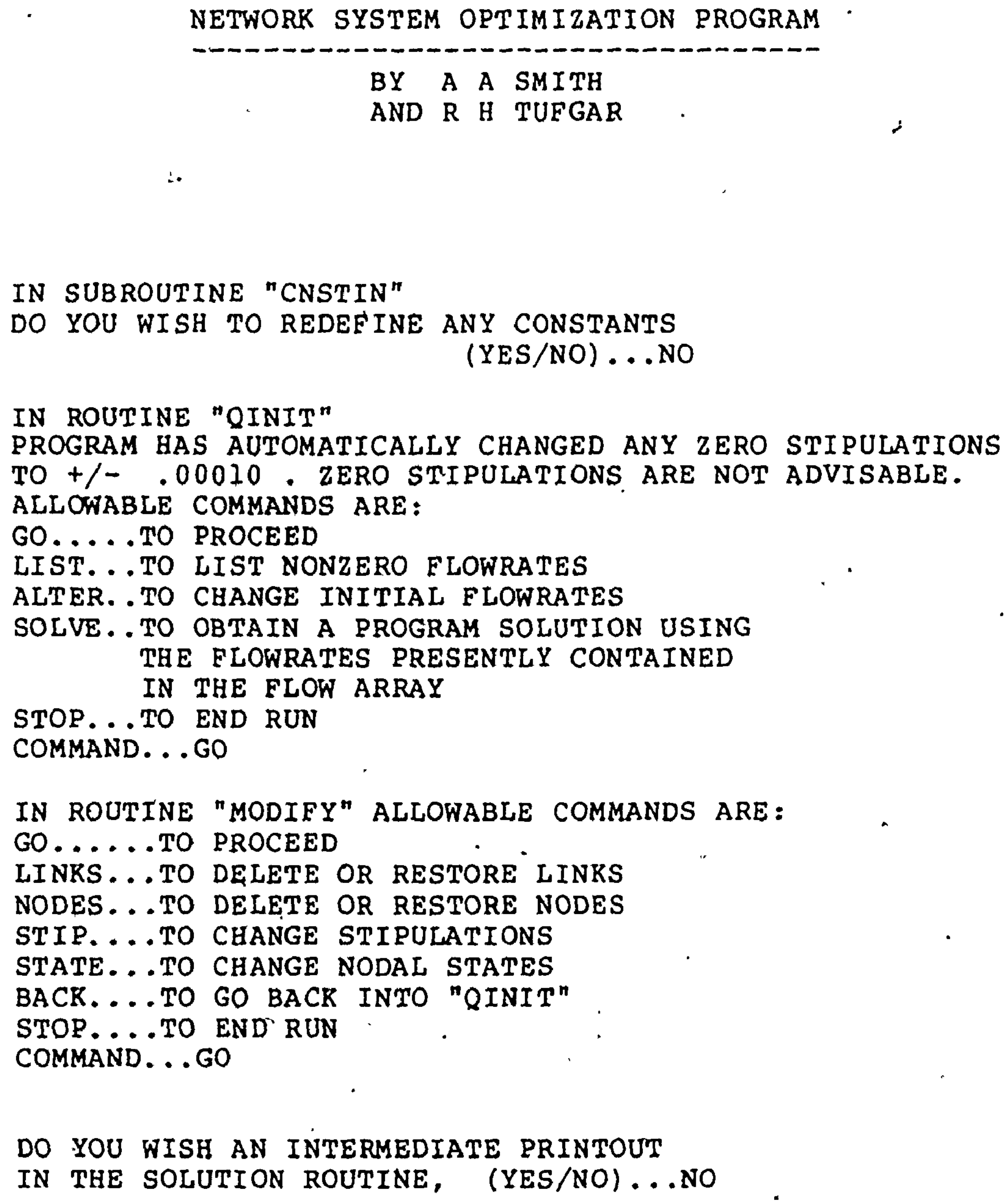
```
                      PROBLEM SOLUTION
   PROCESSING CENTER COSTS
       CENTER                     FLOW        COST (MIL)

  GLANBROOK                       2.610          85.28


   TRANSPORTATION COSTS
  NODE         ROUTE                FLOW  FLEET TONNAGE      COST (MIL)

   12  FRUITLAND              TO
     1 HAMILTON PROPER              .013              2.62             .84
    1  HAMILTON PROPER        TO
    13 HAMILTON MOUNTAIN           1.728            861.54            1.74
   14  DUNDAS                 TO
     1 HAMILTON PROPER              .315            726.92             .92
   10  BINBROOK               TO
     2 GLANBROOK                   2.610            418.29            2.28
    7  ROCTON                 TO
     4 ANCASTER                     .004              1.40             .64
    8  LYNDEN                 TO
     4 ANCASTER                     .006              2.15             .76
    4  ANCASTER               TO
    14 DUNDAS                       .150            162.26            1.03
    6  CARLISLE               TO
     5 WATERDOWN                    .007              1.53             .67
    5  WATERDOWN              TO
    14 DUNDAS                       .025             14.62             .71
    9  MOUNT HOPE             TO
    13 HAMILTON MOUNTAIN            .005              1.07             .56
   13  HAMILTON MOUNTAIN      TO
    10 BINBROOK                    2.609            682.94            2.42
   11  TAPLEYTON              TO
    12 FRUITLAND                    .001               .22             .13

 GLANBROOK LANDFILL SITE AREA =     626.75ACRES


             PROCESS COST...... (MILLION)     85.279
             TRANSPN COST...... (MILLION)     12.702
             TOTAL SYSTEM ..... (MILLION)     97.981
   END OF REPORT
```

FIGURE 6·6 — (contd)



```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                              (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP



 78.41   20:22:28  S#422  NETSOL    CPU SEC    67

 END OF PROGRAM
:
```

FIGURE 6·7 – SOLID WASTE NETWORK SOLUTION CHECK

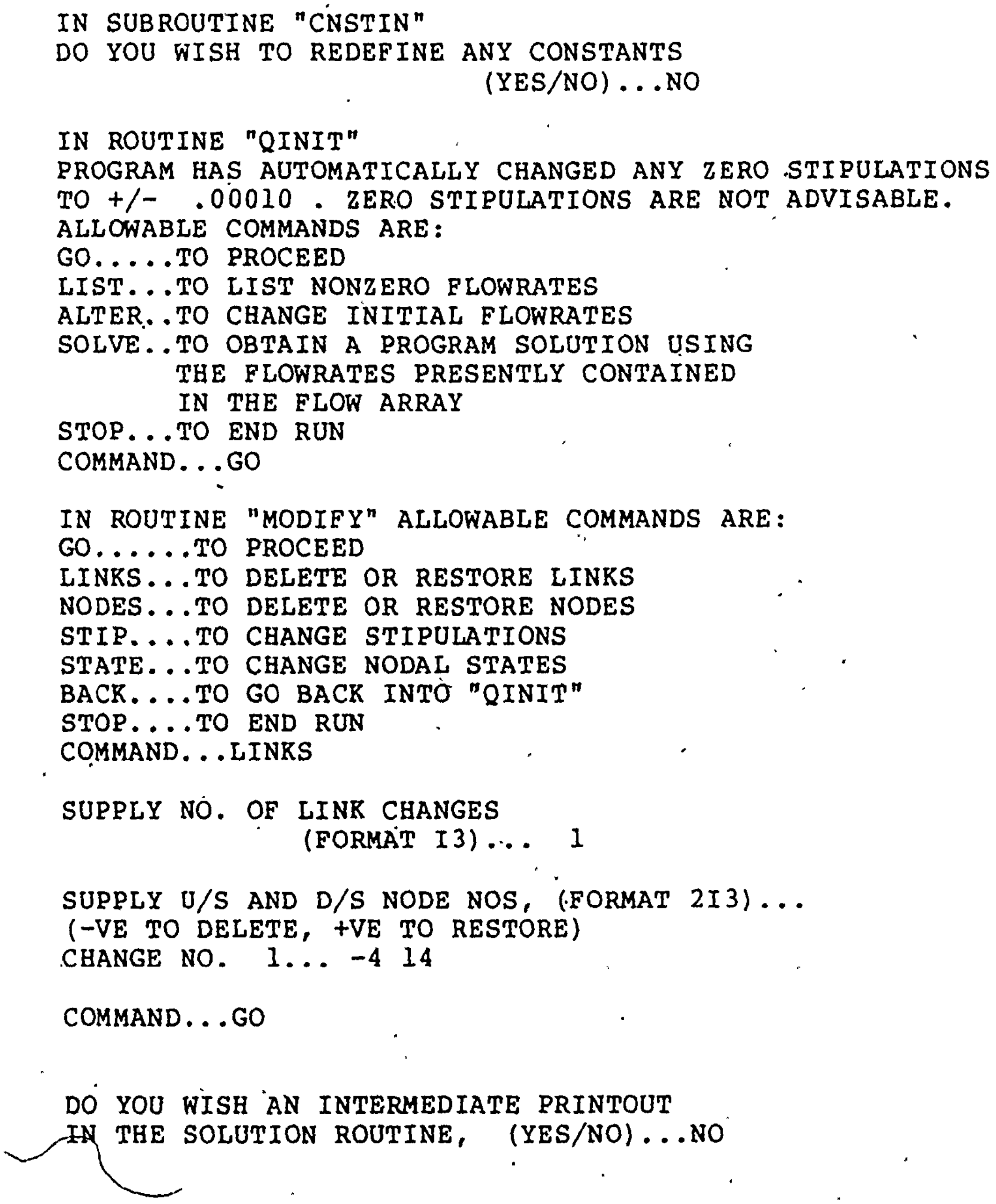


```
                NETWORK SYSTEM OPTIMIZATION PROGRAM
                -----------------------------------
                          BY  A A SMITH
                          AND R H TUFGAR


IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                          (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

## FIGURE 6·7 – (contd)

```
                    PROBLEM SOLUTION
 PROCESSING CENTER COSTS
    CENTER                      FLOW       COST (MIL)

GLANBROOK                       2.610        85.28


 TRANSPORTATION COSTS
NODE        ROUTE                 FLOW  FLEET TONNAGE     COST (MIL)

 12  FRUITLAND               TO
   1 HAMILTON PROPER              .013           2.62           .84
  1  HAMILTON PROPER         TO
  13 HAMILTON MOUNTAIN           1.728         861.54          1.74
 14  DUNDAS                  TO
   1 HAMILTON PROPER              .315         726.92           .92
 10  BINBROOK                TO
   2 GLANBROOK                   2.610         418.29          2.28
  7  ROCTON                  TO
   4 ANCASTER                     .004           1.40           .64
  8  LYNDEN                  TO
   4 ANCASTER                     .006           2.15           .76
  4  ANCASTER                TO
  14 DUNDAS                       .150         162.26          1.03
  6  CARLISLE                TO
   5 WATERDOWN                    .007           1.53           .67
  5  WATERDOWN               TO
  14 DUNDAS                       .025          14.62           .71
  9  MOUNT HOPE              TO
  13 HAMILTON MOUNTAIN            .005           1.07           .56
 13  HAMILTON MOUNTAIN       TO
  10 BINBROOK                    2.609         682.94          2.42
 11  TAPLEYTON               TO
  12 FRUITLAND                    .001            .22           .13

GLANBROOK LANDFILL SITE AREA =    626.75ACRES


           PROCESS COST...... (MILLION)    85.279
           TRANSPN COST...... (MILLION)    12.702
           TOTAL SYSTEM ..... (MILLION)    97.981
  END OF REPORT
```

FIGURE 6·7 — (contd)



```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                            (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...LINKS

SUPPLY NO. OF LINK CHANGES
              (FORMAT I3)...  1

SUPPLY U/S AND D/S NODE NOS, (FORMAT 2I3)...
(-VE TO DELETE, +VE TO RESTORE)
CHANGE NO.  1... -4 14

COMMAND...GO

DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

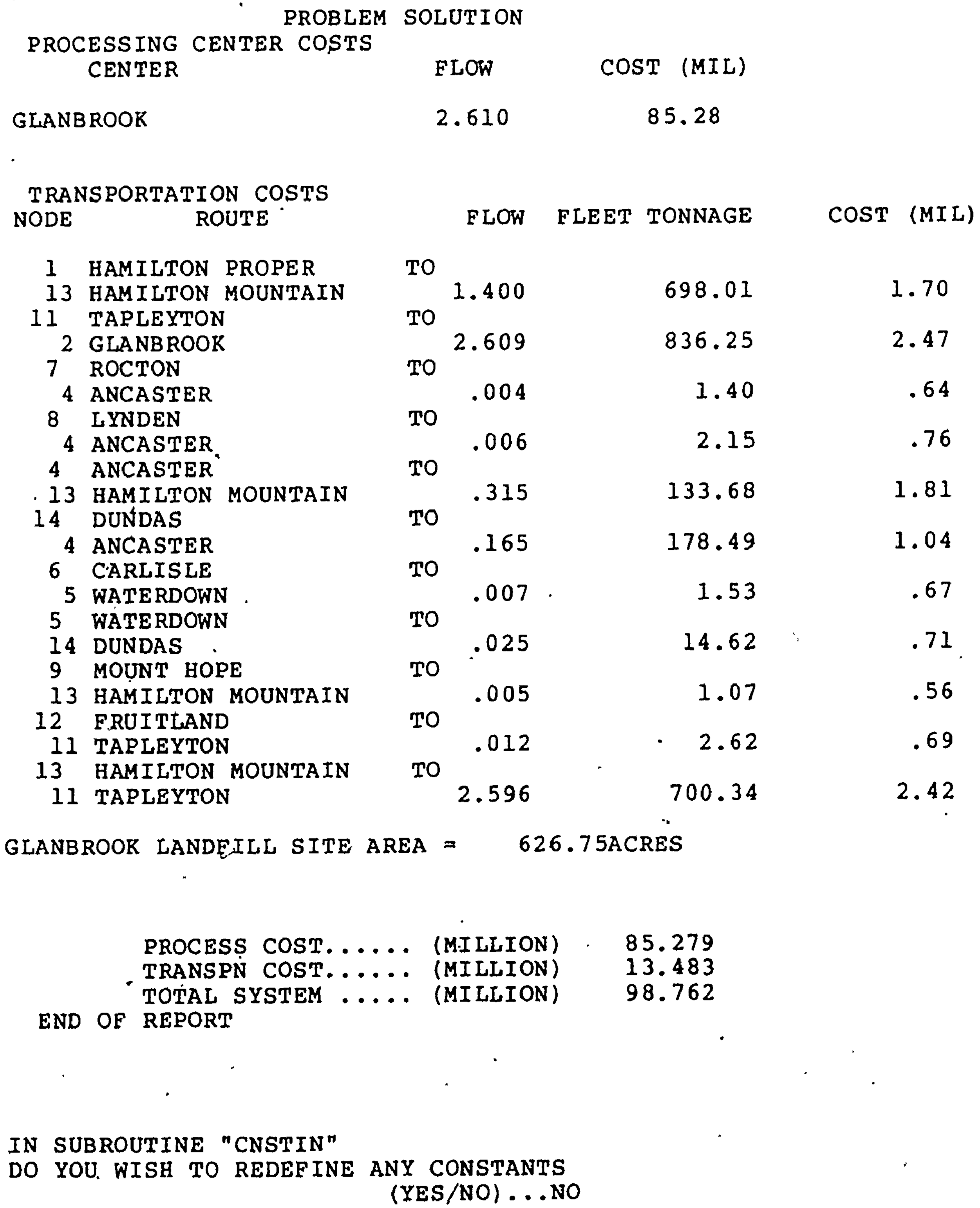
## FIGURE 6·7 — (contd)

```
                    PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                           FLOW        COST (MIL)

 GLANBROOK                             2.610          85.28


  TRANSPORTATION COSTS
 NODE        ROUTE                       FLOW   FLEET TONNAGE      COST (MIL)

  12   FRUITLAND                 TO
    1 HAMILTON PROPER                    .013             2.62             .84
   1  HAMILTON PROPER            TO
   13 HAMILTON MOUNTAIN                 1.578           786.75            1.72
  14   DUNDAS                    TO
    1 HAMILTON PROPER                    .165           380.77             .86
  10   BINBROOK                  TO
    2 GLANBROOK                         2.610           418.29            2.28
   7   ROCTON                    TO
    4 ANCASTER                           .004             1.40             .64
   8   LYNDEN                    TO
    4 ANCASTER                           .006             2.15             .76
   4   ANCASTER                  TO
   13 HAMILTON MOUNTAIN                  .150            63.66            1.61
   6   CARLISLE                  TO
    5 WATERDOWN                          .007             1.53             .67
   5   WATERDOWN                 TO
   14 DUNDAS                             .025            14.62             .71
   9   MOUNT HOPE                TO
   13 HAMILTON MOUNTAIN                  .005             1.07             .56
  13   HAMILTON MOUNTAIN         TO
   10 BINBROOK                          2.609           682.94            2.42
  11   TAPLEYTON                 TO
   12 FRUITLAND                          .001              .22             .13

GLANBROOK LANDFILL SITE AREA =    626.75ACRES



         PROCESS COST...... (MILLION)    85.279
         TRANSPN COST...... (MILLION)    13.208
         TOTAL SYSTEM ..... (MILLION)    98.487
  END OF REPORT
```

FIGURE 6·7 – (contd)

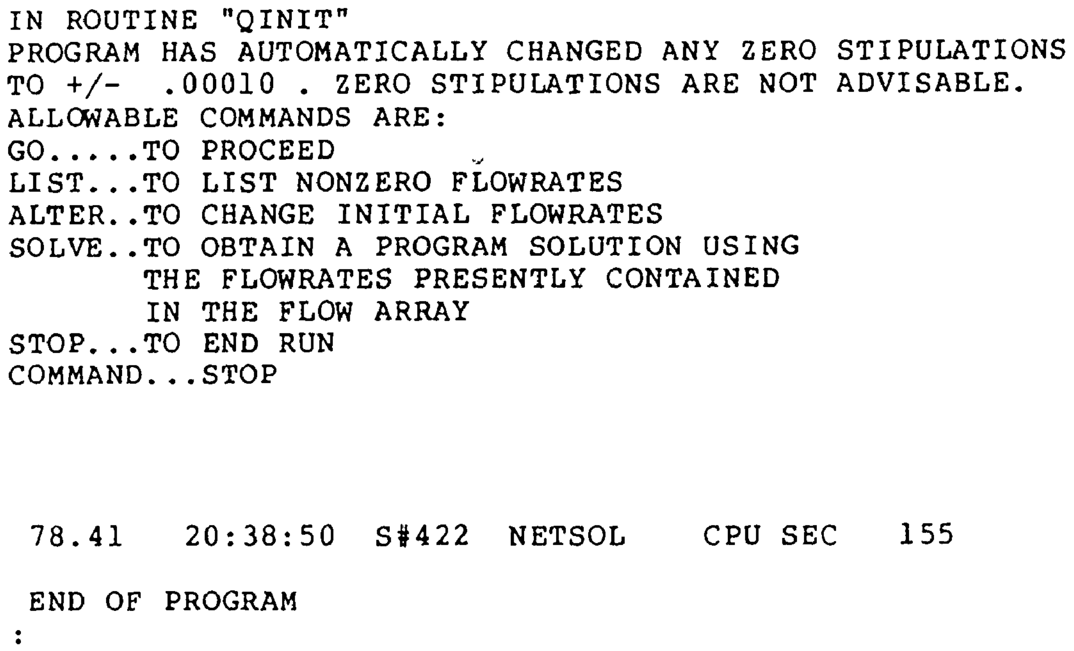
```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                         (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...LINKS

SUPPLY NO. OF LINK CHANGES
              (FORMAT I3)...  1

SUPPLY U/S AND D/S NODE NOS, (FORMAT 2I3)...
(-VE TO DELETE, +VE TO RESTORE)
CHANGE NO.  1...-14  1

COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

FIGURE 6·7 – (contd)



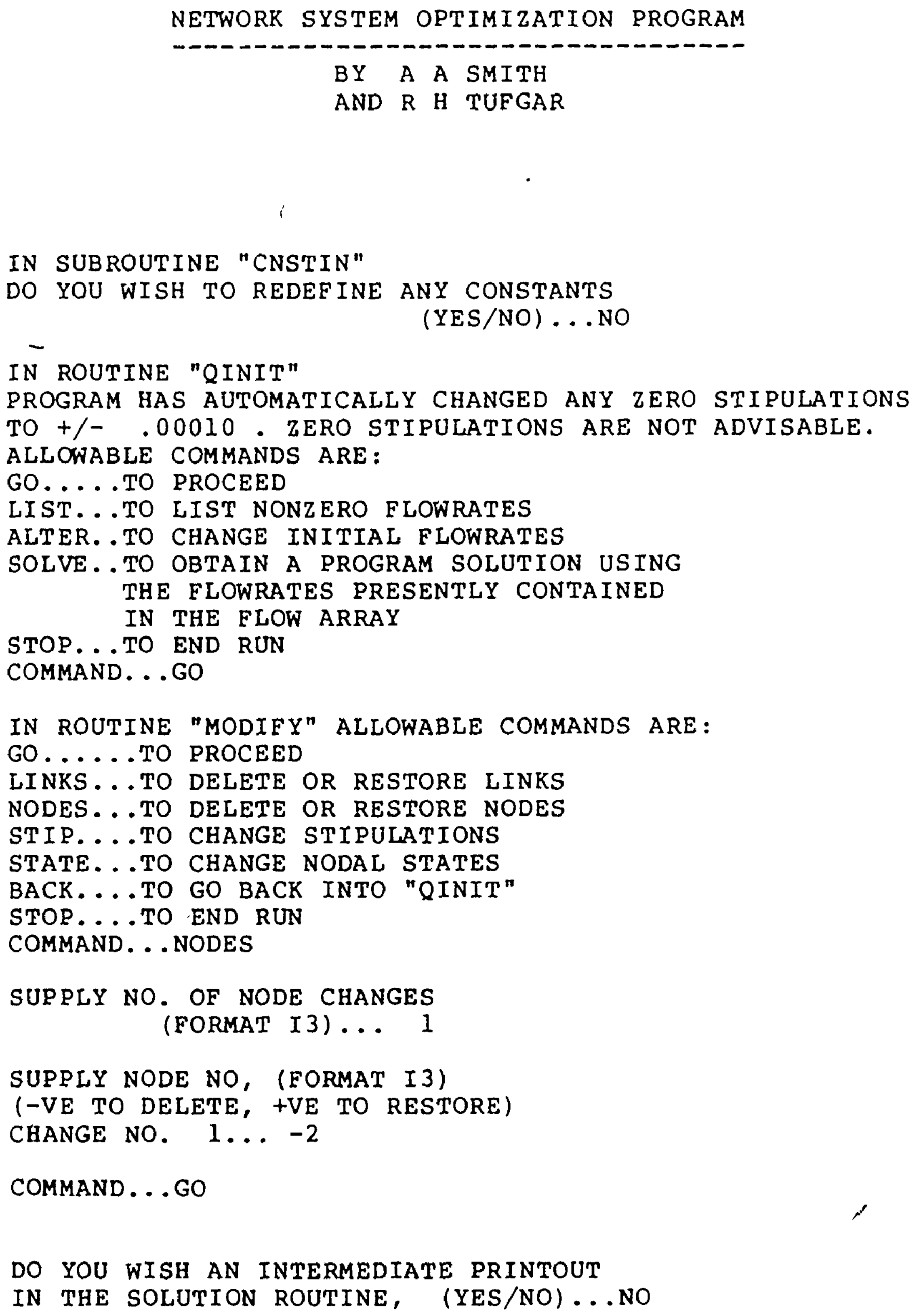
```
                    PROBLEM SOLUTION
 PROCESSING CENTER COSTS
    CENTER                      FLOW       COST (MIL)

GLANBROOK                       2.610        85.28

 TRANSPORTATION COSTS
NODE         ROUTE                FLOW  FLEET TONNAGE     COST (MIL)

  1  HAMILTON PROPER       TO
  13 HAMILTON MOUNTAIN      1.400           698.01          1.70
 11  TAPLEYTON             TO
   2 GLANBROOK              2.609           836.25          2.47
  7  ROCTON                TO
   4 ANCASTER                .004             1.40           .64
  8  LYNDEN                TO
   4 ANCASTER                .006             2.15           .76
  4  ANCASTER              TO
  13 HAMILTON MOUNTAIN       .315           133.68          1.81
 14  DUNDAS                TO
   4 ANCASTER                .165           178.49          1.04
  6  CARLISLE              TO
   5 WATERDOWN               .007             1.53           .67
  5  WATERDOWN             TO
  14 DUNDAS                  .025            14.62           .71
  9  MOUNT HOPE            TO
  13 HAMILTON MOUNTAIN       .005             1.07           .56
 12  FRUITLAND             TO
  11 TAPLEYTON               .012             2.62           .69
 13  HAMILTON MOUNTAIN     TO
  11 TAPLEYTON              2.596           700.34          2.42
GLANBROOK LANDFILL SITE AREA =    626.75ACRES


           PROCESS COST...... (MILLION)    85.279
           TRANSPN COST...... (MILLION)    13.483
           TOTAL SYSTEM ..... (MILLION)    98.762
  END OF REPORT



IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                             (YES/NO)...NO
```

FIGURE 6·7 — (contd)



```
IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP



 78.41   20:38:50  S#422  NETSOL    CPU SEC   155

 END OF PROGRAM
:
```

FIGURE 6-8 – SOLID WASTE NETWORK ALTERNATE SOLUTIONS



```
            NETWORK SYSTEM OPTIMIZATION PROGRAM
            ------------------------------------
                       BY  A A SMITH
                       AND R H TUFGAR




IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                           (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...NODES

SUPPLY NO. OF NODE CHANGES
         (FORMAT I3)...  1

SUPPLY NODE NO, (FORMAT I3)
(-VE TO DELETE, +VE TO RESTORE)
CHANGE NO.  1... -2

COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

FIGURE 6·8 – (contd)

```
                    PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                      FLOW       COST (MIL)

 HAGERSVILLE                      2.610         28.29


  TRANSPORTATION COSTS
 NODE          ROUTE                FLOW  FLEET TONNAGE      COST (MIL)

  12   FRUITLAND             TO
    1  HAMILTON PROPER              .013           2.62             .84
   1   HAMILTON PROPER       TO
   13  HAMILTON MOUNTAIN           1.728         861.54            1.74
  14   DUNDAS                TO
    1  HAMILTON PROPER              .315         726.92             .92
  13   HAMILTON MOUNTAIN     TO
    3  HAGERSVILLE                 2.610            .00           79.17
   7   ROCTON                TO
    4  ANCASTER                     .004           1.40             .64
   8   LYNDEN                TO
    4  ANCASTER                     .006           2.15             .76
   4   ANCASTER              TO
   14  DUNDAS                       .150         162.26            1.03
   6   CARLISLE              TO
    5  WATERDOWN                    .007           1.53             .67
   5   WATERDOWN             TO
   14  DUNDAS                       .025          14.62             .71
   9   MOUNT HOPE            TO
   13  HAMILTON MOUNTAIN            .005           1.07             .56
  10   BINBROOK              TO
   13  HAMILTON MOUNTAIN            .001            .29             .19
  11   TAPLEYTON             TO
   12  FRUITLAND                    .001            .22             .13

RAIL FLOWRATE TO HAGERSVILLE =  50056.72TONS/WK


HAGERSVILLE LANDFILL SITE AREA =    626.75ACRES


         PROCESS COST...... (MILLION)    28.288
         TRANSPN COST...... (MILLION)    87.368
         TOTAL SYSTEM ..... (MILLION)   115.656
  END OF REPORT
```

FIGURE 6·8 — (contd)

```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                             (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...NODES

SUPPLY NO. OF NODE CHANGES
         (FORMAT I3)...  1

SUPPLY NODE NO, (FORMAT I3)
(-VE TO DELETE, +VE TO RESTORE)
CHANGE NO.  1... -3

COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

## FIGURE 6·8 – (contd)

```
                    PROBLEM  SOLUTION
  PROCESSING CENTER COSTS
      CENTER                    FLOW       COST (MIL)

HAMILTON PROPER                 2.610        380.48


  TRANSPORTATION COSTS
NODE         ROUTE                FLOW  FLEET TONNAGE      COST (MIL)

  12  FRUITLAND               TO
    1 HAMILTON PROPER             .013           2.62             .84
  13  HAMILTON MOUNTAIN       TO
    1 HAMILTON PROPER             .882         439.84            1.61
  14  DUNDAS                  TO
    1 HAMILTON PROPER             .315         726.92             .92
   7  ROCTON                  TO
    4 ANCASTER                    .004           1.40             .64
   8  LYNDEN                  TO
    4 ANCASTER                    .006           2.15             .76
   4  ANCASTER                TO
   14 DUNDAS                      .150         162.26            1.03
   6  CARLISLE                TO
    5 WATERDOWN                   .007           1.53             .67
   5  WATERDOWN               TO
   14 DUNDAS                      .025          14.62             .71
   9  MOUNT HOPE              TO
   13 HAMILTON MOUNTAIN           .005           1.07             .56
  10  BINBROOK                TO
   13 HAMILTON MOUNTAIN           .001            .29             .19
  11  TAPLEYTON               TO
   12 FRUITLAND                   .001            .22             .13


        PROCESS COST...... (MILLION)   380.477
        TRANSPN COST...... (MILLION)     8.069
        TOTAL SYSTEM ..... (MILLION)   388.546
  END OF REPORT
```

FIGURE 6·8 – (contd)

```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                         (YES/NO)...YES


SUPPLY NEW INTEREST RATE (%) (FORMAT F10.0)... 4.5


SUPPLY NEW INCINERATOR EFFICIENCY (%)
                      (FORMAT F10.0)... 75.0

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE..
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

# FIGURE 6·8 — (contd)

```
                    PROBLEM SOLUTION
 PROCESSING CENTER COSTS
     CENTER                     FLOW       COST (MIL)

HAMILTON PROPER                 2.610        178.31


 TRANSPORTATION COSTS
NODE          ROUTE               FLOW  FLEET TONNAGE      COST (MIL)

 12  FRUITLAND               TO
   1 HAMILTON PROPER              .013            2.62            .84
 13  HAMILTON MOUNTAIN       TO
   1 HAMILTON PROPER              .882          439.84           1.61
 14  DUNDAS                  TO
   1 HAMILTON PROPER              .315          726.92            .92
   7 ROCTON                  TO
   4 ANCASTER                     .004            1.40            .64
   8 LYNDEN                  TO
   4 ANCASTER                     .006            2.15            .76
   4 ANCASTER                TO
  14 DUNDAS                       .150          162.26           1.03
   6 CARLISLE                TO
   5 WATERDOWN                    .007            1.53            .67
   5 WATERDOWN               TO
  14 DUNDAS                       .025           14.62            .71
   9 MOUNT HOPE              TO
  13 HAMILTON MOUNTAIN            .005            1.07            .56
  10 BINBROOK                TO
  13 HAMILTON MOUNTAIN            .001             .29            .19
  11 TAPLEYTON               TO
  12 FRUITLAND                    .001             .22            .13


        PROCESS COST...... (MILLION)   178.312
        TRANSPN COST...... (MILLION)     8.069
        TOTAL SYSTEM ..... (MILLION)   186.381
 END OF REPORT
```

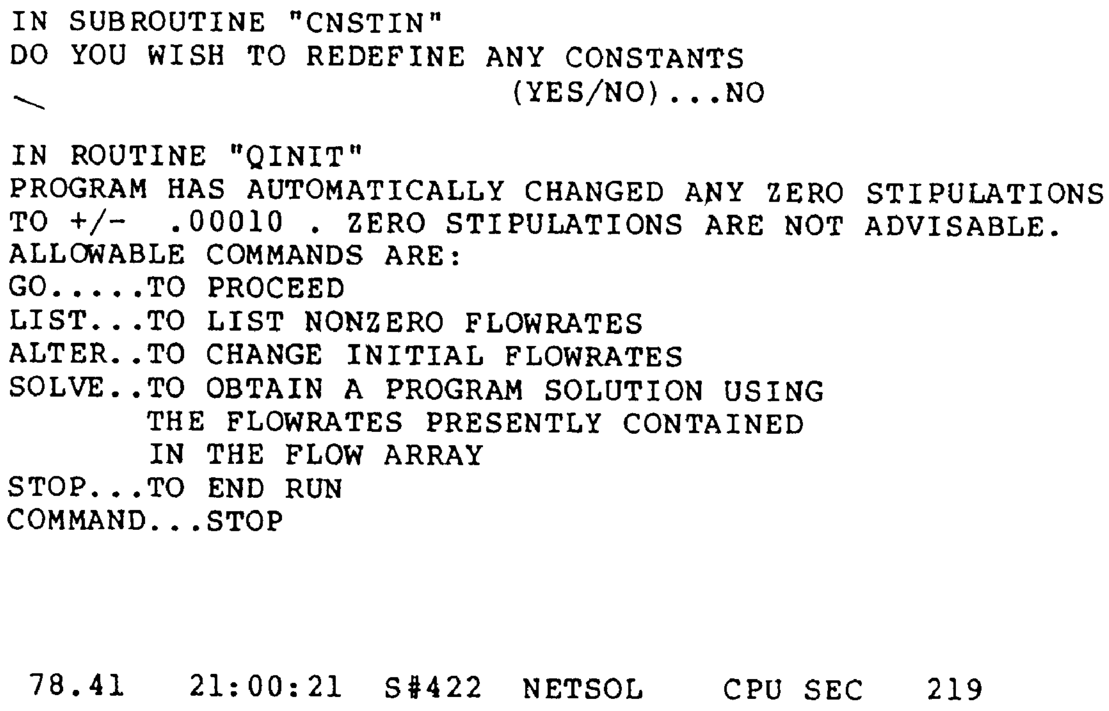
FIGURE 6·8 – (contd)

```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                         (YES/NO)...YES


SUPPLY NEW INTEREST RATE (%) (FORMAT F10.0)... 4.5


SUPPLY NEW INCINERATOR EFFICIENCY (%)
                      (FORMAT F10.0)... 100.0

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO


DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...NO
```

FIGURE 6·8 — (contd)

```
                 PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                     FLOW       COST (MIL)

 HAMILTON PROPER                2.610         148.61


  TRANSPORTATION COSTS
 NODE         ROUTE                FLOW  FLEET TONNAGE      COST (MIL)

  12  FRUITLAND              TO
    1 HAMILTON PROPER              .013           2.62             .84
  13  HAMILTON MOUNTAIN      TO
    1 HAMILTON PROPER              .882         439.84            1.61
  14  DUNDAS                 TO
    1 HAMILTON PROPER              .315         726.92             .92
   7  ROCTON                 TO
    4 ANCASTER                     .004           1.40             .64
   8  LYNDEN                 TO
    4 ANCASTER                     .006           2.15             .76
   4  ANCASTER               TO
   14 DUNDAS                       .150         162.26            1.03
   6  CARLISLE               TO
    5 WATERDOWN                    .007           1.53             .67
   5  WATERDOWN              TO
   14 DUNDAS                       .025          14.62             .71
   9  MOUNT HOPE             TO
   13 HAMILTON MOUNTAIN            .005           1.07             .56
  10  BINBROOK               TO
   13 HAMILTON MOUNTAIN            .001            .29             .19
  11  TAPLEYTON              TO
   12 FRUITLAND                    .001            .22             .13


        PROCESS COST...... (MILLION)   148.611
        TRANSPN COST...... (MILLION)     8.069
        TOTAL SYSTEM ..... (MILLION)   156.681
  END OF REPORT
```

FIGURE 6·8 – (contd)



```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                           (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP



 78.41   21:00:21  S#422  NETSOL    CPU SEC   219
```

# TABLE 6·1 – WATER SUPPLY PROBLEM DATA FILE

```
 41     2    54    MAXIMUM DAY
 1    BUBU RIVER              1127.8     +2.3634
 2    MAKUTAPORA WELLS        1112.5     +0.3939
 3    MLIMWA RES.(PD1)        1151.0     -0.0257
 4    CENTRAL RES.(PD2)       1158.2     -0.0757
 5    IMAGI RES.(PD3)         1196.3     -0.0334
 6    NDIUKA RES(PD4EAST)     1234.4     -0.1283
 7    MIGANGA RES(PD4/WST)    1234.4     -0.0174
10    PUMPS (PD2)             1106.4      0.0
11    DIST. A-1               1105.6     -0.2770
12    DIST. B-2               1185.0     -0.2865
13    PUMPS (PD3-WEST)        1181.0      0.0
14    DIST. D-2(S.W.)         1177.2     -0.0092
15    DIST. F-2(S.W.)         1172.3     -0.0102
16    DIST. F-2(S.E.)         1164.5     -0.0365
17    DIST. E-2(WEST)         1166.0     -0.0158
18    DIST. F-2(N.W.)         1138.8     -0.0263
19    DIST. E-2(CEN)          1129.1     -0.0459
20    DIST. G-2(N.W.)         1136.7     -0.0090
21    DIST. F-2(CEN)          1141.9     -0.0167
22    DIST. G-2(CEN)          1138.2     -0.0201
23    JNCTN -(G-2/NE)         1137.3      0.0
24    JNCTN-(G-2/SE)          1139.7      0.0
25    DIST. I-2(S.W.)         1150.1     -0.0399
26    JNCTN-(H-2/N)           1124.8     -0.0497
27    DIST. H-2               1130.9     -0.0257
28    JNCTN-(H-2/S)           1137.0      0.0
29    DIST. I-2               1143.7     -0.0387
30    JNCTN-(I-2/S)           1148.9     -0.1871
31    DIST. A-3               1215.0     -0.0464
32    DIST. B-3               1166.8     -0.0296
33    DIST. C-3               1154.0     -0.0312
34    DIST. E-3               1171.4     -0.0289
35    DIST. G-3               1155.9     -0.0162
36    JNCTN (I-3/W)           1157.1     -0.1729
37    JNCTN (B-3/S)           1160.7      0.0
38    DIST. D-3(WEST)         1185.1     -0.0218
39    DIST. D-3(CEN)          1184.2     -0.0219
40    DIST. D-3(EAST)         1207.1     -0.0159
41    DIST. F-3(SOUTH)        1179.9     -0.0059
 8    DIST. H-3               1176.0     -0.0114
 9    DIST. B-4               1239.0     -0.1952
    1    12    2440.0
   12    13    2300.0
   13    14     790.0
   14    15    1630.0
   15    16    1760.0
    2    10    1500.0
   10     3    1300.0
```

## TABLE 6·1 — (contd)

| | | |
|---|---|---|
| 10 | 3 | 1300.0 |
| 3 | 11 | 2500.0 |
| 10 | 17 | 1250.0 |
| 17 | 18 | 800.0 |
| 18 | 15 | 930.0 |
| 17 | 19 | 870.0 |
| 19 | 26 | 1680.0 |
| 18 | 20 | 1280.0 |
| 20 | 23 | 1280.0 |
| 23 | 28 | 770.0 |
| 19 | 20 | 695.0 |
| 26 | 27 | 620.0 |
| 27 | 28 | 525.0 |
| 28 | 29 | 520.0 |
| 29 | 30 | 450.0 |
| 18 | 21 | 980.0 |
| 21 | 16 | 1030.0 |
| 20 | 16 | 1275.0 |
| 20 | 22 | 825.0 |
| 22 | 24 | 1070.0 |
| 22 | 23 | 650.0 |
| 23 | 25 | 970.0 |
| 16 | 24 | 820.0 |
| 24 | 25 | 900.0 |
| 25 | 30 | 500.0 |
| 13 | 31 | 600.0 |
| 31 | 32 | 1800.0 |
| 32 | 33 | 1920.0 |
| 33 | 34 | 780.0 |
| 34 | 35 | 550.0 |
| 35 | 36 | 500.0 |
| 32 | 37 | 1300.0 |
| 37 | 38 | 1430.0 |
| 38 | 39 | 1050.0 |
| 39 | 40 | 1200.0 |
| 40 | 41 | 760.0 |
| 41 | 8 | 465.0 |
| 8 | 36 | 1460.0 |
| 38 | 9 | 1300.0 |
| 40 | 5 | 700.0 |
| 16 | 4 | 1700.0 |
| 37 | 33 | 1200.0 |
| 37 | 39 | 1200.0 |
| 35 | 8 | 750.0 |
| 24 | 36 | 450.0 |
| 9 | 7 | 1.0 |
| 36 | 6 | 1.0 |
| 37 | 40 | 1950.0 |

# TABLE 6·2 – NODAL STATE CHANGES FOR PUMP CONSOLIDATION

| NODE | DESCRIPTION | ELEVATIONS (meters) | | |
|---|---|---|---|---|
| | | OLD | NEW | CHANGE |
| 12 | DIST B-2 | 1185.0 | 1193.9 | 8.9 |
| 14 | DIST D-2 (S.W.) | 1177.2 | 1180.9 | 3.7 |
| 15 | DIST F-2 (S.W.) | 1172.3 | 1176.0 | 3.7 |
| 18 | DIST F-2 (N.W.) | 1138.8 | 1142.5 | 3.7 |
| 20 | DIST G-2 (N.W.) | 1136.7 | 1140.4 | 3.7 |
| 31 | DIST A-3 | 1215.0 | 1269.5 | 54.5 |
| 32 | DIST B-3 | 1166.8 | 1221.3 | 54.5 |
| 33 | DIST C-3 | 1154.0 | 1174.6 | 20.6 |
| 35 | DIST G-3 | 1155.9 | 1179.6 | 23.7 |
| 37 | JNCTN (B-3/S) | 1160.7 | 1215.2 | 54.5 |

## TABLE 6.3 — DATA FILE FOR SOLID WASTE NETWORK

```
 14    3   26
 1    HAMILTON PROPER                             1.400
 2    GLANBROOK                                    0.00
 3    HAGERSVILLE                                 0.000
 4    ANCASTER                                    0.140
 5    WATERDOWN                                   0.018
 6    CARLISLE                                    0.007
 7    ROCTON                                      0.004
 8    LYNDEN                                      0.006
 9    MOUNT HOPE                                  0.005
10    BINBROOK                                    0.001
11    TAPLEYTON                                   0.001
12    FRUITLAND                                   0.012
13    HAMILTON MOUNTAIN                           0.876
14    DUNDAS                      VELOCITY*       0.140
   1    5   13.2                    (27)
   1   12    7.8                    (31)
   1   13   11.2                    (18)
   1   14   14.4                    ( 5)
   2    9   14.8                    (30)
   2   10    6.0                    (30)
   2   11   12.0                    (30)
   [illegible]   13   16.6                    ( 1)
   4    7   13.1                    (30)
   4    8   13.4                    (30)
   4    9   13.4                    (30)
   4   13   14.3                    (27)
   4   14   13.5                    (10)
   5    6    8.2                    (30)
   5   14    7.3                    (10)
   6    7   19.3                    (30)
   6   14   13.7                    (24)
   7    8   19.3                    (30)
   7   14   16.7                    (28)
   8   14   16.1                    (28)
   9   10   10.4                    (30)
   9   13    8.0                    (30)
  10   11   10.4                    (30)
  10   13    9.8                    (30)
  11   12    6.0                    (22)
  11   13   10.1                    (30)
```

* Velocity data - not included in data file used for network problem.

# CHAPTER 7

## CONCLUSIONS AND DISCUSSION

The objective of this thesis is to develop an econometric model useful in solving network problems. The preceding chapters discuss the development of the model package, NETSOL, in detail while this chapter provides an overview of the work done. The econometric model is constructed to answer a number of questions which arise in network problems as outlined in Chapter 1. Following on the experience of practical applications during the development of the model, special facilities have been incorporated which enhance its usefulness in problem solving in both batch and time sharing mode. To exemplify its use, the model is applied to both distribution and collection type network problems (i.e. water supply and solid waste). Since no computer model used in practical applications is ever complete, recommendations for possible further development of the model are proposed.

The network problems considered in this study exhibit the common property of providing a group of communities with a public utility service or commodity. The package provides a technique whereby the economic advantages and disadvantages of regionalization of such services may be examined quantitatively. Typical network types include water supply, wastewater collection and solid waste collection. The NETSOL package is designed to suit both collection and distribution type network problems. Both types of networks are made up of three major components:

1) Processing nodes - These points supply the public utility service to the community (i.e. water treatment, waste treatment, etc.). Costs are normally assigned to the associated activities and are a function of the quantity processed.

2) Nonprocessing nodes - The consumers or areas which utilize the service provided are represented by these points (i.e. water consumers or communities generating waste to be collected). Thus, demand is assumed to be concentrated at a discrete number of nodal points rather than distributed spatially over an area. The degree of discretization is limited only by the size and complexity of network which the user is prepared to define and analyze and yet will accurately represent the consumer areas. The costs associated with any activity at these points is fixed and normally disregarded in a network analysis (e.g. local collection of solid waste, or the provision of balancing storage for fluctuating water demand).

3) Transportation links - Conveyance of material between nodes is assumed to take place along any number of feasible links connecting the nodes. The flowrates in these links are treated as the primary design variables and costs are generated as a function of these flowrate values.

The processing rates, upon which processing costs are dependent, are derived from the conveyance rates through nodal continuity relationships. The system costs comprise the transportation and processing costs which are separable functions of the flowrates. Due to economies of scale found in network components, the cost functions are generally concave and monotonic.

A general and much used mathematical statement formulation is adopted for the NETSOL package permitting a wider choice of optimization algorithms that can be incorporated in the model. The mathematical statement comprises the minimization of the objective function (i.e. total system costs) subject to the linear constraints formed by continuity equations at the nodes. Other mathematical statements which have been developed for transportation algorithms are investigated but are not adopted due to the fewer available solution algorithms.

A two step approach is taken in developing the NETSOL package for the solution of network problems. Firstly, a heuristic approach is utilized allowing the engineer to combine intuitive and common sense approaches to problem solving and facilitating the use of engineering judgment in the determination of the final solution. Secondly, the model, through the inclusion of an optimization routine, is given the capacity to solve accurately and rapidly, a large number of alternatives from which the optimal solution may be selected. The program package is designed to enable the user to easily set up a model with the minimum of development and yet allow for the inclusion of complex design functions. The package also facilitates the swapping of optimization routines so that an accurate and efficient algorithm may be adopted. A number of existing solution algorithms are investigated for possible use in the package by comparing the advantages and disadvantages of each (Chapter 4). The algorithms are found to share one or more of the following disadvantages:

(1) The algorithm is difficult to apply (i.e. some algorithms require explicit functions which are often difficult to attain with design oriented subroutines).

(2) Excessive storage is required for application.

(3) The algorithm is computationally expensive.

(4) The algorithm does not guarantee a global minimum.

(5) The methods frequently fail to take advantage of special properties of the solution (see below).

From all of the routines investigated, the iterative linear programming (ILP) algorithm is found to be the most useful. Through the use of the SIMPLEX linear programming algorithm, the ILP routine requires a relatively small amount of storage, is computationally efficient and is very stable. The ILP routine is easily applied in NETSOL since both are derived from a common mathematical statement. The main drawback to the ILP routine is that, although found to be very useful in practice, it occasionally converges to a local minimum, especially in small networks which generally exhibit relatively fewer degrees of freedom.

The difficulties encountered in choosing an optimization routine supports the need for a heuristic approach to problem solving. To facilitate the input of engineering judgment, an iteractive command structure is set up permitting the user to make alterations to the initial flow assumptions, obtain solution costs or modify the network constituents (i.e. links, nodes, states and stipulations). A modular approach is taken in constructing the package to not only facilitate the swapping of optimization routines, but allow for the inclusion of complex design functions with as little modification as possible to the package itself.

In view of the disadvantages of the optimization routines investigated, the properties of the network problems are investigated to determine if a new algorithm can be developed to take advantage of any special characteristics and attain improved convergence characteristics. The main properties of the network problems include the existence of an objective function made up of the sum of separable concave functions, linear constraints and a sparse structural matrix. It is shown in Chapter 2 that, as a direct result of these properties, the minimum solution to a network problem is a basic solution and that these basic solutions lie at the vertices of the feasible solution set formed by the constraints. A solution algorithm, HYVRST, is then developed in Chapter 5 to take advantage of this important property by using a direct search technique which is limited to the vertices. This approach leads to a very stable algorithm with good convergence properties and which can be computationally efficient depending upon the complexity of the functions used in evaluating the objective function at the vertices. The algorithm is developed through a modification of the linear SIMPLEX technique, taking advantage of the economical matrix manipulation methods used. The utilization of the objective function to direct the search technique in HYVRST is, however, quite different from SIMPLEX. The decision is based not simply on local gradients (i.e. cost coefficients) as in SIMPLEX but on the total cost at adjacent vertices. Another major difference found to exist between HYVRST and SIMPLEX is that the HYVRST technique can be sensitive to null nodes in a network and SIMPLEX is not. A perturbation technique is built into the package to obviate any difficulties of this type.

To avoid convergence to a local minimum, an extended search capability is built into the HYVRST routine to ensure that the search for an alternative and possibly preferable minimum is continued following convergence to the first apparent minimum. Thus, the probability of a computed minimum being global may be increased by the additional expenditure of computer time (and cost) as desired by the user.

Testing shows that the HYVRST routine has good convergence properties but is computationally more expensive than the ILP routine. Substantial savings can be made by starting the HYVRST search from an initial solution that appears relatively close to the optimum. The ILP routine has good convergence properties in determining the most economical overall or regional policy, that is, finding which processing nodes should be used to service each customer or group of customer nodes. The HYVRST routine is best suited to discretizing the network policy, that is, choosing among the feasible links for the most economical conveyance policy. It appears therefore that a good approach to use is to utilize, initially, the ILP algorithm to find an optimal solution which, in practice, has been found to be suitable, especially in a preliminary analysis. If a higher degree of confidence in the accuracy of the solution is required, the solution found by the ILP routine can be used as a starting solution for HYVRST, thereby saving a substantial amount of computational effort over that required when no starting solution is specified.

To illustrate the application of the NETSOL package and further test the ILP and HYVRST algorithms, two sample problems are presented in Chapter 6. The first network analyzed is a water supply network and the second, a solid waste collection network. Both examples follow through the typical steps used in applying NETSOL to a network problem:

(1) Identify the problem, set the boundaries of the regional network and define the questions to be answered by the model.

(2) Set up the network to be solved by assigning nodal points representing the processing centres and consumer zones. Include the transportation links which are technically feasible.

(3) Prepare the data file to represent the system as defined by the requirements shown in the program users guide in Appendix E.

(4) Generate cost functions to define the processing and transportation costs and check that these in fact display economies of scale with respect to the flowrate.

(5) Set up the econometric model and augment subroutines CNSTIN and REPORT to input any ad hoc parameters and output any required design variables. Include the required optimization routine or set of routines.

(6) Utilize the model in answering the original questions on the network.

In applying the NETSOL package to the network problems, it is found to be very useful in their solution. The interactive capabilities of the model are valuable in answering the different questions which arise in network problems and testing the optimality of the final solution. To facilitate application of the model, a users guide is included in Appendix E, as mentioned above, which describes the package and outlines its use. This package is also soon to be included and documented in an extensive civil engineering program library, CEPL (34).

Although the NETSOL package is very capable of solving network problems as it stands, use of the package reveals further facilities or changes which would be useful to incorporate. It may be beneficial to undertake some of the changes listed as follows:

(1) Include "SAVE" and "READ" commands in subroutine QINIT to enable the user to output the values of the design variables (flowrates) associated with a solution found by the model into a retrievable storage location. The stored flowrate values are then available for input during subsequent runs through the use of the "READ" command and can be used to define an initial solution.

(2) Incorporate an "echo-printing" option in the package to facilitate batch usage. This can be used to ensure that the commands and auxiliary data employed in obtaining a solution are explicitly described with the output of results.

(3) Include a "CONTINUITY" command in subroutine QINIT which, upon specification, would output a summary of the nodal inflows or outflows and stipulations. The user would then be given the option of establishing continuity through the slack variables if continuity does not exist. This option would aid the user in maintaining a continuous solution if required by the optimization routine.

(4) In determining a solution cost, the HYVRST algorithm utilizes the user-defined cost routines to calculate the costs associated with every nonzero flowrate and then totals them. In large networks (i.e. 40 nodes, 60 links), it has been found that, during the search for the optimal solution, a significant number of the nonzero flowrates do not change in value from iteration to iteration. A saving in execution time may therefore be realized if a check is made to determine if the recalculation of each cost term is required when determining the cost for a new solution. In smaller networks (i.e. 15 nodes, 20 links), generally all of the flowrates change from iteration to iteration and a method such as this would not save any execution time. Furthermore, with small networks, the addition

of a method such as this would increase the execution time due to the computational effort required to compare the flowrates. This alteration would therefore only decrease execution times when applied to larger problems where execution times can be quite long.

(5) One of the disadvantages of the HYVRST algorithm over the ILP routine is the additional storage required in its application. This limits the problem size to which the algorithm can be applied, with the size depending greatly upon the storage capabilities of the computer system used (i.e storage problems have not been encountered on a CDC 7600; however, the water supply network exemplified in Chapter 6 was the largest that would fit into the intercom compiler of the CDC 6400 at the time of application). If the model is utilized in a system where the storage capabilities will limit the possible network size severely, storage savings techniques, such as transforming structural elements to a 2-bit binary form as discussed in section 2.5 (page 45) should be implemented.

(6) In subroutine DEFINE, the mathematical statement is set up in a slightly different manner for the two network types - collection and distribution. In a collection network, any stipulation associated with a processing node dictates that material to be collected is being generated at that node. This has a distinct advantage in that it reduces the number of nodal points required in the network if processing centres are located in areas which also generate material to be processed. This is generally found to occur in collection networks such as a solid waste network. If the mathematical statements were identical for both collection and distribution networks, the stipulation of a processing node then dictates the upper limit on processing at that node. This is advantageous in that it provides an easy method of limiting processing capacities if required. It is felt that the present mathematical statement form is more advantageous; however, it may be necessary to change the formulation if it is found that numerous problems arise where upper limits on processing capacities are very important to the problem being analyzed. It should also be noted that the incorporation of upper limits to processing

capacity in the cost function (e.g. by means of penalty terms) will generally result in the requirement of concave functions being violated.

(7) It would be beneficial to develop a plotting subroutine to obtain a line plot of a schematic representation of the initial network and/or final solution. The plot could be produced either on a line printer (along with the output of results) or on auxiliary computer hardware designed especially for plotting. A plot facility would be helpful in detecting data errors which are not immediately obvious through checking the data and obtaining a clearer conception of the final solution. This facility would be most beneficial in larger networks.

(8) A specific example is presented in Chapter 6 which involves the optimization of a regional water supply system. The conveyance cost functions are constructed so that a pump-pipe configuration can exist in any transportation link if it proves to be more economical than a pipe by itself. The results show that if a relatively large number of pumps are included in the final system, it may be more economical to consolidate some of the pumps due to the economies of scale realized with

larger pump installations. A technique to use in the consolidation of pumps is described in the chapter. It would be useful to code this technique for use in the package in conjunction with the optimization routine when applied to water supply networks.

(9) Chapter 6 presents the application of the NETSOL package to both a regional water supply and solid waste collection problem. No attempt is made to apply the package to another common network system, wastewater collection, since it is basically similar to solid waste collection. An added complication in the application to wastewater collection, however, is encountered in the transportation cost functions through the interaction of the pipe invert levels and ground levels, and the fact that the total system cost is very sensitive to the "state" variable at the nodes. Application to a regional wastewater network would be beneficial to identify any specialized problems encountered in this type of system.

The above list outlines changes which could be made to the NETSOL package. As the package is further applied to network problems, more changes may become apparent. The package as it exists, however, is fully capable of solving network problems.

The NETSOL package fulfills the main objective of this thesis as outlined in Chapter 1, that is, to develop a package of subroutines which can be used to construct econometric models useful in solving network problems. In developing the package, a heuristic approach is taken to facilitate the input of engineering common sense to the problem solving; and yet, a theoretical optimization routine is included to obtain accurate results. These properties of the package are laid out as auxiliary objectives in Chapter 4 and are also discussed earlier in this section. The suitability of the package to the solution of network problems and its ability to remain consistent with the demands of professional engineering practice is ensured by the application of NETSOL to two real life problems (35, 37). Both applications involve the analysis of water supply networks, one for St. John's, Newfoundland and the other for Dodoma, Tanzania. The undertaking of this thesis and the studies performed prompted the publication of a paper (36) describing the NETSOL package and its use. The St. John's study is a very ad hoc application and does not utilize the NETSOL package as presented in this thesis. In developing the present NETSOL package, the following improvements are made:

(1) Complete redevelopment and modularization of the program.

(2) Generalization of NETSOL to include collection as well as distribution networks within the scope

of the package.

(3) The adoption of a heuristic approach to facilitate the inclusion of an intuitive and common sense approach to problem solving.

(4) An in-depth study of the properties of the network problem and its solution which leads to the development of an accurate and stable optimization routine specially designed for network problems.

The improvements made to the NETSOL package make it easier to apply and increase its usefulness in the solution of network problems. Hopefully, this thesis has provided some insight into the properties of networks and has shown how useful the NETSOL package can be.

# BIBLIOGRAPHY

1. Adams, B.J. A Water Quality Evaluation of Regional Wastewater Management. Urban and Environmental Systems, Report No. UES 74-5, McGill University, March 1974.

2. Adams, B.J. Economics of Regional Wastewater Management. Urban and Environmental Systems, Report No. UES 74-9, McGill University, June 1974.

3. Aguilar, R.J. Systems Analysis and Design. Prentice Hall, 1973.

4. American Public Works Association. Rail Transport of Solid Wastes. APWA, August 1971.

5. Baum, Dr. B. and Parker, C.H. Solid Waste Disposal, Incineration and Landfill (Volume 1). Ann Arbor Science Publishers Inc., 1973.

6. Baum, Dr. B. and Parker, C.H. Solid Waste Disposal, Reuse/Recycle and Pyrolysis (Volume 2). Ann Arbor Science Publishers Inc., 1973.

7. Converse, A.O. Optimum Number and Location of Treatment Plants. Journal Water Pollution Control Federation, Vol. 44, No. 8, pages 1629-1636, August 1972.

8. Cooper, L. Applied Nonlinear Programming for Engineers and Scientists. Aloray, 1974.

9. Dantzig, G.B. Linear Programming and Extensions. Princeton University Press, 1963.

10. Fair, G.M., Geyer, J.C. and Okun, D.A. Water Purification and Wastewater Treatment and Disposal. John Wiley and Sons, 1968.

11. Feldman, E., Lehrer, F.A. and Ray, T.L. Warehouse Location Under Continuous Economies of Scale. Journal of Management Science. Volume 12, No. 9, pages 670-684, May 1966.

12. Fiacco, A.V. and McCormick, G.P. Nonlinear Programming: Sequential Unconstrained Minimization Techniques. John Wiley and Sons, 1968.

13. Gass, S.I. Linear Programming Methods and Applications. 3rd edition, McGraw-Hill Book Co., 1969.

14. Graves, R.L. and Wolfe, P. Recent Advances in Mathematical Programming. McGraw-Hill Book Co., 1963.

15. Griffith, R.E. and Stewart, R.A. A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems. Management Science, Vol. 17, pages 379-392, 1961.

16. Gupta, J.M., Agarwal, S.K. and Khana, P. Optimal Design of Wastewater Collection Systems. Journal of the Environmental Engineering Division, Vol. 102, No. EES, pages 1029-1041, October 1976.

17. Hagerty, D.J., Pavoni, J.L. and Heer, J.E. Jr. Solid Waste Management. Van Nostrand Reinhold Co., 1973.

18. Haggett, P. and Chorley, R.J. Network Analysis in Geography. Edward Arnold, 1969.

19. Himmelblau, D.M. Applied Nonlinear Programming. McGraw-Hill Book Co., 1972.

20. Hitchcock, F.L. The Distribution of a Product From Several Sources to Numerous Localities. Journal Mathematical Physics, Volume 20, pages 224-230, 1941.

21. Kantorovich, L.V. Mathematical Methods in the Organization and Planning of Production. Management Science, Vol. 6, pages 366-422, 1960.

22. Korol, Dr. R.M. Incineration Costs. Private communication from Dr. R.M. Korol, Department of Civil Engineering and Engineering Mechanics, McMaster University, Hamilton, Ontario.

23. Kuehn, A.A. and Hamburger, M.J. A Heuristic Program for Locating Warehouses. Management Science, Vol. 9, pages 643-666, July 1963.

24. Kuester, J.L. and Mize, J.H. Optimization Techniques With Fortran. McGraw Hill Book Co., 1973.

25. Lasdon, L.S. Optimization Theory for Large Systems. The MacMillan Company, 1970.

26. Loucks, D.P., Revelle, C.S. and Lynn, W.R. Linear Programming Models for Water Pollution Control. Management Science, Vol. 14, No. 4, pages B 166-B 181, December 1967.

27. National Center for Resource Recovery Inc. Incineration. Lexington Books, 1974.

28. National Center for Resource Recovery Inc. Municipal Solid Waste Collection. Lexington Books, 1973.

29. National Center for Resource Recovery Inc. Sanitary Landfill. Lexington Books, 1974.

30. Ontario Water Resources Commission. Sewage Treatment Plant Construction Costs. OWRC, Design Approvals Branch, Division of Sanitary Engineering, Publication #1, 1963.

31. Proctor and Redfern Ltd. Hamilton-Wentworth Regional Solid Waste Study. Report submitted to the Regional Municipality of Hamilton-Wentworth by Proctor and Redfern Limited of Canada, 75 Eglinton Ave. E., Toronto, Ontario, 1975.

32. Rosen, J.B. The Gradient Projection Method for Nonlinear Programming. Journal Society Industrial Applied Mathematics, Vol. 8, No. 1, pages 181-217, March 1960.

33. Siddal, J.N. OPTISEP, Designers' Optimization Subroutines. ME/71/DSN/REPI, Faculty of Engineering, McMaster University, Hamilton, Ontario, 1971.

34. Smith, A.A. C.E.P.L.: A Civil Engineering Program Library. McMaster University Bookstore, McMaster University, Hamilton, Ontario, 1974.

35. Smith, A.A. Economic Optimization of a Water Supply System for the New Capital City of Dodoma, Tanzania. Report to Project Planning Associates Limited, Toronto, Ontario, 1976.

36. Smith, A.A. and Tufgar, R.H. Planning Regional Network Systems for Minimum Cost. Canadian Journal of Civil Engineering, Vol. 4, No. 2, pages 202-213, June 1977.

37. Smith, A.A. Systems Economics. In St. John's Regional Water System Study. Report submitted to the Canadian Department of Regional and Economic Expansion and to the Government of Newfoundland and Labrador by Foundation Engineering Company of Canada, 1 Yonge St., Toronto, Ontario, 1974.

38. Summer, W.K. and Spiegel, Z. Ground Water Pollution (A Bibliography). Ann Arbor Science Publishers, 1974.

39. Wanielista, M.P. and Bauer, C.S. Centralization of Waste Treatment Facilities. Journal of Water Pollution Control Federation, Vol. 44, No. 12, pages 2229-2238, December 1972.

40. Wilde, D.J. and Beightler, C.S. Foundations of Optimization. Prentice Hall, 1967.

41. Williamson, W. Aspects of Waste Disposal. Ontario Ministry of the Environment, November 1972.

# APPENDIX A

## PROPERTIES RELATED TO LINEAR CONSTRAINTS

This appendix contains the proofs of two theorems related to the linear constraints of a network problem.

Theorem 1: The feasible space defined by a set of linear constraints is a convex set.

Proof:

Let $\bar{Q}^{F1}$ and $\bar{Q}^{F2}$ represent any two feasible solutions, or design variable vectors, the coordinates of which are:

$$Q_j^{F1} \; , \; j = 1, \; NQ \qquad (A.1)$$

and $Q_j^{F2}$ , $j = 1$, NQ respectively

where: $Q_j^{F1}$ = the $j^{th}$ flow variable for solution 1

NQ = the number of flow variables

Upon substitution into the constraint equations:

$$\sum_{j=1}^{NQ} a_{ij} \, Q_j^{F1} = b_i \qquad \text{for } i = 1, \; N \qquad (A.2)$$

$$\sum_{j=1}^{NQ} a_{ij} Q_j^{F2} = b_i \qquad \text{for } i = 1, N \qquad (A.3)$$

where: $a_{ij}$ = the coefficient from the $i^{th}$ row and $j^{th}$ column of the structural matrix

$b_i$ = the $i^{th}$ nodal stipulation

$N$ = the number of nodes, or constraints

Multiply equation A.2 by $\alpha$ and A.3 by $1-\alpha$ where $0<\alpha<1$:

$$(\alpha) \sum_{j=1}^{NQ} a_{ij} Q_j^{F1} = (\alpha)\ b_i \qquad (A.4)$$

for $i = 1, N$

$$(1-\alpha) \sum_{j=1}^{NQ} a_{ij} Q_j^{F2} = (1-\alpha)\ b_i \qquad (A.5)$$

Add A.4 to A.5 to obtain:

$$\sum_{j=1}^{NQ} a_{ij} (\alpha\, Q_j^{F1} + (1-\alpha)\, Q_j^{F2}) = b_i \qquad (A.6)$$

Let $\quad Q_j = \alpha\, Q_j^{F1} + (1-\alpha)\, Q_j^{F2} \qquad j = 1, NQ$

or, equivalently $\qquad \bar{Q} = \alpha\, \bar{Q}^{F1} + (1-\alpha)\, \bar{Q}^{F2}$

The solution $\bar{Q}$ satisfies the constraints and constitutes a feasible solution. Since $\bar{Q}$ represents any point on a segment joining $\bar{Q}^{F1}$ and $\bar{Q}^{F2}$ and the above applies to any two feasible solutions, then the collection of feasible solutions constitutes a convex set.

Theorem 2: The vertices of a convex set formed by a set of linear constraints constitute basic feasible solutions of that set of constraints.

The vertices of a convex set are formed by the intersection of the hyperplanes representing the constraint equations in a hyperspace. This proof demonstrates that the vertices are basic solutions where a basic solution is defined as a feasible solution with the number of non-zero variables in the solution equal to the number of constraint equations.

Proof:

Let $\bar{Q}^B$ represent a basic feasible solution whose non-zero variables are:

$$Q_j^B \quad , \quad j = 1, N \tag{A.7}$$

The variables are renumbered so that the first N variables make up the basic.

Therefore: $$\sum_{j=1}^{N} a_{ij} Q_j^B = b_i \qquad i = 1, N \tag{A.8}$$

This solution is unique according to the definition of a basic solution (that is, only N variables may be solved for, using a system of N independent equations).

Assume that $\bar{Q}^B$ lies on a segment joining two feasible solutions $\bar{Q}^{F1}$ and $\bar{Q}^{F2}$.

Therefore: $$Q_j^B = \alpha\, Q_j^{F1} + (1-\alpha)\, Q_j^{F2} \qquad j = 1, N \qquad \text{(A.9)}$$

$$0 = \alpha\, Q_j^{F1} + (1-\alpha)\, Q_j^{F2} \qquad j = N+1, NQ$$

for any $\alpha$ in the range $0<\alpha<1$

Therefore: $$Q_j^{F1} = Q_j^{F2} = 0 \qquad j = N+1, NQ \qquad \text{(A.10)}$$

And since the above basic solution is unique:

$$Q_j^{F1} = Q_j^{F2} = Q_j{}^B \qquad j = 1, NQ \qquad \text{(A.11)}$$

Therefore, by contradiction, $\bar{Q}^B$ cannot lie on a segment joining two feasible solutions if it is unique and the basic feasible solution must lie on the vertices of a convex set.

The concept behind Theorem 2 is better understood when illustrated through the use of an example. A two variable network problem is defined by the following mathematical statement:

$$\underset{\bar{Q}^*}{\text{Minimize}}\ Z = \Sigma\, C(Q_i) \qquad i = 1, 2$$

$$\text{subject to} \quad Q_1 < b_1$$

$$Q_2 < b_2$$

$$Q_1+Q_2 > b_3$$

The constraints can be restated in the following equation form:

$$\begin{aligned} Q_1 \quad +Q_3 \qquad\quad &= b_1 \\ Q_2 \quad +Q_4 \quad &= b_2 \\ Q_1+Q_2 \qquad\quad -Q_5 &= b_3 \end{aligned}$$

The constraint equations are plotted on a graph of $Q_1$ vs. $Q_2$ in figure A.1, forming the convex feasible solution space. At vertex number 1, $Q_1$ and $Q_5$ are equal to zero with $Q_2$, $Q_3$ and $Q_4$ being the nonzero variables representing a basic solution. At vertex number 2, $Q_3$ and $Q_5$ are zero with $Q_1$, $Q_2$ and $Q_4$ being the nonzero variables in the basic. Along the line joining the vertices 1 and 2 only $Q_5$ is zero and the solutions formed by nonzero $Q_1$, $Q_2$, $Q_3$ and $Q_4$ are non-basic.

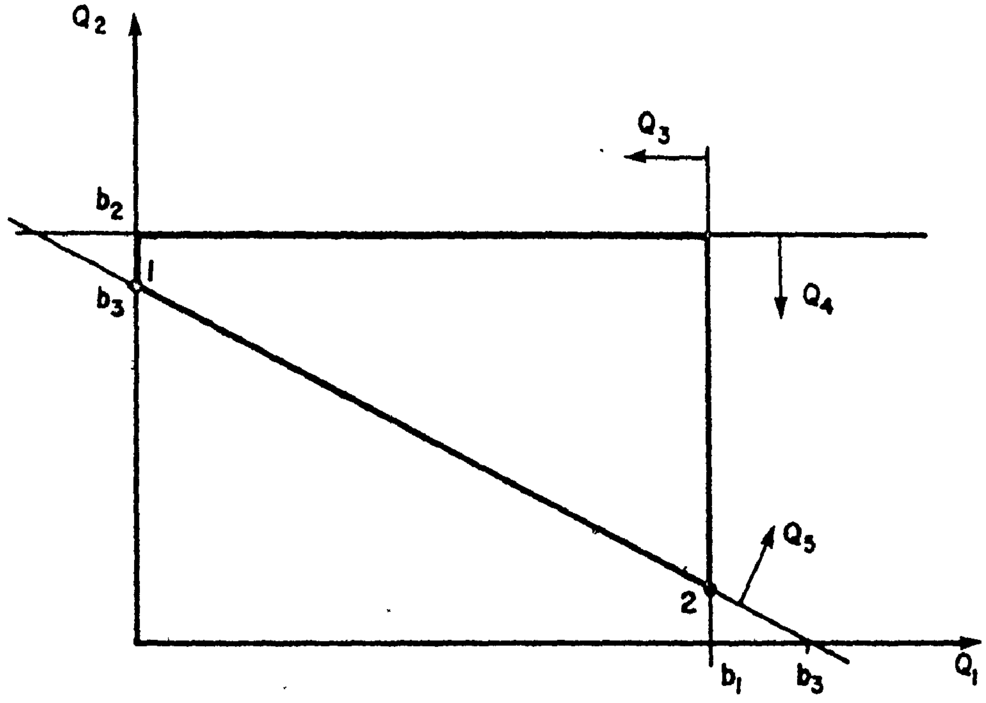


FIGURE A-1 — FEASIBLE SOLUTION SPACE FOR A TWO VARIABLE PROBLEM

# APPENDIX B

## OPTIMAL SOLUTION PROPERTIES

Theorem: Given an objective function comprising the sum of concave functions of single variables, which is to be minimized subject to a set of linear constraints, the solution must be a basic feasible solution of these constraints (i.e. must lie on a vertex of the feasible space defined by the constraint set).

Proof:

If C(Q) represents the concave cost function then C(Q) is concave if and only if the following is true:

$$C(\alpha\ Q_1 + (1-\alpha)\ Q_2) \geq \alpha\ C(Q_1) + (1-\alpha)\ C(Q_2) \qquad \text{(B.1)}$$

where: $0 < \alpha < 1$

and $Q_1$ , $Q_2$ represent two flowrate values for the function C(Q).

The physical significance of this is illustrated through the use of the graph of the function C(Q), shown in figure B.1. The costs given by the term $\alpha\ C(Q_1) + (1-\alpha)\ C(Q_2)$, $0 < \alpha < 1$, form a linear chord approximation to the real function curve with the end points

$(Q_1, C(Q_1))$ and $(Q_2, C(Q_2))$. It is clearly seen that the true cost curve lies above the chord approximation for all values of $\alpha$.

The statement of concavity (B.1) given above can be extended for the concave objective function of a network problem formed by the summation of the concave functions of single variables. Let $\bar{C}(\bar{Q})$ represent the objective function. Therefore, $\bar{C}(\bar{Q})$ is concave if and only if the following is true for a set of P feasible solutions:

$$\bar{C}\left(\sum_{i=1}^{P} \alpha_i \bar{Q}^{Fi}\right) \geq \sum_{i=1}^{P} \left(\alpha_i \bar{C}(\bar{Q}^{Fi})\right) \qquad \text{(B.2)}$$

where: $\alpha_i \geq 0$ and $\sum_{i=1}^{P} \alpha_i = 1$

$\bar{Q}^{Fi}$ is any feasible solution

Let R denote the feasible region, or convex set. Let $\bar{Q}^{B1}$, $\bar{Q}^{B2}$, ... $\bar{Q}^{BP}$ denote the extreme points of R (basic feasible solutions). Also, let $\bar{Q}^{o}$ denote the optimal solution (a minimum).

Therefore:

$$\bar{C}(\bar{Q}^{o}) \leq \bar{C}(\bar{Q}^{Bi}) \quad \text{all } \bar{Q}^{Bi} \in R,\ i = 1, P \qquad \text{(B.3)}$$

If $\bar{Q}^{o}$ is not an extreme point, it can be expressed as a linear combination of the extreme points:

i.e. $$\bar{Q}^{o} = \sum_{i=1}^{P} (\alpha_i \bar{Q}^{Bi}) \text{ where: } \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{P} \alpha_i = 1 \tag{B.4}$$

Therefore:

$$\bar{C}(\bar{Q}^{o}) = \bar{C}( \sum_{i=1}^{P} (\alpha_i \bar{Q}^{Bi})) \tag{B.5}$$

From the previous definition of a concave function (B.2):

$$\bar{C}(\bar{Q}^{o}) \geq \sum_{i=1}^{P} (\alpha_i \bar{C}(\bar{Q}^{Bi})) \tag{B.6}$$

Let $$\bar{C}(\bar{Q}^{BM}) = \min_{i=1}^{P} \bar{C}(\bar{Q}^{Bi}) \tag{B.7}$$

and since

$$\alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{P} \alpha_i = 1$$

then: $$\sum_{i=1}^{P} \alpha_i \bar{C}(\bar{Q}^{Bi}) \geq \bar{C}(\bar{Q}^{BM}) \tag{B.8}$$

Therefore:

$$\bar{C}(\bar{Q}^{o}) \geq \bar{C}(\bar{Q}^{BM}) \tag{B.9}$$

Thus, either $\bar{Q}^{o}$ is an extreme point or there is an extreme point at which the objective function assumes its minimum value. Then, by contradiction, the optimal point lies on a vertex of the convex set.

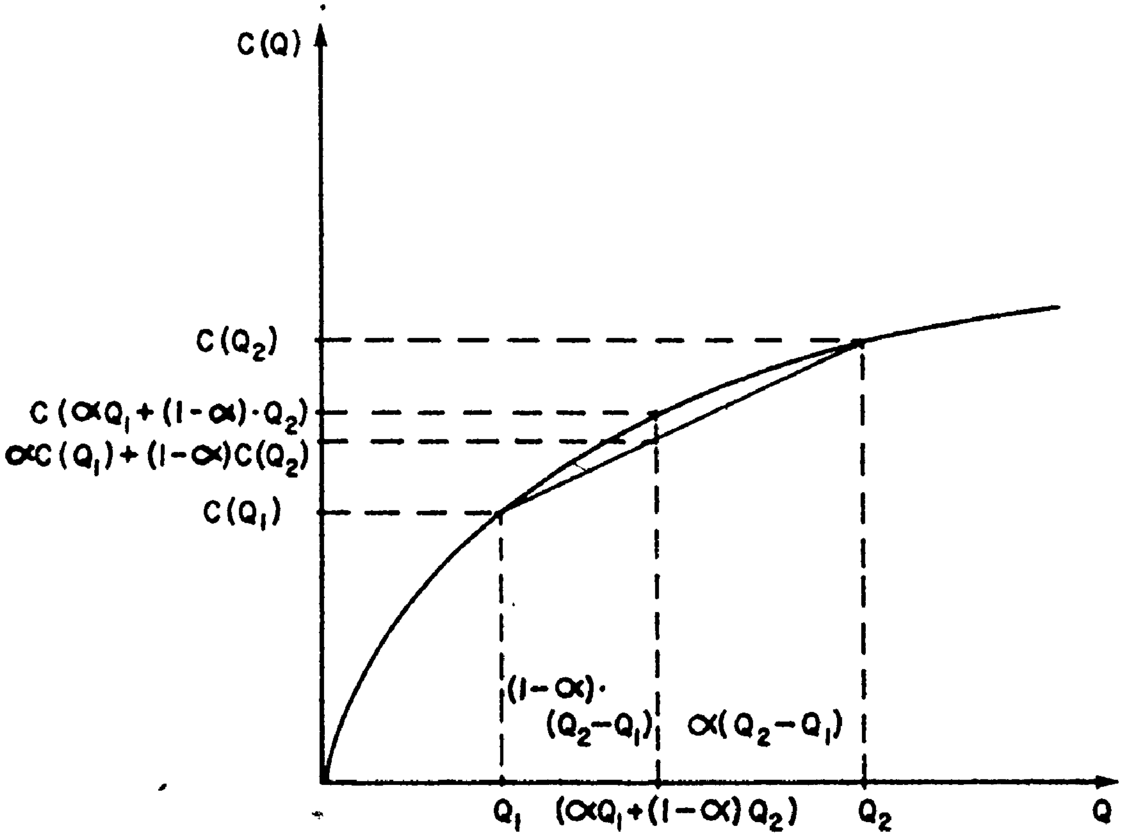

$C(Q)$
$C(Q_2)$
$C(\alpha Q_1 + (1-\alpha)\cdot Q_2)$
$\alpha C(Q_1) + (1-\alpha)C(Q_2)$
$C(Q_1)$
$(1-\alpha)\cdot$
$(Q_2 - Q_1)$
$\alpha(Q_2 - Q_1)$
$Q_1$
$(\alpha Q_1 + (1-\alpha)Q_2)$
$Q_2$
$Q$

FIGURE B·1 — CONCAVE COST CURVE

# APPENDIX C

## LISTING OF THE NETSOL PACKAGE

### Subroutines included

DRIVER PROGRAM

NETSOL

CNSTIN

DEFINE

QINIT

MODIFY

SOLCST

LNKCST

COST1

CENCST

COST2

SOLVE

REPLCQ

REPORT

```
C
C     MAIN PROGRAM
C     DRIVER PROGRAM FOR NETWORK OPTIMIZATION.
C     READS PROBLEM SIZE FROM TAPE NR1.
C     PRINTS OUT THE PROGRAM TITLE.
C      DIMENSIONED ACCORDING TO PROBLEM SIZE.
C      N    = THE TOTAL NUMBER OF NODES.
C      NCEN = THE NUMBER OF PROCESSING NODES.
C      L    = NUMBER OF LINKS.
C      NQ   = THE NUMBER OF FLOW PLUS SLACK VARIABLES.
C             (2*L+N)
C
C      ARRAY SIZE SPECIFICATION.....
C      (N,NQ)..A
C      (5,N)...NAMES
C      (NQ)....C,CPEN,NDS,NUS,Q,XL,
C      (N).....B,STATE,STIP
C
C     ARRAYS STANDARD TO NETWORK PACKAGE.
C
      DIMENSION A(13,55),B(13),C(55),CPEN(55),NAMES(5,13)
      DIMENSION NDS(55),NUS(55),Q(55),STATE(13),STIP(13),XL(55)
C
C     WORKING ARRAYS FOR OPTIMIZATION PACKAGE.
C
      DIMENSION WA(13,13),WB(13),WC(13),WD(13)
      DIMENSION WE(13),WF(13),WG(55)
      INTEGER WF
C
C     DEFINE INPUT AND OUTPUT FILE NUMBERS.
C      NR1 = DATA FILE
C      NR2 = KEYBOARD INPUT DEVICE
C      NW  = OUTPUT DEVICE
C
      NR1=1
      NR2=5
      NW=6
      REWIND NR1
      READ(NR1,10)N,NCEN,L
 10   FORMAT(3I5)
      NQ=2*L+N
      WRITE(NW,20)
 20   FORMAT(///10X,' NETWORK SYSTEM OPTIMIZATION PROGRAM'/
     +       10X,' -----------------------------------'/
     +20X,' BY  A A SMITH'/
     +20X,' AND R H TUFGAR'////)
      CALL NETSOL(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,STIP,XL,
     +            WA,WB,WC,WD,WE,WF,WG,L,N,NCEN,NQ,NR1,NR2,NW)
      STOP
      END
      SUBROUTINE NETSOL(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,STIP,XL,
     +                  WA,WB,WC,WD,WE,WF,WG,L,N,NCEN,NQ,NR1,NR2,NW)
C
C   **************************************************************
C     ORGANIZATIONAL ROUTINE WHICH CALLS THE MODULAR ROUTINES
C     IN THE PROPER SEQUENCE FOR THE SOLUTION OF NETWORK PROBLEMS.
C     SETS UP DYNAMIC DIMENSIONING.
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C               DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NAMES   = TWO-DIMENSIONAL ARRAY CONTAINING THE NAMES
C               ASSIGNED TO THE NODAL POINTS.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     STIP    = ONE-DIMENSIONAL ARRAY CONTAINING THE
C               STIPULATIONS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     WA-WG   = WORKING ARRAYS FOR THE OPTIMIZATION ALGORITHMS.
C     L       = NUMBER OF LINKS IN THE NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C     NR1     = PERIPH. DEVICE NUMBER FOR DATA FILE INPUT.
C     NR2     = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C
C     USES ROUTINES DEFINE CNSTIN,QINIT,MODIFY,SOLVE,REPORT
C                    SOLCST,LNKCST,CENCST.
C     USER DEFINED ROUTINES COST1,COST2.
C   **************************************************************
C
C     ARRAYS STANDARD TO NETWORK PACKAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NAMES(5,N)
      DIMENSION NDS(NQ),NUS(NQ),Q(NQ),STATE(N),STIP(N),XL(NQ)
C
C     WORKING ARRAYS FOR OPTIMIZATION PACKAGE.
C
      DIMENSION WA(N,N),WB(N),WC(N),WD(N)
      DIMENSION WE(N),WF(N),WG(NQ)
      INTEGER WF
```

```
C
      IFLAG=0
      NMOD=1
 10   CONTINUE
      CALL CNSTIN(NR2,NW,QLIMIT)
      IF(NMOD.EQ.2)GO TO 5
      CALL DEFINE(A,B,CPEN,NAMES,NDS,NUS,Q,STATE,STIP,XL,ITYPE,
     +            L,N,NCEN,NQ,NR1,NW,QLIMIT)
  5   CONTINUE
      CALL QINIT(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,XL,
     +           ITYPE,N,NCEN,NQ,NR2,NW,QLIMIT)
      CALL MODIFY(A,B,CPEN,NDS,NUS,STATE,STIP,XL,
     +            ITYPE,N,NCEN,NENTRY,NQ,NR2,NW,QLIMIT)
      IF(NENTRY.EQ.1)GO TO 5
      NMOD=2
      CALL SOLVE(A,B,C,CPEN,NDS,NUS,Q,STATE,XL,WA,WB,WC,
     +           WD,WE,WF,WG,ITYPE,N,NCEN,NQ,NR2,NW,QLIMIT)
      IF(IFLAG.EQ.1)GO TO 10
      CALL REPORT(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,
     +            XL,ITYPE,N,NCEN,NQ,NW,QLIMIT)
      GO TO (10,10,20),NMOD
 20   RETURN
      END
      SUBROUTINE CNSTIN(NR2,NW,QLIMIT)
C
C   **************************************************************
C     USER AUGMENTED SUBROUTINE.
C     THIS SUBROUTINE ENABLES THE USER TO INPUT ANY CONSTANTS
C     REQUIRED IN THE COST SUBROUTINE.
C
C     NR2    = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW     = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT = LIMITING FLOWRATE VALUE. VALUES LESS THAN
C              QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C   **************************************************************
C
C     INSERT ANY LABELLED COMMON BLOCK STATEMENTS HERE.....
C
      QLIMIT=0.001
      DATA YES/1HY/
C
C     START OF USER SUPPLIED CONSTANT EVALUATION.....
C       USER SUPPLIES CODING HERE TO DEFINE ANY CONSTANTS.
C     END OF USER SUPPLIED CONSTANT EVALUATON.....
C
C     OPTION TO BYPASS REDEFINITION OF ANY CONSTANTS.
C
      WRITE(NW,10)
 10   FORMAT(/' IN SUBROUTINE "CNSTIN" '/
     +         ' DO YOU WISH TO REDEFINE ANY CONSTANTS'/
     +         '                          (YES/NO)...')
      READ(NR2,20)ANS
 20   FORMAT(A1)
      IF(ANS.NE.YES)RETURN
C
C     START OF USER DEFINED CODING TO REDEFINE CONSTANTS.....
C       USER SUPPLIES CODING HERE IF CONSTANTS ARE TO BE CHANGED.
C     END OF USER DEFINED CODING TO REDEFINE CONSTANTS.....
C
      RETURN
      END
      SUBROUTINE DEFINE(A,B,CPEN,NAMES,NDS,NUS,Q,STATE,
     +                  STIP,XL,ITYPE,L,N,NCEN,NQ,NR1,NW,QLIMIT)
C
C    ***************************************************************
C      SUBROUTINE TO READ THE NETWORK DEFINITION FROM A
C      SPECIFIED TAPE NR1. ORGANIZES AND STORES THIS DATA
C      IN THE APPROPRIATE ARRAYS FOR LATER USE.
C
C      A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C                DEFINING THE PROBLEM CONSTRAINTS.
C      B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C                STIPULATIONS. (WORKING ARRAY OF STIP)
C      CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C                FOR THE DESIGN VARIABLES.
C      NAMES   = TWO-DIMENSIONAL ARRAY CONTAINING THE NAMES
C                ASSIGNED TO THE NODAL POINTS.
C      NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C                NODAL NUMBERS OF THE DESIGN VARIABLES.
C      NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C                NODAL NUMBERS OF THE DESIGN VARIABLES.
C      Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C      STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C                OF THE NODAL POINTS.
C      STIP    = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C                STIPULATIONS.
C      XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C                ASSOCIATED WITH EACH DESIGN VARIABLE.
C      ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C                ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                          ITYPE=0 FOR A COLLECTION NETWORK.
C      L       = NUMBER OF LINKS IN THE NETWORK.
C      N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C      NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C      NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C                ARTIFICIAL SLACK VARIABLES.
C      NR1     = PERIPH. DEVICE NUMBER FOR DATA FILE INPUT.
C      NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C      QLIMIT  = LIMITING FLOWRATE VALUE. FLOWRATES LESS THAN
C                QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C    ***************************************************************
C
C      DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),CPEN(NQ),NAMES(5,N),NDS(NQ)
      DIMENSION NUS(NQ),Q(NQ),STATE(N),STIP(N),XL(NQ)
C
C   READ DESCRIPTIONS OF NODES
C   BUT FIRST SET STIPULATIONS TO ZERO
C
      DO 15 I=1,N
        STIP(I)=0.0
 15   CONTINUE
      DO 20 II=1,N
        READ(NR1,25) I,(NAMES(J,I),J=1,5),STATE(I),STIP(I)
 25     FORMAT(I3,3X,5A4,2F10.1)
 20   CONTINUE
C
C  READ DESCRIPTIONS OF LINKS
C
      DO 30 II=1,L
        I2=II*2 - 1
        I2P1=I2+1
        READ(NR1,35)NUS(I2),NDS(I2),XL(I2),XL(I2P1),Q(I2),Q(I2P1)
 35     FORMAT(2I5,4F10.1)
        IF(XL(I2P1).LT.1.0E-10) XL(I2P1)=XL(I2)
      IF(Q(I2).LT.1.0E-10) Q(I2)=0.0
      IF(Q(I2P1).LT.1.0E-10) Q(I2P1)=0.0
        NUS(I2P1)=NDS(I2)
        NDS(I2P1)=NUS(I2)
 30   CONTINUE
C
C  SET NUS(),NDS()=ZERO FOR SLACK VARIABLES
C
      DO 38 I=1,N
        II=I+2*L
        NUS(II)=0
      NDS(II)=0
      XL(II)=0.0
      Q(II)=0.0
 38   CONTINUE
C
C  SET STRUCTURAL COEFFS TO ZERO
C
      DO 40 I=1,N
        DO 40 J=1,NQ
          A(I,J)=0.0
 40   CONTINUE
C
C ASSIGN STRUCTURAL COEFFS FOR ALL FLOW VARIABLES
C  OUTFLOW=+VE;INFLOW=-VE
C
      L2=2*L
      DO 50 J=1,L2
        NUSI=NUS(J)
        NDSI=NDS(J)
        A(NUSI,J)=+1.0
        A(NDSI,J)=-1.0
 50   CONTINUE
C
C   ASSIGN STRUCTURAL COEFFS FOR SLACK AND ARTIFICIAL SLACK
C   VARIABLES AND APPLY PERTURBATION (EPS=0.1*QLIMIT) FOR ZERO
C   STIPULATION.
C    SET FLAG ITYPE TO DISTINGUISH BETWEEN COLLECTION AND
C    DISTRIBUTION NETWORK. (DISTRIBUTION NETWORK HAS NEGATIVE STIP.)
C    ITYPE= 0, FOR COLLECTION NET; 1, FOR DISTRIBUTION NET.
C
      EPS=0.1*QLIMIT
      SIGN=1.0
      ITYPE=0
      DO 55 I=1,N
      IF(STIP(I).LT.-1.0E-10)SIGN=-1.0
 55   CONTINUE
      IF(SIGN.LT.0.0)ITYPE=1
      DO 60 I=1,NCEN
      J=L2+I
      A(I,J)=1.0
      IF(STIP(I).LT.1.0E-10)STIP(I)=EPS
 60   CONTINUE
      NCENP1=NCEN+1
      DO 65 I=NCENP1,N
      J=L2+I
      A(I,J)=SIGN
      IF(ABS(STIP(I)).LT.1.0E-10)STIP(I)=SIGN*EPS
 65   CONTINUE
C
C  ASSIGN CONSISTENT INITIAL VALUES TO SLACK AND ARTIFICIAL SLACK
C  VARIABLES BASED ON FIRST ESTIMATES OF Q()
C
      DO 70 I=1,N
        ISLACK=I+L2
        SUM=0.0
        DO 71 J=1,L2
          SUM=SUM+A(I,J)*Q(J)
 71     CONTINUE
        Q(ISLACK)=A(I,ISLACK)*(STIP(I)-SUM)
 70   CONTINUE
C
C  ASSIGN STIPULATIONS TO WORKING ARRAY
C
      DO 80 I=1,N
        B(I)=STIP(I)
 80   CONTINUE
C
C  ZERO CPEN ARRAY
C
      DO 90 I=1,NQ
 90   CPEN(I)=0.0
      RETURN
      END
      SUBROUTINE QINIT(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,XL,
     +                 ITYPE,N,NCEN,NQ,NR2,NW,QLIMIT)
C
C   *************************************************************
C     CONVERSATIONAL ROUTINE TO FACILITATE CHANGES TO
C     INITIAL FLOWRATES AND TO OBTAIN A TRIAL SOLUTION COST
C     FOR THE SPECIFIED INITIAL FLOWRATES.
C
C     A      = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C              DEFINING THE PROBLEM CONSTRAINTS.
C     B      = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C              STIPULATIONS. (WORKING ARRAY OF STIP)
C     C      = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C              THE DESIGN VARIABLE FLOWRATES.
C     CPEN   = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C              FOR THE DESIGN VARIABLES.
C     NAMES  = TWO-DIMENSIONAL ARRAY CONTAINING THE NAMES
C              ASSIGNED TO THE NODAL POINTS.
C     NDS    = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS    = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q      = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE  = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C              OF THE NODAL POINTS.
C     XL     = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C              ASSOCIATED WITH EACH DESIGN VARIABLE.
C     ITYPE  = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C              ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                        ITYPE=0 FOR A COLLECTION NETWORK.
C     N      = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN   = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ     = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C              ARTIFICIAL SLACK VARIABLES.
C     NR2    = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW     = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT = LIMITING FLOWRATE VALUE. FLOWRATES LESS THAN
C              QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C
C     USES ROUTINES SOLCST, REPORT, LNKCST, CENCST,
C     USER DEFINED ROUTINES COST1, COST2.
C   *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NAMES(5,N)
      DIMENSION NDS(NQ),NUS(NQ),Q(NQ),STATE(N),XL(NQ)
C
C     DEFINE ALLOWABLE COMMANDS.
C
      DATA GO,XLIST,ALTER,SOLVE,STOP/4HGO  ,4HLIST,4HALTE,
     +     4HSOLV,4HSTOP/
      DATA YES/1HY/
C
```

```
      L2=NQ-N
      EPS=0.1*QLIMIT
      WRITE(NW,200)EPS
C
C     OUTPUT COMMAND OPTIONS.
C
 200  FORMAT(' IN ROUTINE "QINIT" '/
     +' PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS'/
     +' TO +/-',F8.5,' . ZERO STIPULATIONS ARE NOT ADVISABLE.'/
     +' ALLOWABLE COMMANDS ARE:'/
     +         ' GO.....TO PROCEED',/,
     +         ' LIST...TO LIST NONZERO FLOWRATES',/,
     +         ' ALTER..TO CHANGE INITIAL FLOWRATES'/
     +         ' SOLVE..TO OBTAIN A PROGRAM SOLUTION USING'/
     +         '        THE FLOWRATES PRESENTLY CONTAINED'/
     +         '        IN THE FLOW ARRAY'/
     +         ' STOP...TO END RUN')
 205  WRITE(NW,210)
 210  FORMAT(' COMMAND...')
      READ(NR2,220)CMND
 220  FORMAT(A4)
      IC=0
      IF(CMND.EQ.GO) IC=1
      IF(CMND.EQ.XLIST) IC=2
      IF(CMND.EQ.ALTER) IC=3
      IF(CMND.EQ.SOLVE) IC=4
      IF(CMND.EQ.STOP) STOP
      IF(IC.GT.0) GOTO 240
      WRITE(NW,230)CMND
 230  FORMAT(1X,A4,' INVALID..RETYPE')
      GOTO 205
 240  GO TO(100,110,5,300),IC
C
C     CHANGE INITIAL FLOWRATE ASSUMPTIONS. (CHECKS CONTINUITY)
C
 5    WRITE(NW,10)
 10   FORMAT(/' THE FLOW ARRAY CONTAINS PREVIOUSLY ASSIGNED'/
     +          ' OR CALCULATED FLOW VALUES. DO YOU WISH TO '/
     +          ' ZERO THE FLOW ARRAY ELEMENTS? (YES/NO)...')
      READ(NR2,14)ANS1
 14   FORMAT(A1)
      IF(ANS1.NE.YES)GO TO 15
      DO 16 J=1,NQ
      Q(J)=0.0
 16   CONTINUE
 15   CONTINUE
      WRITE(NW,17)
 17   FORMAT(/' SUPPLY NUMBER OF FLOW CHANGES,'/
     +          '                       (FORMAT I3)...')
      READ(NR2,20)NO
 20   FORMAT(I3)
      IF(NO.LE.NQ)GO TO 30
      WRITE(NW,40)NO,NQ
 40   FORMAT(/' NO =',I4,' EXCEEDES NQ =',I3)
```

```
      GO TO 15
 30   IF(NO.EQ.0)GO TO 205
      WRITE(NW,50)
 50   FORMAT(/' SUPPLY UPSTREAM NODE NUMBER, DOWNSTREAM NODE'/
     +' NUMBER AND FLOW VARIABLE VALUE , (FORMAT I3,I3,F15.5)')
      DO 60 I=1,NO
 65   WRITE(NW,70)I
 70   FORMAT(/' CHANGE NUMBER ',I3,'...')
      READ(NR2,80)IUS,IDS,Q2
 80   FORMAT(2I3,F15.5)
      IF(IDS.NE.0)GO TO 85
      IF(IUS.GT.N)GO TO 86
      ISLACK=L2+IUS
      Q(ISLACK)=Q2
      GO TO 60
 85   CONTINUE
      DO 90 J=1,L2
      IF(IUS.EQ.NUS(J))GO TO 91
      GO TO 90
 91   IQ=J
      IF(IDS.EQ.NDS(J))GO TO 92
 90   CONTINUE
 86   CONTINUE
      WRITE(NW,93)IUS,IDS
 93   FORMAT(/' LINK',I4,' TO',I4,' DOES NOT EXIST'/
     +' PLEASE RESPECIFY ')
      GO TO 65
 92   Q(IQ)=Q2
 60   CONTINUE
C
C     CHECK CONTINUITY.
C
      DO 95 I=1,N
      SUM=0.0
      DO 96 J=1,NQ
      SUM=SUM+A(I,J)*Q(J)
 96   CONTINUE
      IF(ABS(SUM-B(I)).LT.1.0E-10)GO TO 95
      WRITE(NW,97)I,SUM,B(I)
 97   FORMAT(/' WARNING, CONTINUITY NOT ESTABLISHED AT NODE',I3/
     +         ' ERRORS WILL RESULT IF CONTINUITY IS'/
     +         ' REQUIRED IN THE SOLUTION ROUTINE'/
     +' SUM=',F15.5,' STIP=',F15.5)
      GO TO 205
 95   CONTINUE
      GO TO 205
C
C     LIST INITIAL FLOWRATE ASSUMPTIONS.
C
 110  WRITE(NW,140)
 140  FORMAT(/' INITIAL FLOW ASSUMPTIONS'/
     +' U/S NODE   D/S NODE   FLOW(MGD)')
      DO 130 I=1,L2
      IF(ABS(Q(I)).LT.1.0E-9) GOTO 130
      WRITE(NW,150)NUS(I),NDS(I),Q(I)
  150 FORMAT(I5,I10,F15.5)
  130 CONTINUE
      DO 135 I=1,N
      ISLACK=L2+I
      IF(ABS(Q(ISLACK)).LT.1.0E-10)GO TO 135
      WRITE(NW,136)I,Q(ISLACK)
  136 FORMAT(I5,9X,'0',F15.5)
  135 CONTINUE
      GO TO 205
C
C     OBTAIN A SOLUTION FOR VALUES IN Q(). (CHECKS FEASABILITY)
C
  300 CONTINUE
      ICONT=0
      DO 310 I=1,N
      SUM=0.0
      ISLACK=L2+I
      DO 320 J=1,NQ
      SUM=SUM+A(I,J)*Q(J)
  320 CONTINUE
      IF(ABS(SUM-B(I)).LT.1.0E-10)GO TO 310
      ICONT=1
      Q(ISLACK)=A(I,ISLACK)*(B(I)-SUM)
  310 CONTINUE
      IF(ICONT.EQ.1)WRITE(NW,330)
  330 FORMAT(/' WARNING, CONTINUITY NOT ESTABLISHED IN SOLUTION,'/
     +         ' SLACK VARIABLES ALTERED TO ESTABLISH CONTINUITY')
      DO 340 I=I,N
      ISLACK=L2+I
      IF(Q(ISLACK).GT.-1.0E-10)GO TO 340
      WRITE(NW,350)
  350 FORMAT(/' NEGATIVE SLACK VARIABLE FOUND, SOLUTION IS'/
     +         ' INFEASABLE. PLEASE RESPECIFY SOLUTION.')
      GO TO 205
  340 CONTINUE
      CALL SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
     +            COST,ITYPE,N,NCEN,NQ)
      DO 360 I=1,NQ
      C(I)=C(I)-CPEN(I)
  360 CONTINUE
      CALL REPORT(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,
     +            XL,ITYPE,N,NCEN,NQ,NW,QLIMIT)
      GO TO 205
  100 CONTINUE
      RETURN
      END
      SUBROUTINE MODIFY(A,B,CPEN,NDS,NUS,STATE,STIP,XL,
     +                  ITYPE,N,NCEN,NENTRY,NQ,NR2,NW,QLIMIT)
C
C    ****************************************************************
C     CONVERSATIONAL ROUTINE TO ALLOW THE USER TO MODIFY THE
C     NETWORK BY REMOVING LINKS OR NODES OR CHANGING THE
C     STIPULATIONS OR STATES.
C
C     A      = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C              DEFINING THE PROBLEM CONSTRAINTS.
C     B      = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C              STIPULATIONS, (WORKING ARRAY OF STIP)
C     CPEN   = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C              FOR THE DESIGN VARIABLES.
C     NDS    = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS    = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     STATE  = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C              OF THE NODAL POINTS.
C     STIP   = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C              STIPULATIONS.
C     XL     = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C              ASSOCIATED WITH EACH DESIGN VARIABLE.
C     ITYPE  = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C              ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                        ITYPE=2 FOR A COLLECTION NETWORK.
C     N      = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN   = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NENTRY = FLAG SET BY SUBROUTINE MODIFY TO DIRECT EXECUTION
C              INTO SUBROUTINE QINIT. NENTRY=1 TO GO INTO QINIT.
C     NQ     = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C              ARTIFICIAL SLACK VARIABLES.
C     NR2    = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW     = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT = LIMITING FLOWRATE VALUE. FLOWRATES LESS THAN
C              QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C    ****************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),CPEN(NQ),NDS(NQ)
      DIMENSION NUS(NQ),STATE(N),STIP(N),XL(NQ)
C
C     DEFINE ALLOWABLE COMMANDS.
C
      DATA GO,XLINKS,XNODES,XSTIP,XSTATE,BACK,STOP/4HGO  ,4HLINK,
     +     4HNODE,4HSTIP,4HSTAT,4HBACK,4HSTOP/
      DATA YES/1HY/
      L2=NQ-N
      L=L2/2
      NENTRY=0
C
C     OUTPUT COMMAND OPTIONS.
```

```
C
      WRITE(NW,25)
 25   FORMAT(' IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:',/,
     +         ' GO......TO PROCEED',/,
     +         ' LINKS...TO DELETE OR RESTORE LINKS'/
     +         ' NODES...TO DELETE OR RESTORE NODES'/
     +         ' STIP....TO CHANGE STIPULATIONS',/,
     +         ' STATE...TO CHANGE NODAL STATES',/,
     +         ' BACK....TO GO BACK INTO "QINIT"',/,
     +         ' STOP....TO END RUN')
 30   WRITE(NW,35)
 35   FORMAT(' COMMAND...')
      READ(NR2,40)CMND
 40   FORMAT(A4)
      IC=0
      IF(CMND.EQ.STOP) STOP
      IF(CMND.EQ.GO) RETURN
      IF(CMND.EQ.XLINKS) IC=1
      IF(CMND.EQ.XNODES) IC=2
      IF(CMND.EQ.XSTIP) IC=3
      IF(CMND.EQ.XSTATE) IC=4
      IF(CMND.EQ.BACK) IC=5
      IF(IC.GT.0) GOTO 50
      WRITE(NW,45)CMND
 45   FORMAT(1X,A4,' INVALID...RETYPE')
      GOTO 30
 50   GO TO(100,200,400,600,700),IC
C
C     DELETE OR RESTORE LINKS
C
 100  WRITE(NW,105)
 105  FORMAT(' SUPPLY NO. OF LINK CHANGES'/
     +         '                   (FORMAT I3)...')
      READ(NR2,110)NLINK
 110  FORMAT(I3)
      IF (NLINK.EQ.0)GO TO 30
      IF(NLINK.LE.L2) GOTO 120
      WRITE(NW,115)L2
 115  FORMAT(' ONLY',I3,' LINKS IN SYSTEM')
      GOTO 100
 120  CONTINUE
      WRITE(NW,125)
 125  FORMAT(' SUPPLY U/S AND D/S NODE NOS, (FORMAT 2I3)...'/
     +         ' (-VE TO DELETE, +VE TO RESTORE)')
      DO 130 I=1,NLINK
 135  WRITE(NW,140)I
 140  FORMAT(' CHANGE NO.',I3,'...')
      READ(NR2,145)IUS,IDS
 145  FORMAT(2I3)
      SIGN=0.0
      IF(IUS.LT.0) SIGN=1.0
      IUS=IABS(IUS)
      IDS=IABS(IDS)
      DO 150 J=1,L2
      IF(IUS.EQ.NUS(J)) GOTO 155
      GOTO 150
  155 LINK=J
      IF(IDS.EQ.NDS(J)) GOTO 165
  150 CONTINUE
      WRITE(NW,160)IUS,IDS
  160 FORMAT(' LINK',I3,' TO',I3,' DOES NOT EXIST...RESPECIFY')
      GOTO 135
  165 IF(SIGN*CPEN(LINK).GT.0.1)WRITE(NW,170)IUS,IDS
      IF(SIGN.LT.0.1.AND.CPEN(LINK).LT.0.1)WRITE(NW,175)IUS,IDS
  170 FORMAT(' LINK',I3,'-',I3,' ALREADY DELETED')
  175 FORMAT(' LINK',I3,'-',I3,' ALREADY AVAILABLE')
      CPEN(LINK)=SIGN*1.0E10
  130 CONTINUE
      GOTO 30
C
C     DELETE OR RESTORE NODES:
C
  200 WRITE(NW,205)
  205 FORMAT(' SUPPLY NO. OF NODE CHANGES'/
     +       '                (FORMAT I3)...')
      READ(NR2,210)NNODE
  210 FORMAT(I3)
      IF(NNODE.LE.N) GOTO 220
      WRITE(NW,215)N
  215 FORMAT(' ONLY',I3,' NODES IN SYSTEM...RESPECIFY')
      GOTO 200
  220 CONTINUE
      WRITE(NW,225)
  225 FORMAT(' SUPPLY NODE NO, (FORMAT I3)'/
     +       ' (-VE TO DELETE, +VE TO RESTORE)')
      DO 245 J=1,NNODE
  230 WRITE(NW,235) J
  235 FORMAT(' CHANGE NO.',I3,'...')
      READ(NR2,240) JN
  240 FORMAT(I3)
      IF(JN.LE.N) GO TO 242
      WRITE(NW,243)JN
  243 FORMAT(' NODE',I3,'DOES NOT EXIST, PLEASE RESPECIFY')
      GO TO 230
  242 CONTINUE
      SIGN=1.0
      IF(JN.LT.0) SIGN=0.0
      JN=IABS(JN)
      IF(SIGN.GT.0.5) GOTO 250
      IF(ABS(B(JN)).LT.0.1*QLIMIT)WRITE(NW,255) JN
  255 FORMAT(' NODE',I3,' ALREADY INACTIVE')
      GOTO 260
  250 IF(ABS(STIP(JN)-B(JN)).LT.1.0E-10)WRITE(NW,265) JN
  265 FORMAT(' NODE',I3,' ALREADY ACTIVE')
  260 CONTINUE
      IF(SIGN.LT.0.5)B(JN)=0.1*QLIMIT
      IF(SIGN.GT.0.5)B(JN)=STIP(JN)
      IF(ITYPE.EQ.1)GO TO 245
      IF(JN.GT.NCEN)GO TO 245
      ISLACK=L2+JN
      IF(SIGN.GT.0.5)CPEN(ISLACK)=0.0
      IF(SIGN.LT.0.5)CPEN(ISLACK)=1.0E10
  245 CONTINUE
      GOTO 30
C
C     CHANGE STIPULATIONS.
C
  400 CONTINUE
      WRITE(NW,410)
  410 FORMAT(/' SUPPLY NO. OF STIPULATION CHANGES'/
     +       '                         (FORMAT I3)...')
      READ(NR2,420)NSTIP
  420 FORMAT(I3)
      IF(NSTIP.LE.N)GO TO 430
      WRITE(NW,440)NSTIP
  440 FORMAT(/' NSTIP=',I3,' THERE ARE ONLY',I3,' NODES AVAILABLE,'
     +,' RESPECIFY')
      GO TO 400
  430 IF(NSTIP.EQ.0)GO TO 30
      WRITE(NW,460)
  460 FORMAT(' SUPPLY NODE NO. AND NEW STIPULATION,'/
     +       '                FORMAT I3,F15.5'/)
      DO 450 I=1,NSTIP
  510 WRITE(NW,470)I
  470 FORMAT(' CHANGE NO.',I3,'...')
      READ(NR2,480)ISTIP,VSTIP
  480 FORMAT(I3,F15.5)
      IF(ISTIP.LE.N)GO TO 490
      WRITE(NW,500)ISTIP
  500 FORMAT(/' NODE NO.',I3,' DOES NOT EXIST, RESPECIFY')
      GO TO 510
  490 CONTINUE
      IF(ABS(VSTIP).LT.1.0E-10)WRITE(NW,485)
  485 FORMAT(' CAUTION- ZERO STIPULATIONS MAY CAUSE INSTABILITY'/
     +' IN THE SOLUTION ROUTINE. USE AN INSIGNIFICANT VALUE'/
     +' INSTEAD (0.1*QLIMIT)')
      ISLACK=L2+ISTIP
      IF(A(ISTIP,ISLACK)*VSTIP.GT.-1.0E-10)GO TO 495
      WRITE(NW,494)
  494 FORMAT(' NEW STIPULATION HAS INCORRECT SIGN,'/
     +'             PLEASE RESPECIFY')
      GO TO 510
  495 CONTINUE
      B(ISTIP)=VSTIP
  450 CONTINUE
      GO TO 30
C
C     CHANGE NODAL STATE.
C
  600 CONTINUE
      WRITE(NW,610)
  610 FORMAT(/' SUPPLY NO. OF STATE CHANGES'/
     +        '                    (FORMAT I3)...')
      READ(NR2,615)NSTATE
  615 FORMAT(I3)
      IF(NSTATE.LE.N) GO TO 620
      WRITE(NW,625)NSTATE,N
  625 FORMAT(/' NSTATE=',I3,' THERE ARE ONLY',I3,' NODES AVAILABLE,'/
     +        ' PLEASE RESPECIFY')
      GO TO 600
  620 IF(NSTATE.EQ.0)GO TO 30
      WRITE(NW,640)
  640 FORMAT(' SUPPLY NODE NO. AND NEW STATE, FORMAT I3,F15.5'/)
      DO 630 I=1,NSTATE
  645 WRITE(NW,650)I
  650 FORMAT(' CHANGE NO.',I3,' ...')
      READ(NR2,655)ISTATE,VSTATE
  655 FORMAT(I3,F15.5)
      IF(ISTATE.LE.N) GO TO 660
      WRITE(NW,665)ISTATE
  665 FORMAT(/' NODE NO.',I3,' DOES NOT EXIST, RESPECIFY')
      GO TO 645
  660 CONTINUE
      STATE(ISTATE)=VSTATE
  630 CONTINUE
      GO TO 30
C
C     GO BACK TO "QINIT"
C
  700 CONTINUE
      NENTRY=1
      RETURN
      END
      SUBROUTINE SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
     +                  COST,ITYPE,N,NCEN,NQ)
C
C    *************************************************************
C     THIS SUBROUTINE CALLS LNKCST AND CENCST TO CALCULATE
C     THE COST COEFFICIENTS FOR A SPECIFIED SOLUTION. IT THEN
C     CALCULATES THE TOTAL COST FOR THE SPECIFIED SOLUTION.
C
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLE FLOWRATES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     COST    = COMPUTED SOLUTION COST.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                         ITYPE=0 FOR A COLLECTION NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C
C     USES ROUTINES LNKCST, CENCST,
C     USER DEFINED COST1,COST2.
C    *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION B(N),C(NQ),CPEN(NQ),NDS(NQ)
      DIMENSION NUS(NQ),Q(NQ),STATE(N),XL(NQ)
C
      L2=NQ-N
C
C     CALCULATE COST COEFFICIENTS.
C
      CALL LNKCST(C,CPEN,NDS,NUS,Q,STATE,XL,N,NQ)
      CALL CENCST(B,C,CPEN,Q,ITYPE,N,NCEN,NQ)
C
      COST=0.0
      DO 10 I=1,L2
 10   COST=COST+C(I)*Q(I)
      IF(ITYPE.EQ.0) GO TO 40
C
C     COST CALCULATION FOR DISTRIBUTION NETWORKS.
```

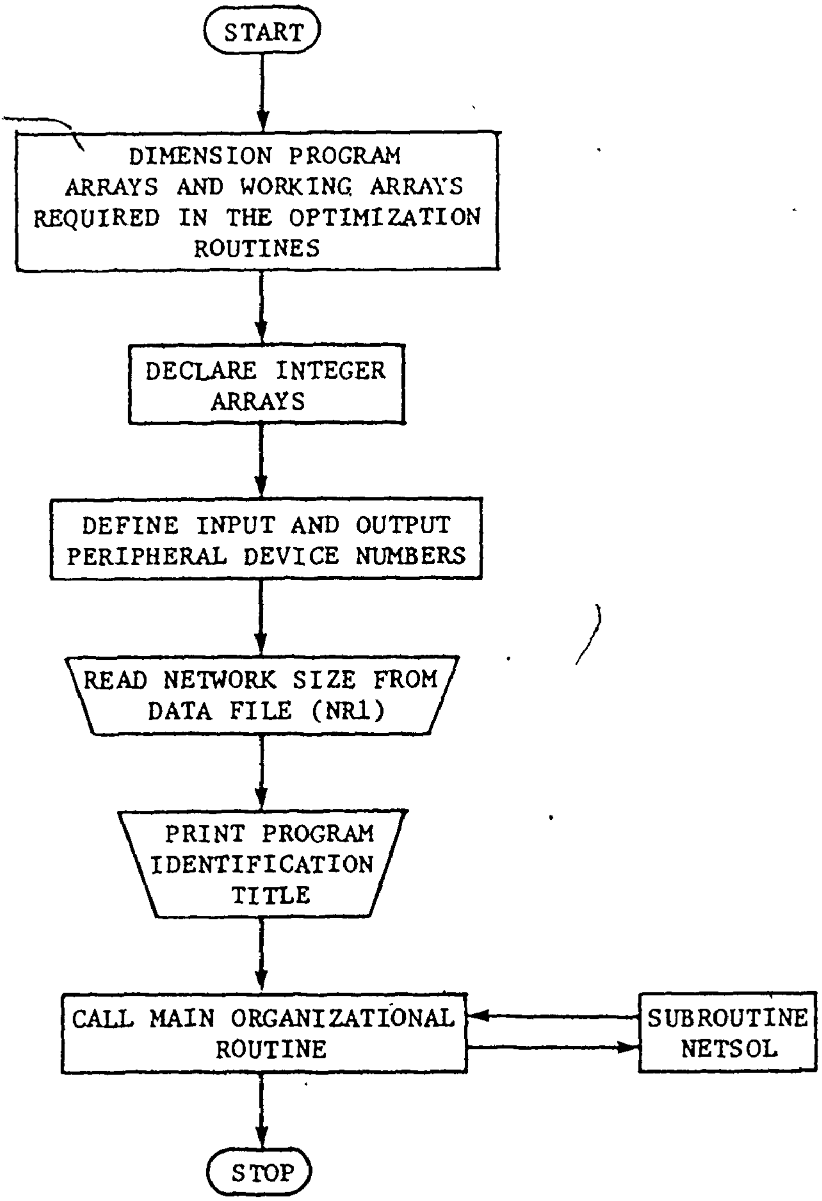
```
C
      DO 20 I=1,NCEN
      J=L2+I
      QJ=B(I)-Q(J)
 20   COST=COST+C(J)*QJ
      NCENP1=NCEN+1
      DO 30 I=NCENP1,N
      J=L2+I
 30   COST=COST+C(J)*Q(J)
      GO TO 50
C
C     COST CALCULATION FOR COLLECTION NETWORKS.
C
 40   CONTINUE
      DO 60 I=1,N
      J=L2+I
      COST=COST+C(J)*Q(J)
 60   CONTINUE
 50   CONTINUE
      RETURN
      END
      SUBROUTINE LNKCST(C,CPEN,NDS,NUS,Q,STATE,XL,N,NQ)
C
C    *************************************************************
C     CALCULATES COST COEFFICIENTS FOR TRANSPORTATION LINKS USING
C     COST1.
C
C     C       = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLE FLOWRATES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C
C     USES USER DEFINED SUBROUTINE COST1.
C    *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION C(NQ),CPEN(NQ),NDS(NQ),NUS(NQ)
      DIMENSION Q(NQ),STATE(N),XL(NQ)
C
      L2=NQ-N
      DO 10 I=1,L2
      C(I)=0.0
      IF(Q(I).LT.1.0E-10)GO TO 20
      QI=Q(I)
      NUSI=NUS(I)
      NDSI=NDS(I)
      STUS=STATE(NUSI)
      STDS=STATE(NDSI)
      XLI=XL(I)
C
C     COST ROUTINE
C
      CALL COST1(COST,I,QI,STDS,STUS,XLI)
C
      C(I)=COST/QI
 20   C(I)=C(I)+CPEN(I)
 10   CONTINUE
      RETURN
      END
      SUBROUTINE COST1(COST,I,QI,STDS,STUS,XLI)
C
C   *************************************************************
C     USER SUPPLIED COST ROUTINE*****
C     COMPUTES THE TOTAL TRANSPORTATION COST FOR THE (I)TH FLOW
C     VARIABLE.
C     ROUTINE SHOULD CONTAIN ANY RELEVANT LABELLED COMMON
C     BLOCK STORAGE USED IN ROUTINES "CNSTIN" AND "REPORT"
C     FOR EITHER INPUT OF PARAMETERS OR RETURN OF DESIGN
C     PARAMETERS.
C
C     COST    = COMPUTED TRANSPORTATION COST FOR Q(I).
C     I       = FLOW VARIABLE IDENTIFICATION NUMBER.
C     QI      = CURRENT VALUE OF FLOWRATE Q(I).
C     STDS    = STATE AT DOWNSTREAM NODE OF Q(I).
C     STUS    = STATE AT UPSTREAM NODE OF Q(I).
C     XLI     = LENGTH OF LINK FOR Q(I).
C   *************************************************************
C
C     START OF USER PROVIDED CODING*****
C     USER SUPPLIES THE CODING NEEDED TO PROVIDE THE COST OF
C     TRANSPORTATION FOR THE (I)TH FLOW VARIABLE HAVING THE
C     CURRENT VALUE QI. THIS MAY REQUIRE THE CODING OF DESIGN
C     COMPUTATIONS. ANY SPECIFIC CONSTANTS OR VARIABLES
C     REQUIRED FOR INPUT OR OUTPUT MAY BE TRANSFERRED THROUGH
C     COMMON BLOCK.
C
C     COST=.....
C
C     END OF USER PROVIDED CODING*****
      RETURN
      END
      SUBROUTINE CENCST(B,C,CPEN,Q,ITYPE,N,NCEN,NQ)
C
C    *************************************************************
C     CALCULATES THE COST COEFFICIENTS FOR PROCESSING NODES.
C
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLE FLOWRATES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                         ITYPE=0 FOR A COLLECTION NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C
C     USES USER SUPPLIED ROUTINE COST2.
C    *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION B(N),C(NQ),CPEN(NQ),Q(NQ)
C
      DO 10 I=1,NCEN
      ICEN=(NQ-N)+I
      QI=B(I)-Q(ICEN)
      IF(ITYPE.EQ.0)QI=Q(ICEN)
      C(ICEN)=0.0
      IF(QI.LT.1.0E-10)GO TO 20
C
C     USER SUPPLIED COST ROUTINE.
C
      CALL COST2(COST,ICEN,QI)
C
      C(ICEN)=COST/QI
 20   C(ICEN)=C(ICEN)+CPEN(ICEN)
 10   CONTINUE
      NCENP1=NCEN+1
      DO 30 I=NCENP1,N
      II=(NQ-N)+I
      C(II)=1.0E10
 30   CONTINUE
      RETURN
      END
      SUBROUTINE COST2(COST,ICEN,QI)
C
C   **************************************************************
C     USER SUPPLIED SUBROUTINE*****
C     COMPUTES THE TOTAL PROCESSING COST FOR A SPECIFIED
C     PROCESSING CENTRE AND FLOW QI. ROUTINE SHOULD CONTAIN
C     ANY RELEVANT LABELLED COMMON BLOCK STORAGE USED IN ROUTINES
C     "CNSTIN" AND "REPORT" FOR EITHER INPUT OF PARAMETERS OR
C     RETURN OF DESIGN PARAMETERS.
C
C     COST   = COMPUTED PROCESSING COST FOR NODE ICEN.
C     ICEN   = NUMBER OF PROCESSING NODE BEING COSTED.
C     QI     = VOLUME RATE OF FLOW BEING PROCESSED.
C   **************************************************************
C
C     START OF USER PROVIDED CODING*****
C     THE USER PROVIDES CODING HERE TO PROVIDE THE COST OF
C     PROCESSING AT THE RATE, QI, FOR THE CENTRE, ICEN.
C     ANY RELEVANT DESIGN DATA SHOULD BE CALCULATED HERE AND
C     AND TRANSFERRED TO REPORT THROUGH COMMON BLOCK.
C
C     COST=.....
C
C     END OF USER PROVIDED CODING*****
      RETURN
      END
      SUBROUTINE SOLVE(A,B,C,CPEN,NDS,NUS,Q,STATE,XL,WA,WB,WC,
     +                 WD,WE,WF,WG,ITYPE,N,NCEN,NQ,NR2,NW,QLIMIT)
C
C    ************************************************************
C     SOLUTION SUBROUTINE TO DETERMINE OPTIMAL SOLUTION.
C     USES AN ITERATIVE LINEAR PROGRAMMING APPROACH.
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C               DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLE FLOWRATES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     WA-WG   = WORKING ARRAYS FOR OPTIMIZATION ROUTINE.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                         ITYPE=0 FOR A COLLECTION NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C     NR2     = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT  = LIMITING FLOWRATE VALUE. FLOWRATES LESS THAN
C               QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C
C     USES ROUTINES SIMPLE,REPLCQ,LNKCST,CENCST
C     USER SUPPLIED ROUTINES COST1,COST2
C    ************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NDS(NQ)
      DIMENSION NUS(NQ),Q(NQ),STATE(N),XL(NQ)
C
C      DIMENSION WORKING ARRAYS.
C
      DIMENSION WA(N,N),WB(N),WC(N),WD(N),WE(N),WF(N),WG(NQ)
      INTEGER WF
C
      DATA YES/1HY/
      L2=NQ-N
      MITER=5
      NSTOP=5*N
      NNDEX=1
      IDATA=0
C
C     INTERMEDIATE PRINTOUT IS AVAILABLE
      WRITE(NW,100)
  100 FORMAT(' IN ROUTINE "SOLVE"...DO YOU WANT DETAILS',/,
     +       ' OF COSTS DURING ITERATIONS. (YES/NO)...')
      READ(NR2,105)ANS
  105 FORMAT(A1)
      WRITE(NW,195)
  195 FORMAT(/' DO YOU WISH TO USE THE "REPLCQ" SUBROUTINE'/
     +       '                                 (YES/NO)...')
      READ(NR2,105)ANS1
      ITER=0
      DO 10 I=1,NQ
  10  WG(I)=Q(I)
C
C     START OF ITERATIVE SEARCH.
C
  30  CONTINUE
      IF(ITER.EQ.MITER) GO TO 130
      ITER=ITER+1
      WRITE(NW,120)ITER
  120 FORMAT(' ITERATION',I3)
C
C     CALCULATION OF COST COEFFICIENTS.
C
      IF(ANS1.NE.YES)GO TO 200
      CALL REPLCQ(A,NDS,NUS,Q,N,NCEN,NQ)
  200 CONTINUE
      DO 190 I=1,NQ
      IF(Q(I).LT.QLIMIT)Q(I)=QLIMIT
  190 CONTINUE
      CALL LNKCST(C,CPEN,NDS,NUS,Q,STATE,XL,N,NQ)
      CALL CENCST(B,C,CPEN,Q,ITYPE,N,NCEN,NQ)
C
C     CHANGE SIGN OF PROCESSING COST COEFFICIENTS FOR DISTRIBUTION
C     NETWORKS DUE TO OBJECTIVE FUNCTION.
C
      IF(ITYPE.EQ.0)GO TO 225
      DO 220 I=1,NCEN
      J=L2+I
      C(J)=2.0*CPEN(J)-C(J)
  220 CONTINUE
  225 CONTINUE
      DO 35 I=1,NQ
  35  Q(I)=WG(I)
C
C     OPTIMIZATION ROUTINE
C
      CALL SIMPLE(NQ,N,A,B,C,NSTOP,IDATA,NNDEX,WG,U,WA,WB,WC,
     +            WD,WE,WF)
      IF(ANS.NE.YES) GOTO 110
      NFLOW=L2+NCEN
      DO 40 I=1,NFLOW
      WRITE(NW,41)I,NUS(I),NDS(I),WG(I),C(I)
 41   FORMAT(' Q(',I2,') FROM NODES',I4,' TO',I4,2F12.1)
 40   CONTINUE
      WRITE(NW,42)U
 42   FORMAT(/' OBJECTIVE FUNCTION VALUE=',F15.1//)
 110  CONTINUE
      IERROR=0
      DO 20 I=1,NQ
      ERR=ABS(WG(I)-Q(I))
      IF(ERR.GT.QLIMIT)IERROR=1
 20   Q(I)=WG(I)
      IF(IERROR.EQ.1)GO TO 30
C
      GO TO 140
 130  CONTINUE
      WRITE(NW,150)
 150  FORMAT(//' MAXIMUM NUMBER OF ITERATIONS ENCOUNTERED, DO'/
     +' YOU WISH TO TRY AN ADDITIONAL 5 ITERATIONS'/
     +'                              (YES/NO) ...')
      READ(NR2,160)ANS2
 160  FORMAT(A1)
      IF(ANS2.NE.YES)GO TO 140
      MITER=MITER+5
      GO TO 30
 140  CONTINUE
      IF(ITYPE.EQ.0)GO TO 235
      DO 230 I=1,NCEN
      J=L2+I
      C(J)=2.0*CPEN(J)-C(J)
 230  CONTINUE
 235  CONTINUE
      RETURN
      END
      SUBROUTINE REPLCQ(A,NDS,NUS,Q,N,NCEN,NQ)
C
C   *************************************************************
C     ROUTINE TO CALCULATE REPLACEMENT VALUES FOR THE Q ARRAY.
C
C     A      = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C              DEFINING THE PROBLEM CONSTRAINTS.
C     NDS    = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS    = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C              NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q      = ARRAY OF FLOWRATE VARIABLES.
C     N      = TOTAL NUMBER OF NODES.
C     NCEN   = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ     = TOTAL NUMBER OF FLOWRATE VARIABLES.
C   *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),Q(NQ),NUS(NQ),NDS(NQ)
C
      L2=NQ-N
      DO 10 I=1,N
      COUNT=0.0
      SUM=0.0
      DO 20 J=1,L2
      IF(NUS(J).NE.I)GO TO 20
      IF(Q(J).LE.0.0)GO TO 20
      SUM=SUM+Q(J)
      COUNT=COUNT+1.0
 20   CONTINUE
      IF(SUM.EQ.0.0)GO TO 10
      DO 30 J=1,L2
      IF(NUS(J).NE.I)GO TO 30
      IF(Q(J).GT.0.0)GO TO 30
      IF(SUM.EQ.0.0) GO TO 30
      Q(J)=SUM/COUNT
 30   CONTINUE
 10   CONTINUE
      RETURN
      END
      SUBROUTINE REPORT(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,
     +                  XL,ITYPE,N,NCEN,NQ,NW,QLIMIT)
C
C   *************************************************************
C     USER AUGMENTED SUBROUTINE*****
C     SUBROUTINE TO OUTPUT SOLUTION DETAILS.
C     ONLY FUNDAMENTAL OUTPUT IS PROVIDED (IE. FLOWS AND COSTS).
C     IF REQUIRED, SUPLEMENTARY OUTPUT OF RELEVANT DATA MUST
C     BE PROVIDED FOR BY THE USER (SUCH AS DESIGN DATA OR COST
C     BREAKDOWN).
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C               DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE-DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLE FLOWRATES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NAMES   = TWO-DIMENSIONAL ARRAY CONTAINING THE NAMES
C               ASSIGNED TO THE NODAL POINTS.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. ITYPE=1 FOR A DISTRIBUTION NETWORK,
C                         ITYPE=0 FOR A COLLECTION NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT  = LIMITING FLOWRATE VALUE. FLOWRATES LESS THAN
C               QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C   *************************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NAMES(5,N)
      DIMENSION NDS(NQ),NUS(NQ),Q(NQ),STATE(N),XL(NQ)
      INTEGER*4 NAMES
C
C     ENTER ANY LABELLED COMMON BLOCKS REQUIRED TO TRANSFER
C     ADDITIONAL OUTPUT DATA.
C     START OF USER SUPPLIED COMMON*****
C
C     END OF USER SUPPLIED COMMON*****
C
```

```
      L2=NQ-N
      L=L2/2
C
C     PRINT RESULTS.
C
      WRITE(NW,10)
 10   FORMAT(//20X,'PROBLEM SOLUTION',/,
     +        '    PROCESSING CENTER COSTS',/,
     +        7X,'CENTER',17X,'FLOW',7X,'COST (MIL)'/)
      TOTPRC=0.0
      DO 100 I=1,NCEN
        ISLACK=L2 + I
        QPROC=B(I) - Q(ISLACK)
      IF(ITYPE.EQ.0)QPROC=Q(ISLACK)
      IF(QPROC.LT.QLIMIT)GO TO 100
        PROCST=C(ISLACK)*QPROC/1000000.0
        TOTPRC=TOTPRC+PROCST
        WRITE(NW,110)(NAMES(J,I),J=1,5),QPROC,PROCST
 110    FORMAT(2X,5A4,F13.2,F14.2)
 100  CONTINUE
C
C     NON-PROCESSING NODE COSTS.
C
      TOTDMC=0.0
      NDEM=L2+NCEN+1
      DO 30 J=NDEM,NQ
      IF(Q(J).LT.QLIMIT)GO TO 30
 40   WRITE(NW,2040)
 2040 FORMAT(//'    NON-PROCESSING CENTER COSTS'/,
     +7X,'CENTER',17X,'FLOW',7X,'COST (MIL)'/)
      NCENP1=NCEN+1
      DO 70 I=NCENP1,N
      ISLACK=L2+I
      QNPROC=Q(ISLACK)
      CSTNPR=C(ISLACK)*QNPROC/1000000.0
      TOTDMC=TOTDMC+CSTNPR
      WRITE(NW,2050)(NAMES(II,I),II=1,5),QNPROC,CSTNPR
 2050 FORMAT(2X,5A4,F13.2,F15.2)
 70   CONTINUE
 30   CONTINUE
      WRITE(NW,90)
C
C     TRANSPORTATION COSTS.
C
 90   FORMAT(//'    TRANSPORTATION COSTS',/,
     +        2X,'NODE',8X,'ROUTE',13X,'FLOW',5X,'COST (MIL)'/)
      TOTLKC=0.0
      DO 44 I=1,L2
        IF(Q(I).LT.QLIMIT)GO TO 44
        NUSI=NUS(I)
        NDSI=NDS(I)
        CSTLNK=C(I)*Q(I)/1000000.0
        TOTLKC=TOTLKC+CSTLNK
        WRITE(NW,46)NUSI,(NAMES(J,NUSI),J=1,5)
 46       FORMAT(1X,I4,2X,5A4,' TO')
        WRITE(NW,47)NDSI,(NAMES(J,NDSI),J=1,5),Q(I),CSTLNK
 47       FORMAT(2X,I4,1X,5A4,F9.2,F13.2)
 44     CONTINUE
C
C       START OF USER SUPPLIED PORTION*****
C       THE USER SUPPLIES CODING HERE
C       TO AUGMENT OUTPUT.
C       END OF USER SUPPLIED PORTION*****
C
C
C       PRINT COST SUMMARY.
C
        WRITE(NW,60)TOTPRC
 60     FORMAT(//10X,'PROCESS COST...... (MILLION)',F10.3)
        IF(TOTDMC.GT.1.0E-10)WRITE(NW,2090)TOTDMC
 2090   FORMAT(10X,'NON-PROCESS COST.. (MILLION)',F10.3)
        WRITE(NW,61)TOTLKC
 61     FORMAT(10X,'TRANSPN COST...... (MILLION)',F10.3)
        TOTCST=TOTLKC+TOTPRC+TOTDMC
        WRITE(NW,62)TOTCST
 62     FORMAT(10X,'TOTAL SYSTEM ..... (MILLION)',F10.3)
        WRITE(NW,63)
 63     FORMAT('   END OF REPORT',///)
        RETURN
        END
```

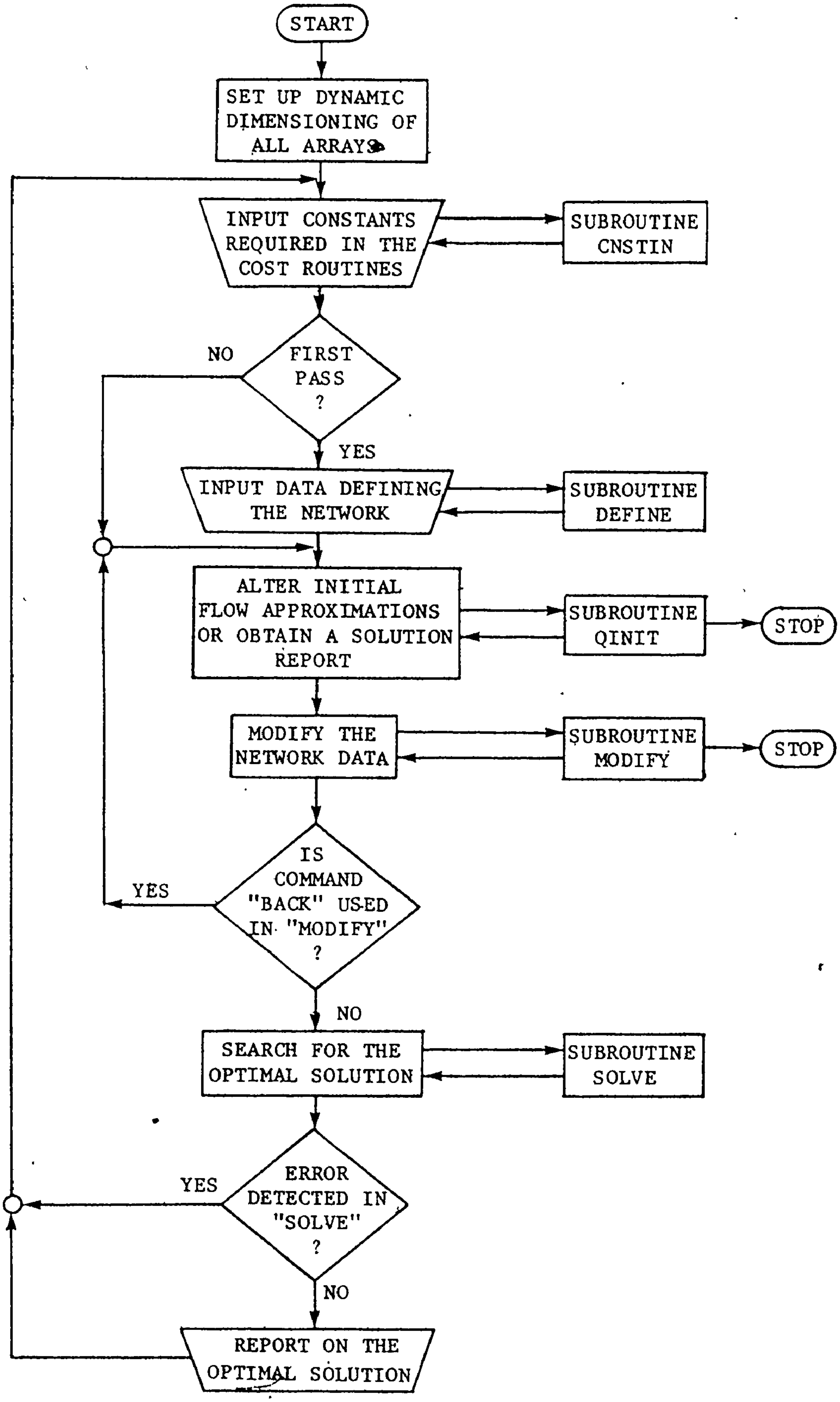
# APPENDIX D

## FLOWCHARTS FOR THE NETSOL PACKAGE ROUTINES

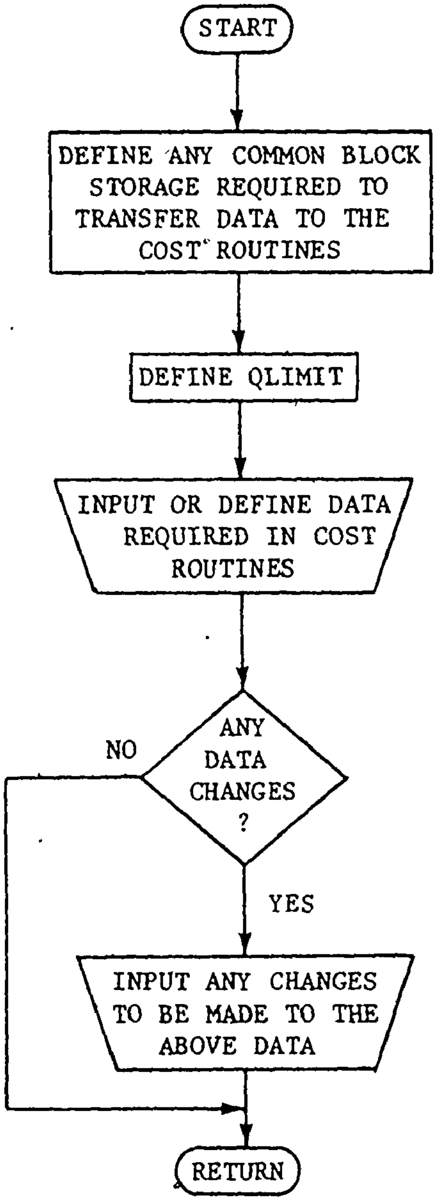
PROGRAM NETWRK


START
DIMENSION PROGRAM ARRAYS AND WORKING ARRAYS REQUIRED IN THE OPTIMIZATION ROUTINES
DECLARE INTEGER ARRAYS
DEFINE INPUT AND OUTPUT PERIPHERAL DEVICE NUMBERS
READ NETWORK SIZE FROM DATA FILE (NR1)
PRINT PROGRAM IDENTIFICATION TITLE
CALL MAIN ORGANIZATIONAL ROUTINE
SUBROUTINE NETSOL
STOP

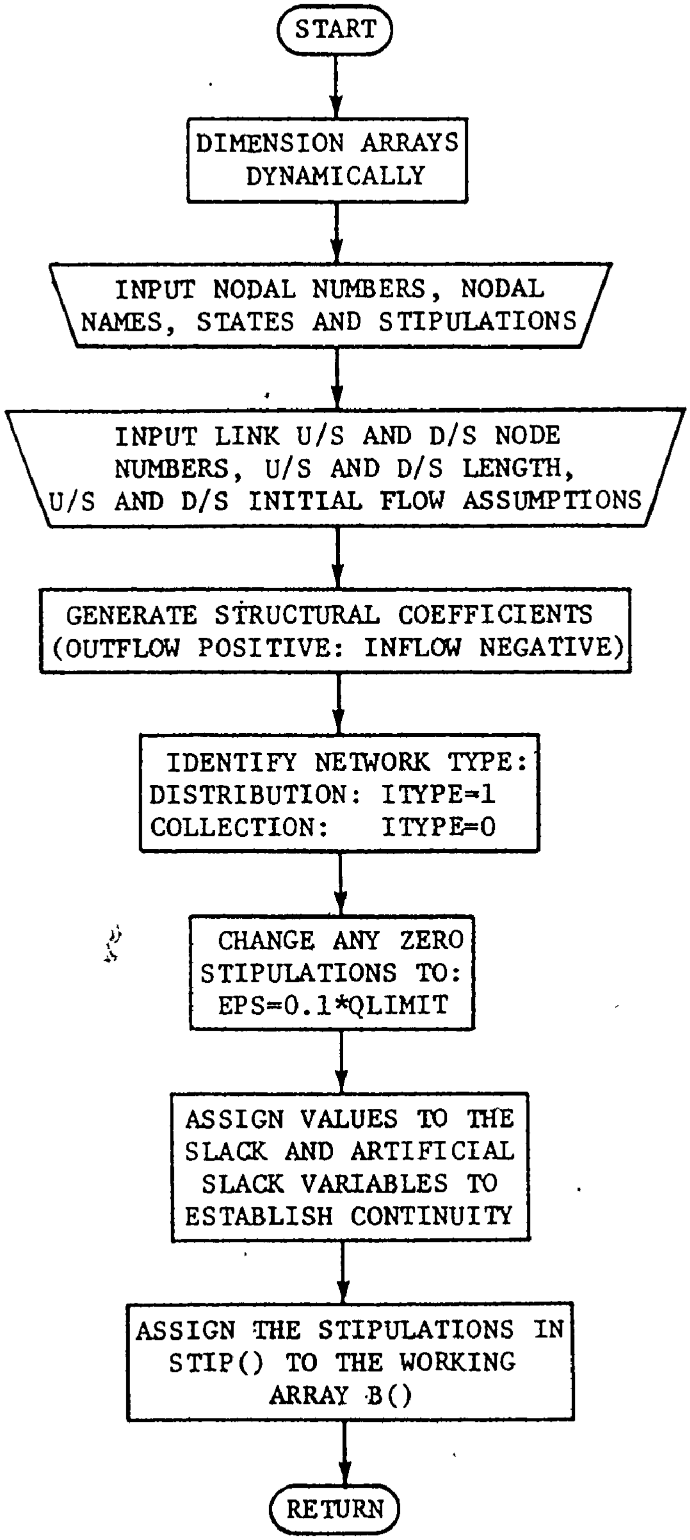
SUBROUTINE NETSOL


START
SET UP DYNAMIC DIMENSIONING OF ALL ARRAYS
INPUT CONSTANTS REQUIRED IN THE COST ROUTINES
SUBROUTINE CNSTIN
FIRST PASS ?
NO
YES
INPUT DATA DEFINING THE NETWORK
SUBROUTINE DEFINE
ALTER INITIAL FLOW APPROXIMATIONS OR OBTAIN A SOLUTION REPORT
SUBROUTINE QINIT
STOP
MODIFY THE NETWORK DATA
SUBROUTINE MODIFY
STOP
IS COMMAND "BACK" USED IN "MODIFY" ?
YES
NO
SEARCH FOR THE OPTIMAL SOLUTION
SUBROUTINE SOLVE
ERROR DETECTED IN "SOLVE" ?
YES
NO
REPORT ON THE OPTIMAL SOLUTION

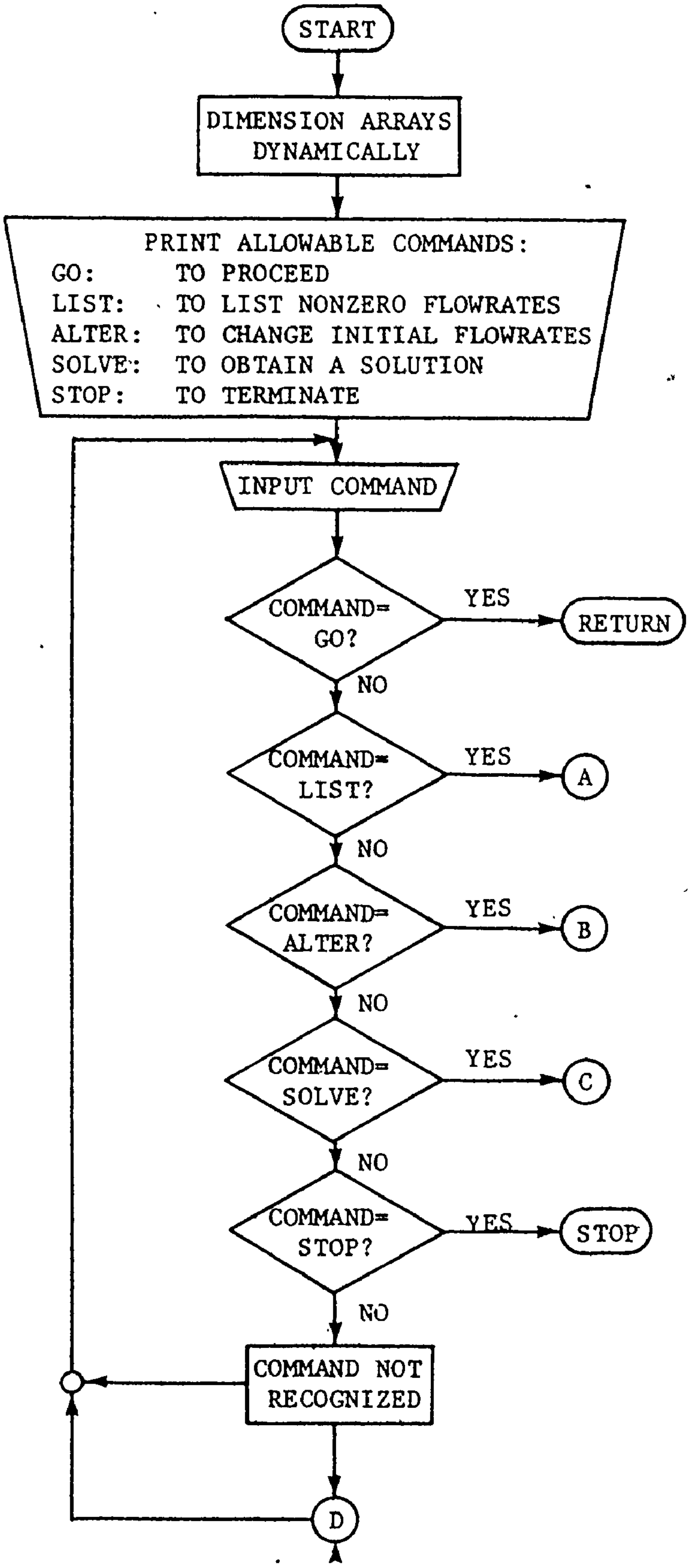
SUBROUTINE CNSTIN


START
DEFINE ANY COMMON BLOCK STORAGE REQUIRED TO TRANSFER DATA TO THE COST ROUTINES
DEFINE QLIMIT
INPUT OR DEFINE DATA REQUIRED IN COST ROUTINES
ANY DATA CHANGES ?
NO
YES
INPUT ANY CHANGES TO BE MADE TO THE ABOVE DATA
RETURN

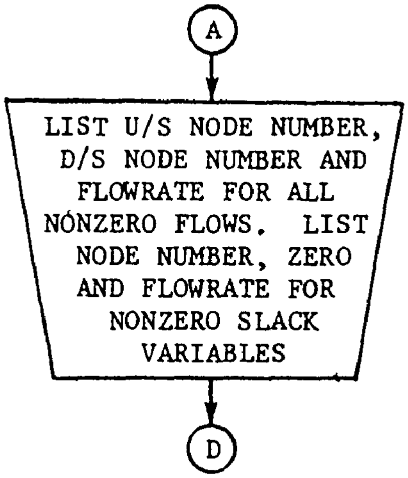
SUBROUTINE DEFINE


START
DIMENSION ARRAYS DYNAMICALLY
INPUT NODAL NUMBERS, NODAL NAMES, STATES AND STIPULATIONS
INPUT LINK U/S AND D/S NODE NUMBERS, U/S AND D/S LENGTH, U/S AND D/S INITIAL FLOW ASSUMPTIONS
GENERATE STRUCTURAL COEFFICIENTS (OUTFLOW POSITIVE: INFLOW NEGATIVE)
IDENTIFY NETWORK TYPE:
DISTRIBUTION: ITYPE=1
COLLECTION: ITYPE=0
CHANGE ANY ZERO STIPULATIONS TO: EPS=0.1*QLIMIT
ASSIGN VALUES TO THE SLACK AND ARTIFICIAL SLACK VARIABLES TO ESTABLISH CONTINUITY
ASSIGN THE STIPULATIONS IN STIP() TO THE WORKING ARRAY B()
RETURN

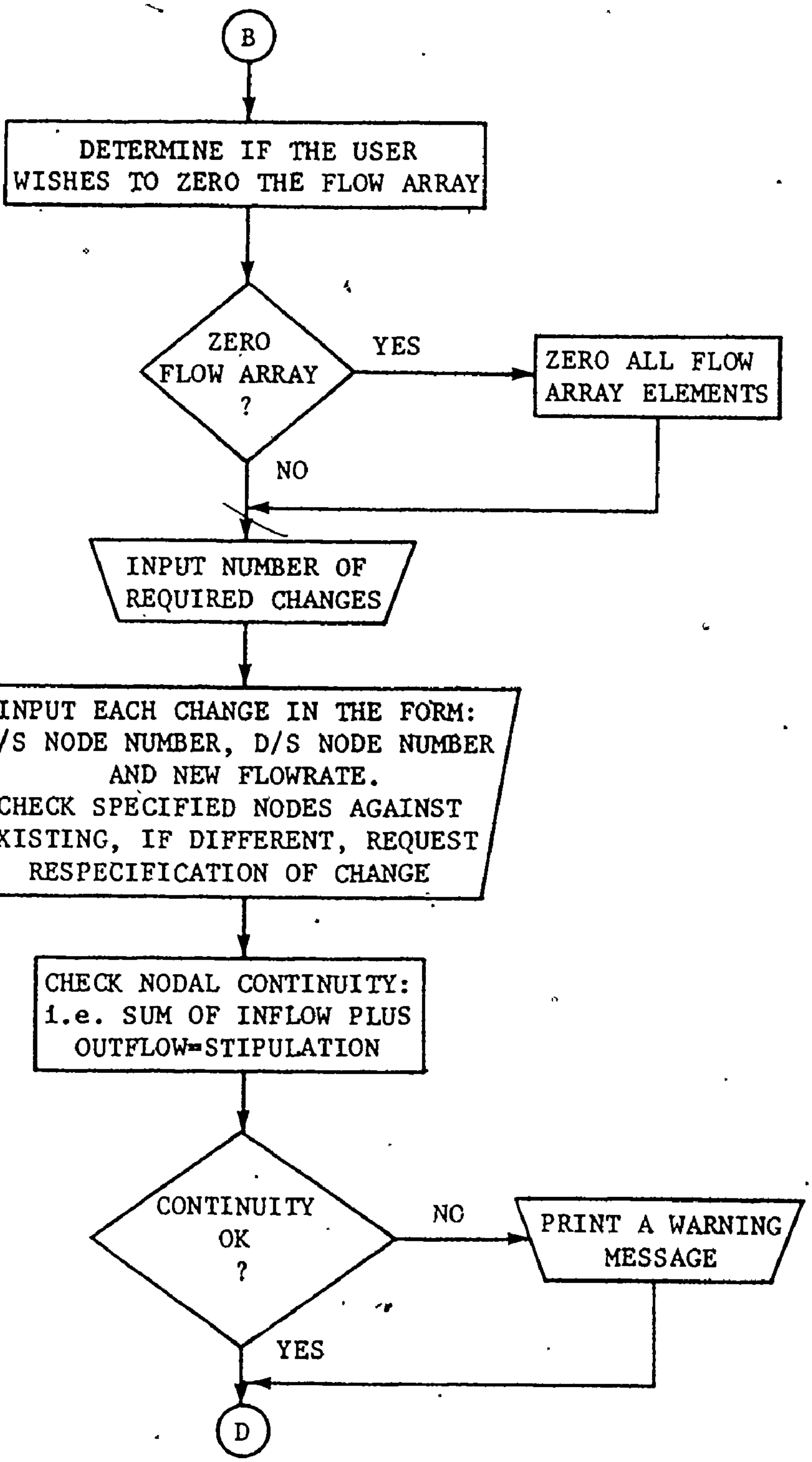
## SUBROUTINE QINIT



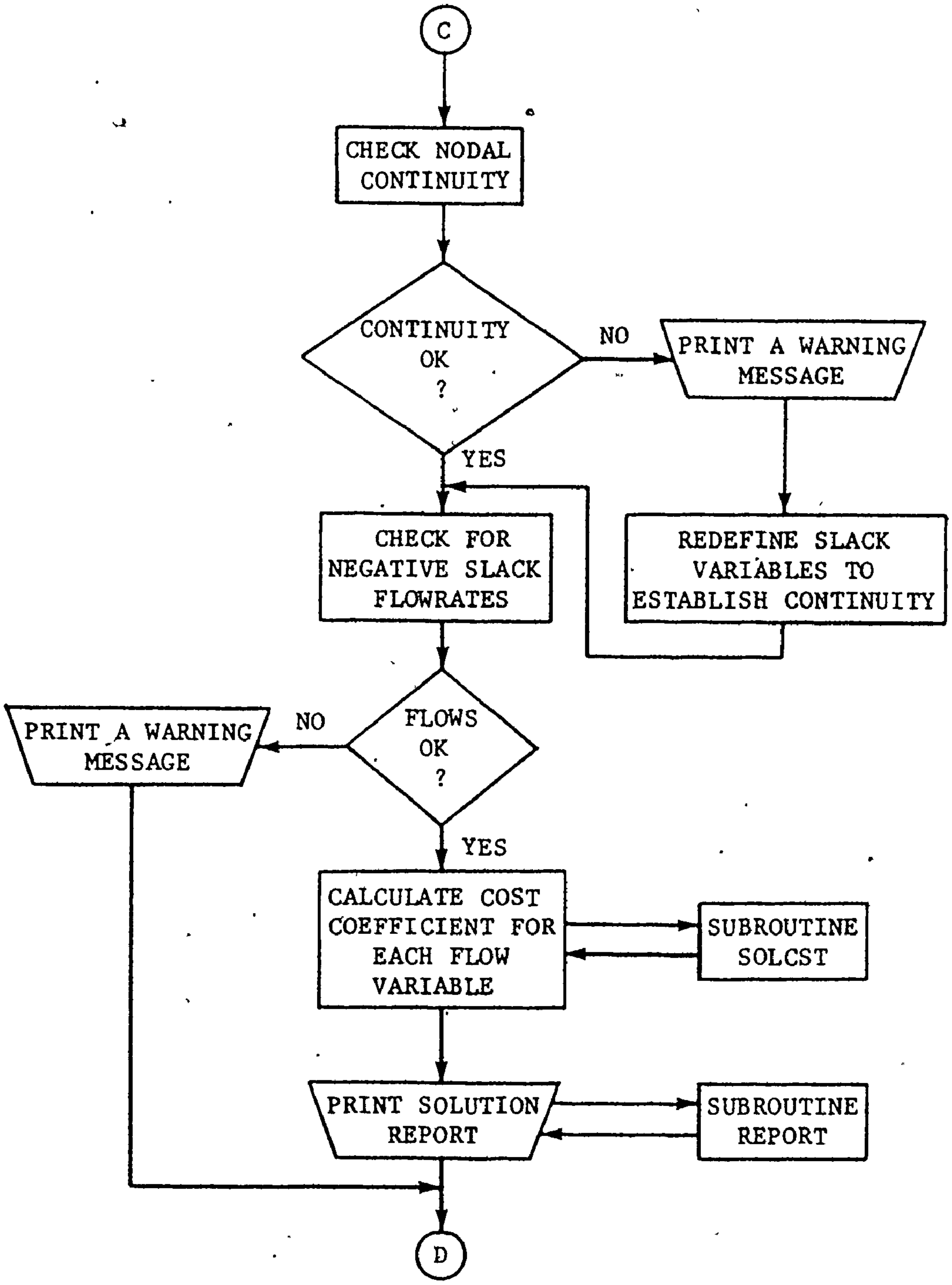
SUBROUTINE QINIT (cont.)


A
LIST U/S NODE NUMBER, D/S NODE NUMBER AND FLOWRATE FOR ALL NONZERO FLOWS. LIST NODE NUMBER, ZERO AND FLOWRATE FOR NONZERO SLACK VARIABLES
D

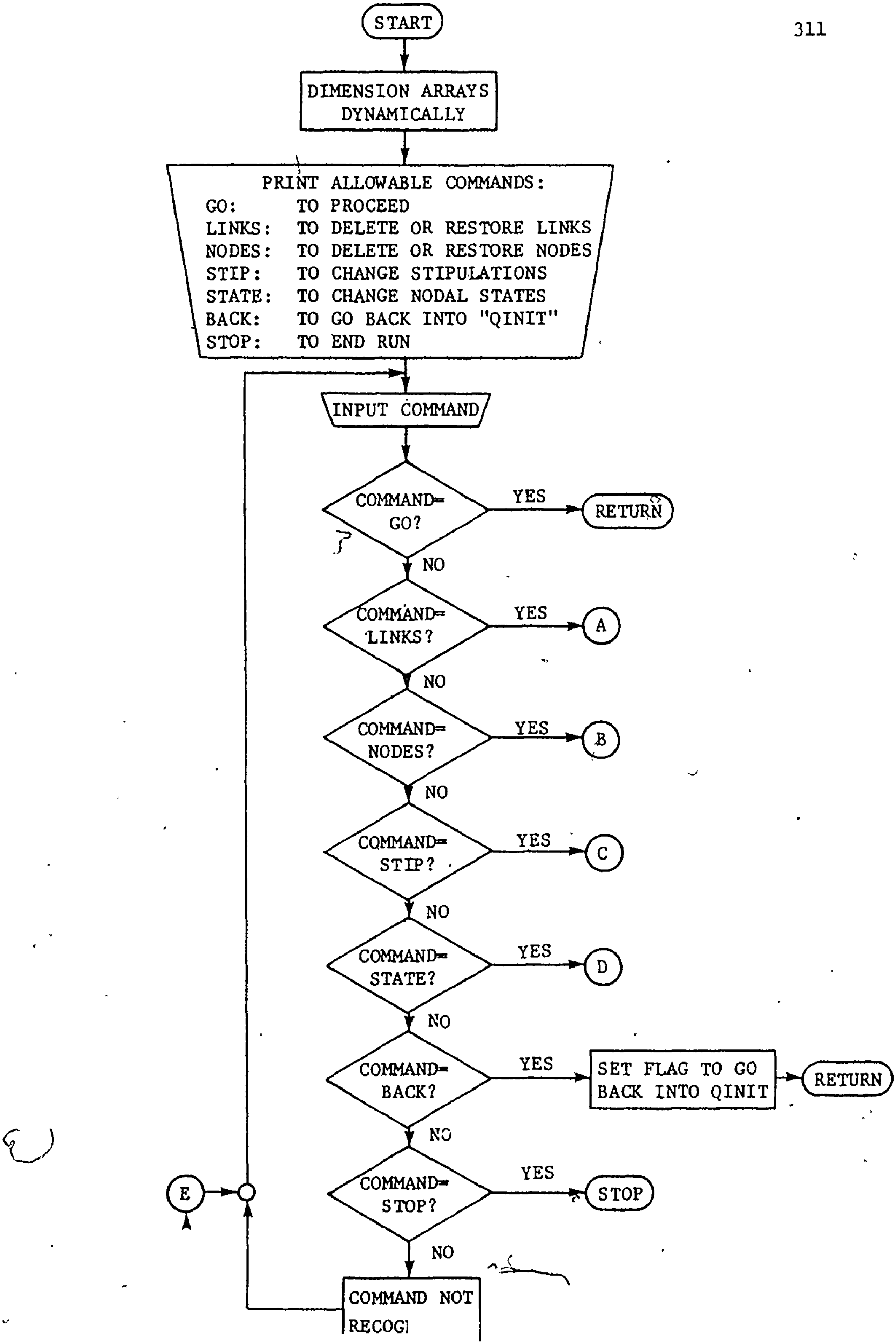
SUBROUTINE QINIT (cont.)


B
DETERMINE IF THE USER WISHES TO ZERO THE FLOW ARRAY
ZERO FLOW ARRAY ?
YES
ZERO ALL FLOW ARRAY ELEMENTS
NO
INPUT NUMBER OF REQUIRED CHANGES
INPUT EACH CHANGE IN THE FORM: U/S NODE NUMBER, D/S NODE NUMBER AND NEW FLOWRATE. CHECK SPECIFIED NODES AGAINST EXISTING, IF DIFFERENT, REQUEST RESPECIFICATION OF CHANGE
CHECK NODAL CONTINUITY: i.e. SUM OF INFLOW PLUS OUTFLOW=STIPULATION
CONTINUITY OK ?
NO
PRINT A WARNING MESSAGE
YES
D

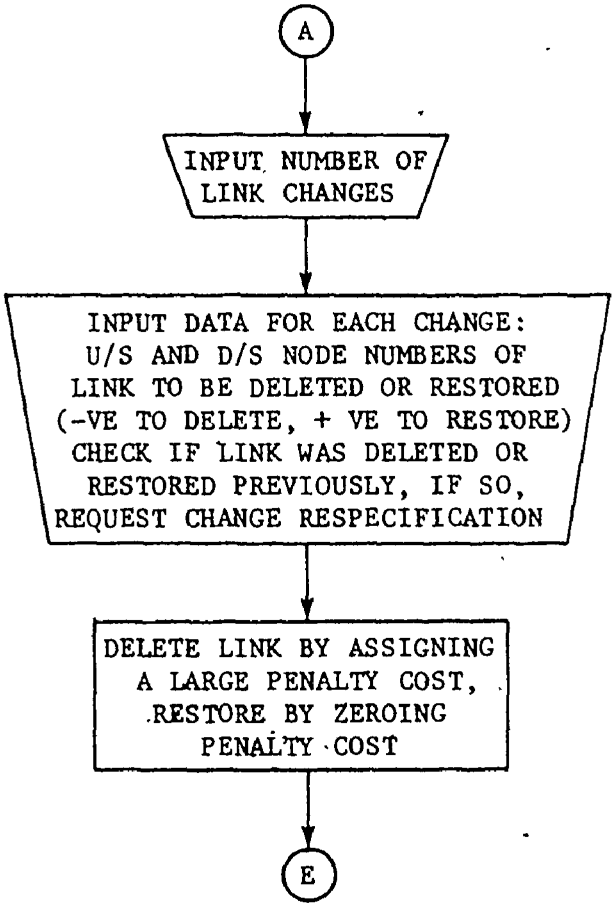
SUBROUTINE QINIT (cont.)



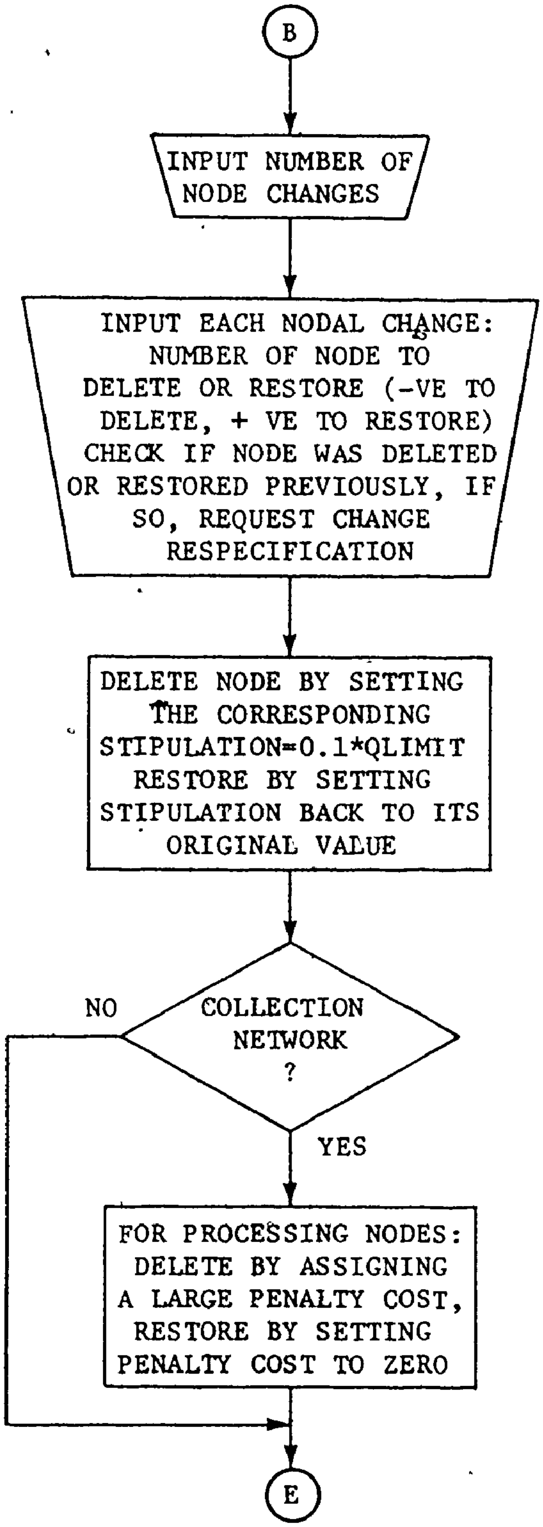
## SUBROUTINE MODIFY


START
DIMENSION ARRAYS DYNAMICALLY
PRINT ALLOWABLE COMMANDS:
GO: TO PROCEED
LINKS: TO DELETE OR RESTORE LINKS
NODES: TO DELETE OR RESTORE NODES
STIP: TO CHANGE STIPULATIONS
STATE: TO CHANGE NODAL STATES
BACK: TO GO BACK INTO "QINIT"
STOP: TO END RUN
INPUT COMMAND
COMMAND= GO?
YES
RETURN
NO
COMMAND= LINKS?
YES
A
NO
COMMAND= NODES?
YES
B
NO
COMMAND= STIP?
YES
C
NO
COMMAND= STATE?
YES
D
NO
COMMAND= BACK?
YES
SET FLAG TO GO BACK INTO QINIT
RETURN
NO
COMMAND= STOP?
YES
STOP
NO
E
COMMAND NOT RECOGI

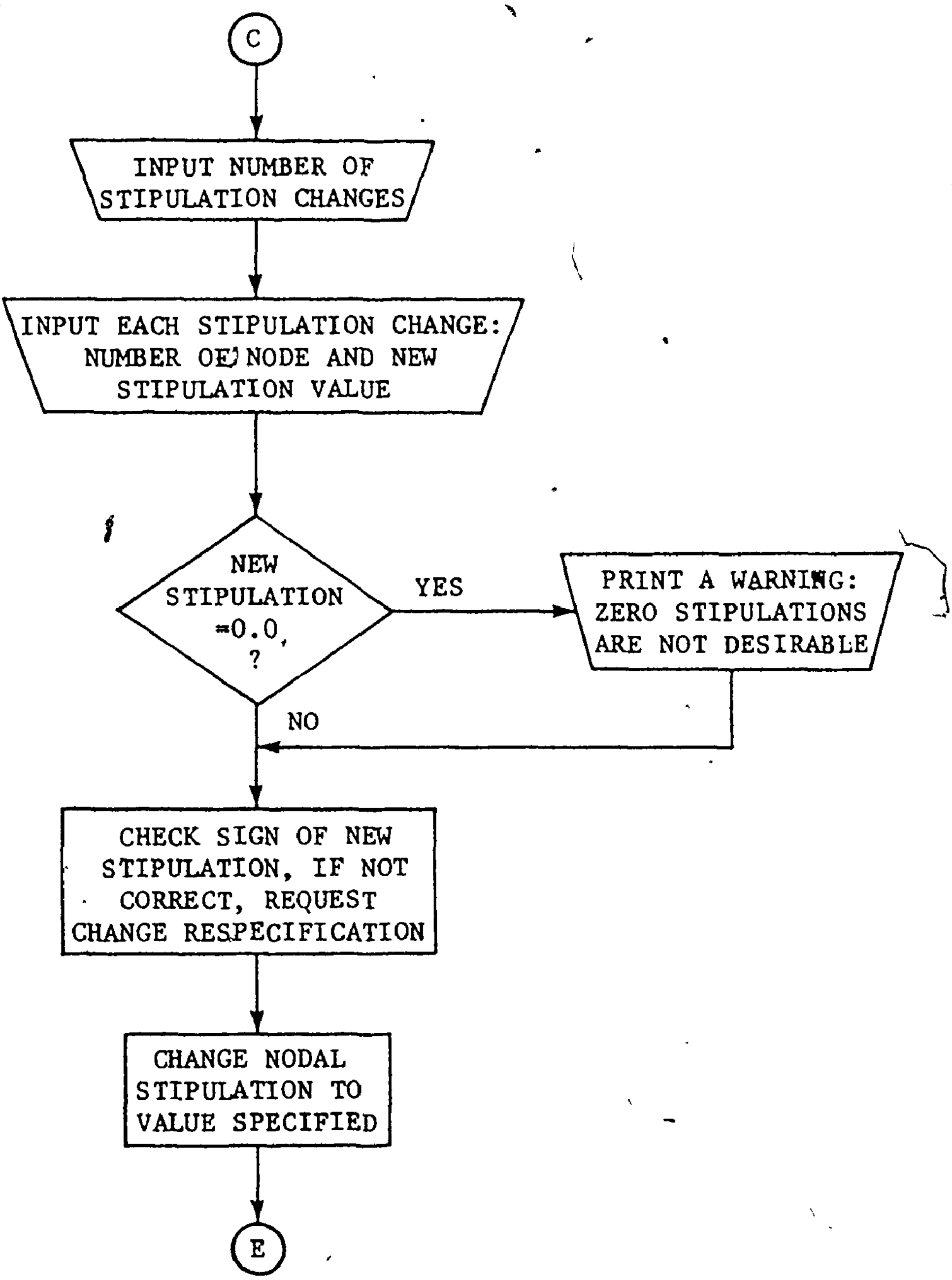
SUBROUTINE MODIFY (cont.)


A
INPUT NUMBER OF LINK CHANGES
INPUT DATA FOR EACH CHANGE: U/S AND D/S NODE NUMBERS OF LINK TO BE DELETED OR RESTORED (-VE TO DELETE, + VE TO RESTORE) CHECK IF LINK WAS DELETED OR RESTORED PREVIOUSLY, IF SO, REQUEST CHANGE RESPECIFICATION
DELETE LINK BY ASSIGNING A LARGE PENALTY COST, RESTORE BY ZEROING PENALTY COST
E

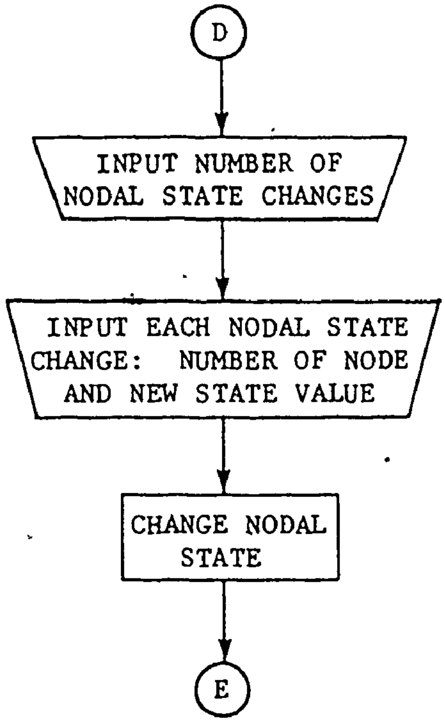
SUBROUTINE MODIFY (cont.)


B
INPUT NUMBER OF NODE CHANGES
INPUT EACH NODAL CHANGE: NUMBER OF NODE TO DELETE OR RESTORE (-VE TO DELETE, + VE TO RESTORE) CHECK IF NODE WAS DELETED OR RESTORED PREVIOUSLY, IF SO, REQUEST CHANGE RESPECIFICATION
DELETE NODE BY SETTING THE CORRESPONDING STIPULATION=0.1*QLIMIT RESTORE BY SETTING STIPULATION BACK TO ITS ORIGINAL VALUE
COLLECTION NETWORK ?
NO
YES
FOR PROCESSING NODES: DELETE BY ASSIGNING A LARGE PENALTY COST, RESTORE BY SETTING PENALTY COST TO ZERO
E

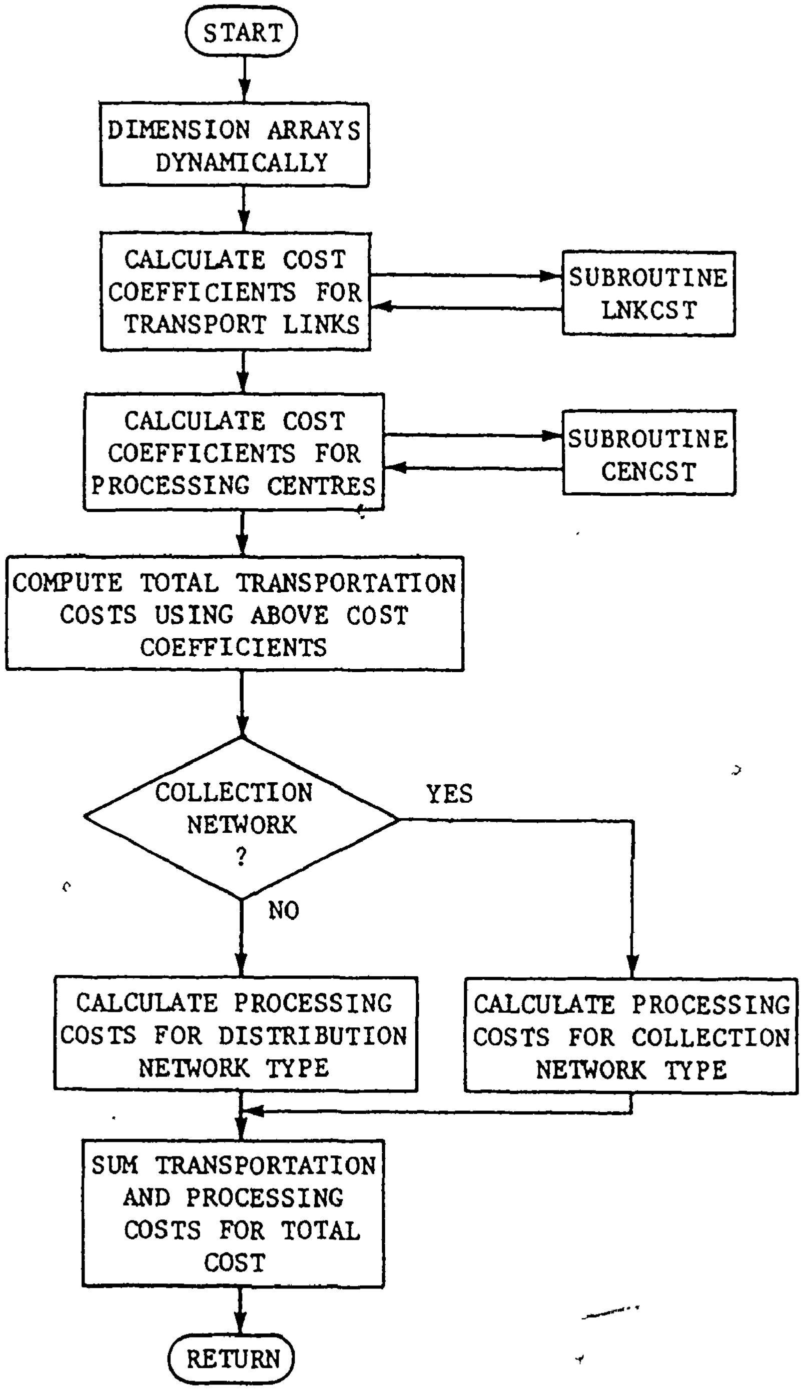
SUBROUTINE MODIFY (cont.)


C
INPUT NUMBER OF STIPULATION CHANGES
INPUT EACH STIPULATION CHANGE: NUMBER OF NODE AND NEW STIPULATION VALUE
NEW STIPULATION =0.0 ?
YES
PRINT A WARNING: ZERO STIPULATIONS ARE NOT DESIRABLE
NO
CHECK SIGN OF NEW STIPULATION, IF NOT CORRECT, REQUEST CHANGE RESPECIFICATION
CHANGE NODAL STIPULATION TO VALUE SPECIFIED
E

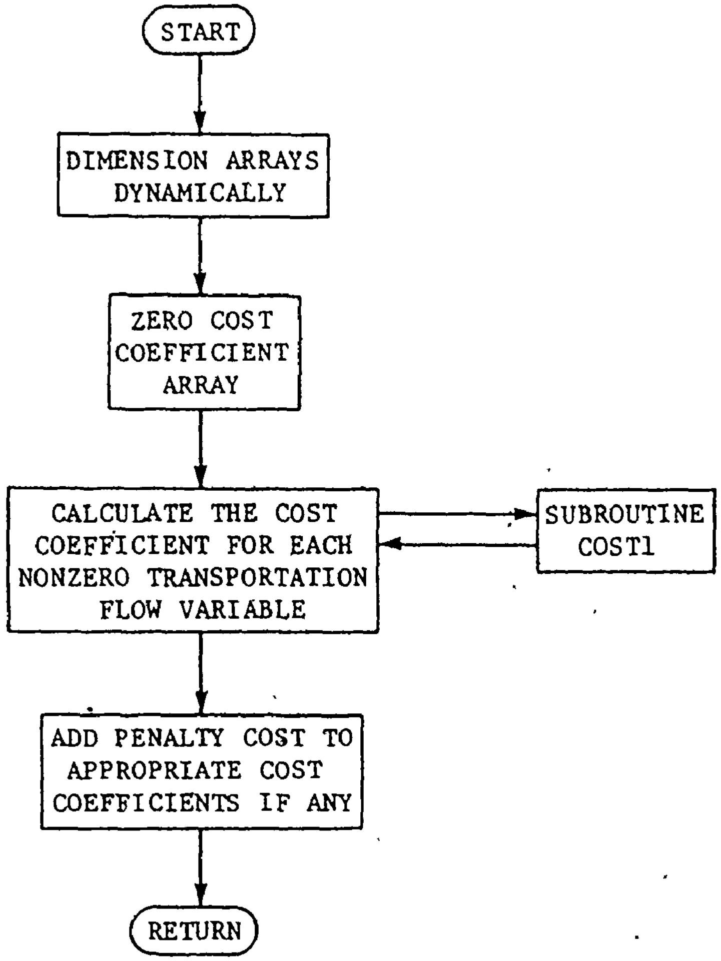
SUBROUTINE MODIFY (cont.)


D
INPUT NUMBER OF NODAL STATE CHANGES
INPUT EACH NODAL STATE CHANGE: NUMBER OF NODE AND NEW STATE VALUE
CHANGE NODAL STATE
E

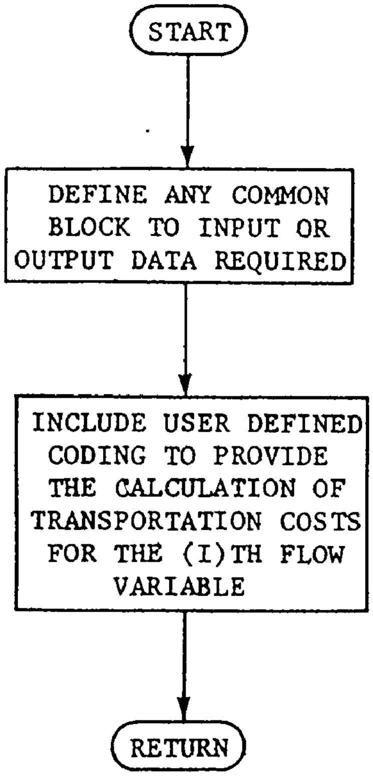
SUBROUTINE SOLCST


START
DIMENSION ARRAYS DYNAMICALLY
CALCULATE COST COEFFICIENTS FOR TRANSPORT LINKS
SUBROUTINE LNKCST
CALCULATE COST COEFFICIENTS FOR PROCESSING CENTRES
SUBROUTINE CENCST
COMPUTE TOTAL TRANSPORTATION COSTS USING ABOVE COST COEFFICIENTS
COLLECTION NETWORK ?
YES
NO
CALCULATE PROCESSING COSTS FOR DISTRIBUTION NETWORK TYPE
CALCULATE PROCESSING COSTS FOR COLLECTION NETWORK TYPE
SUM TRANSPORTATION AND PROCESSING COSTS FOR TOTAL COST
RETURN

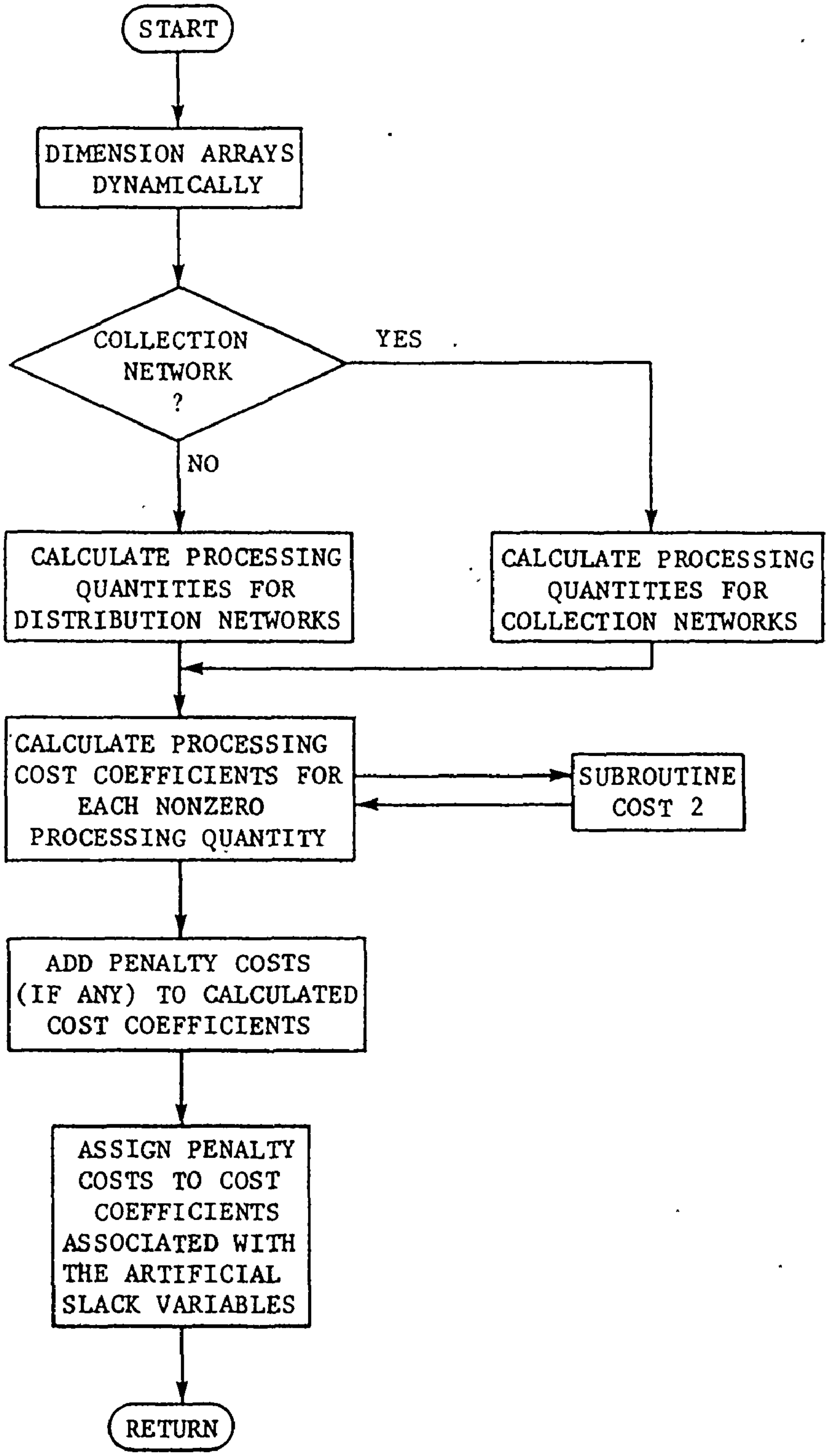
SUBROUTINE LNKCST


START
DIMENSION ARRAYS DYNAMICALLY
ZERO COST COEFFICIENT ARRAY
CALCULATE THE COST COEFFICIENT FOR EACH NONZERO TRANSPORTATION FLOW VARIABLE
SUBROUTINE COST1
ADD PENALTY COST TO APPROPRIATE COST COEFFICIENTS IF ANY
RETURN

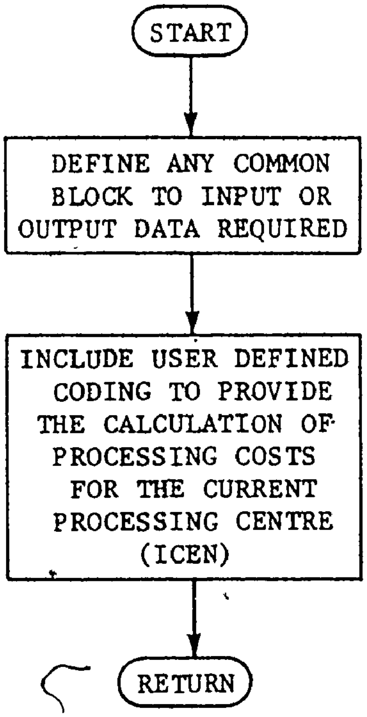
## SUBROUTINE COST1


START
DEFINE ANY COMMON BLOCK TO INPUT OR OUTPUT DATA REQUIRED
INCLUDE USER DEFINED CODING TO PROVIDE THE CALCULATION OF TRANSPORTATION COSTS FOR THE (I)TH FLOW VARIABLE
RETURN

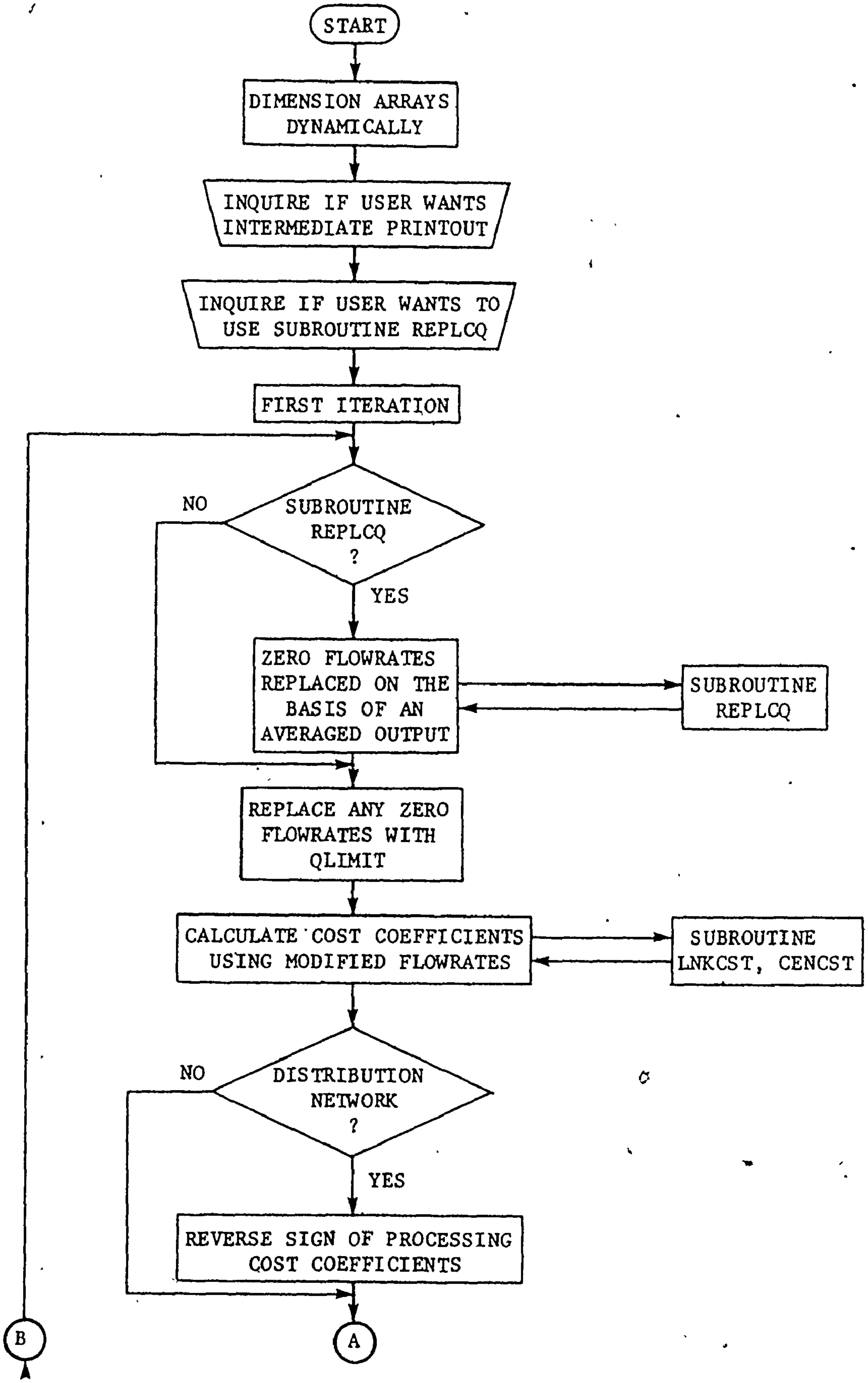
SUBROUTINE CENCST

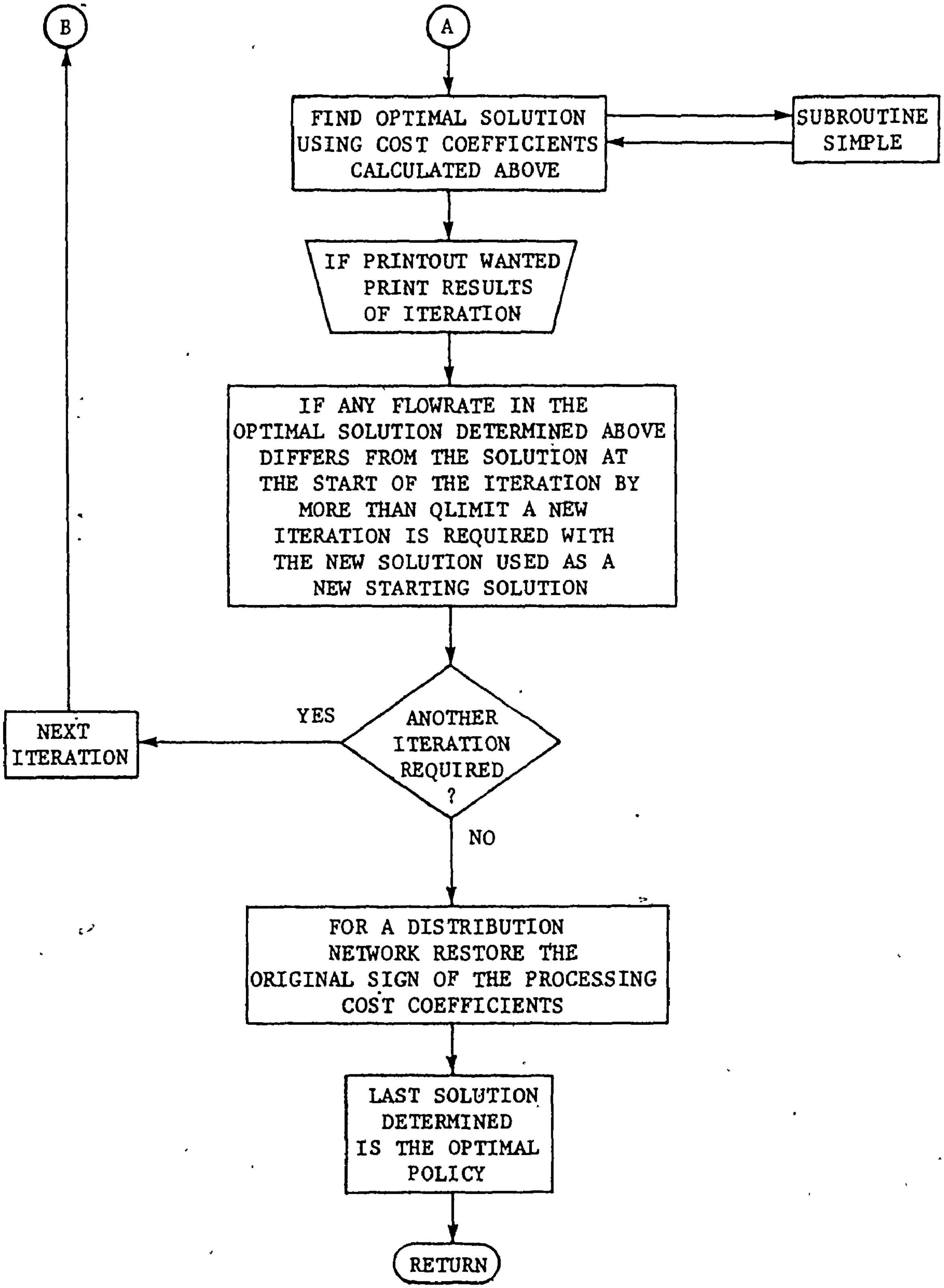

START
DIMENSION ARRAYS DYNAMICALLY
COLLECTION NETWORK ?
YES
NO
CALCULATE PROCESSING QUANTITIES FOR DISTRIBUTION NETWORKS
CALCULATE PROCESSING QUANTITIES FOR COLLECTION NETWORKS
CALCULATE PROCESSING COST COEFFICIENTS FOR EACH NONZERO PROCESSING QUANTITY
SUBROUTINE COST 2
ADD PENALTY COSTS (IF ANY) TO CALCULATED COST COEFFICIENTS
ASSIGN PENALTY COSTS TO COST COEFFICIENTS ASSOCIATED WITH THE ARTIFICIAL SLACK VARIABLES
RETURN

SUBROUTINE COST2

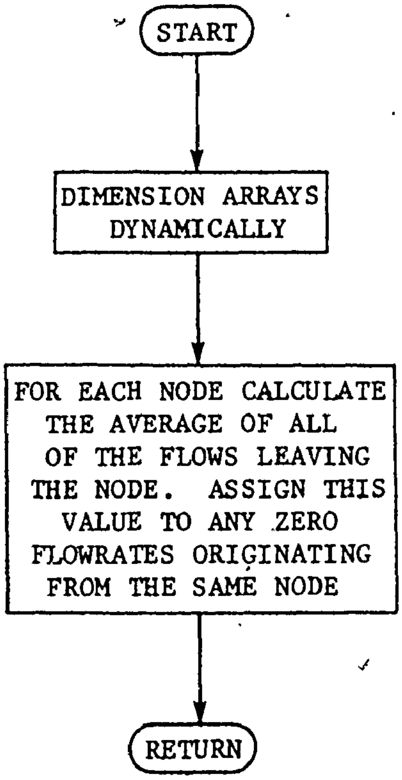

START
DEFINE ANY COMMON BLOCK TO INPUT OR OUTPUT DATA REQUIRED
INCLUDE USER DEFINED CODING TO PROVIDE THE CALCULATION OF PROCESSING COSTS FOR THE CURRENT PROCESSING CENTRE (ICEN)
RETURN

## SUBROUTINE SOLVE

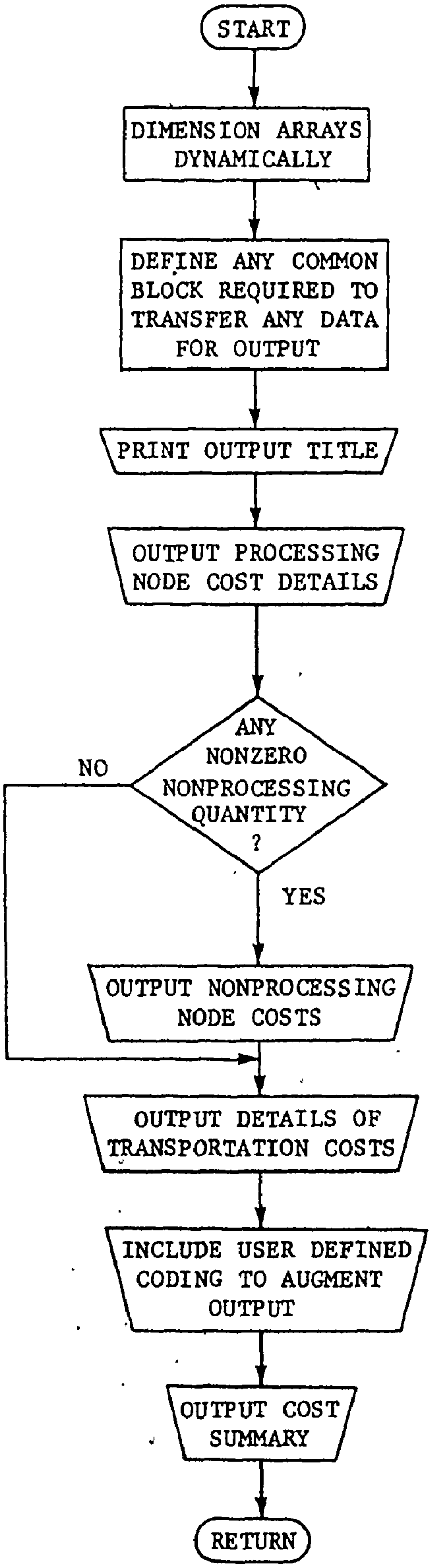

START
DIMENSION ARRAYS DYNAMICALLY
INQUIRE IF USER WANTS INTERMEDIATE PRINTOUT
INQUIRE IF USER WANTS TO USE SUBROUTINE REPLCQ
FIRST ITERATION
SUBROUTINE REPLCQ ?
NO
YES
ZERO FLOWRATES REPLACED ON THE BASIS OF AN AVERAGED OUTPUT
SUBROUTINE REPLCQ
REPLACE ANY ZERO FLOWRATES WITH QLIMIT
CALCULATE COST COEFFICIENTS USING MODIFIED FLOWRATES
SUBROUTINE LNKCST, CENCST
DISTRIBUTION NETWORK ?
NO
YES
REVERSE SIGN OF PROCESSING COST COEFFICIENTS
B
A


B
A
FIND OPTIMAL SOLUTION USING COST COEFFICIENTS CALCULATED ABOVE
SUBROUTINE SIMPLE
IF PRINTOUT WANTED PRINT RESULTS OF ITERATION
IF ANY FLOWRATE IN THE OPTIMAL SOLUTION DETERMINED ABOVE DIFFERS FROM THE SOLUTION AT THE START OF THE ITERATION BY MORE THAN QLIMIT A NEW ITERATION IS REQUIRED WITH THE NEW SOLUTION USED AS A NEW STARTING SOLUTION
ANOTHER ITERATION REQUIRED ?
YES
NEXT ITERATION
NO
FOR A DISTRIBUTION NETWORK RESTORE THE ORIGINAL SIGN OF THE PROCESSING COST COEFFICIENTS
LAST SOLUTION DETERMINED IS THE OPTIMAL POLICY
RETURN

SUBROUTINE REPLCQ


START
DIMENSION ARRAYS DYNAMICALLY
FOR EACH NODE CALCULATE THE AVERAGE OF ALL OF THE FLOWS LEAVING THE NODE. ASSIGN THIS VALUE TO ANY ZERO FLOWRATES ORIGINATING FROM THE SAME NODE
RETURN

SUBROUTINE REPORT


START
DIMENSION ARRAYS DYNAMICALLY
DEFINE ANY COMMON BLOCK REQUIRED TO TRANSFER ANY DATA FOR OUTPUT
PRINT OUTPUT TITLE
OUTPUT PROCESSING NODE COST DETAILS
ANY NONZERO NONPROCESSING QUANTITY ?
NO
YES
OUTPUT NONPROCESSING NODE COSTS
OUTPUT DETAILS OF TRANSPORTATION COSTS
INCLUDE USER DEFINED CODING TO AUGMENT OUTPUT
OUTPUT COST SUMMARY
RETURN

# APPENDIX E

## A USERS GUIDE FOR THE USE OF THE NETSOL PACKAGE

## ***ECONOMIC OPTIMIZATION OF NETWORK SYSTEMS***

### (1) PURPOSE

THE PACKAGE IS DESIGNED TO FACILITATE THE OPTIMIZATION, IN ECONOMIC TERMS, OF, A DISTRIBUTION OR COLLECTION NETWORK WHEREBY SOME SERVICE UTILITY CONCENTRATED AT ONE OR MORE CENTRAL SITES IS CONNECTED TO A NUMBER OF NODES REPRESENTING CONSUMERS OR CUSTOMERS OF THE SERVICE (E.G. WATER DISTRIBUTION OR THE COLLECTION AND TREATMENT OF WASTE WATER OR SOLID WASTE). THE NETWORK SYSTEM TO BE OPTIMIZED MUST BE DESCRIBED BY DEFINING THE SET OF POSSIBLE PROCESSING CENTRES AND CONSUMER NODES, QUANTIFYING AT EACH NODE THE PROCESSING CAPACITY AND CONSUMER LOAD RESPECTIVELY. IN ADDITION THE TOPOLOGY OF THE NETWORK MUST BE DEFINED BY LISTING ALL POSSIBLE LINKS WHICH MAY BE USED, TOGETHER WITH SOME MEASURE OF IMPEDANCE OR RESISTANCE TO FLOW. FINALLY, FOR EACH PROBLEM, THE USER MUST PROVIDE SUBROUTINES TO COMPUTE THE FOLLOWING COSTS.

(A) COST OF PROCESSING AT EACH CENTRAL SITE AS A FUNCTION OF THE TOTAL PROCESS FLOW AT THAT NODE.

(B). COST OF CONVEYING OR TRANSPORTING THE SERVICE TO THE CONSUMERS AS A FUNCTION OF THE VOLUME RATE OF FLOW AND WHICHEVER MEASURE OF IMPEDANCE MAY BE APPROPRIATE.

THE ROUTINE OPERATES ON THE NETWORK SYSTEM THUS DEFINED AND SELECTS THE PROCESSING SITES AND THE CONVEYANCE SCHEME WHICH MOST ECONOMICALLY ~~SATIFIES~~ THE CONSTRAINTS ON THE SYSTEM.

OUTPUT OF THE RESULTS MAY BE AUGMENTED BY THE USER TO INCLUDE QUANTITIES COMPUTED IN THE COURSE OF CALCULATING PROCESSING AND TRANSPORTATION COSTS.

SEE SECTION (4A) FOR FORMATING OF INPUT DATA.

### (2) METHOD

SUBROUTINE NETSOL IS AN ORGANIZATIONAL SUBPROGRAM WHICH CALLS, IN THE APPROPRIATE SEQUENCE, OTHER ROUTINES WHICH CARRY OUT SPECIFIC TASKS IN SETTING UP, MODIFYING AND SOLVING THE MATHEMATICAL MODEL OF THE NETWORK SYSTEM. TWO USER SUPPLIED ROUTINES, COST1 AND COST2, PROVIDE THE COST FUNCTIONS FOR TRANSPORTATION AND PROCESSING COSTS. THE COST FUNCTION MUST BE CONCAVE AND MONOTONIC, IN WHICH CASE IT CAN BE SHOWN THAT THE OPTIMAL SOLUTION LIES AT A VERTEX OF THE FEASIBLE SPACE DEFINED BY THE CONSTRAINTS OF THE NODES.

PRESENTLY THERE ARE TWO SOLUTION ALGORITHMS AVAILABLE WHICH WILL DETERMINE THE OPTIMAL SOLUTION. ONE SUCH ALGORITHM UTILIZES AN ITERATIVE LINEAR PROGRAMMING (ILP) APPROACH IN CONJUNCTION WITH A SIMPLEX LINEAR PROGRAMMING ROUTINE. THE ILP ALGORITHM PROVIDES A 'COARSE SEARCH' TECHNIQUE WHICH, IN PRACTICE, HAS PROVEN TO BE A FAST AND EFFICIENT METHOD (USUALLY 2 TO 4 ITERATIONS ARE REQUIRED). THE SOLUTION CONVERGED UPON BY THIS METHOD IS NOT NECESSARILY OPTIMAL, HOWEVER, IT IS GENERALLY QUITE CLOSE TO THE OPTIMAL SOLUTION IN COST. THE SECOND ALGORITHM, HYVRST, UTILIZES A MODIFIED SIMPLEX APPROACH DESIGNED SPECIFICALLY FOR PROBLEMS WITH CONCAVE COST FUNCTIONS AND LINEAR CONSTRAINTS. A REFINED DIRECT SEARCH METHOD IS USED IN HYVRST WHICH IS MORE ACCURATE THAN THE ILP ALGORITHM BUT CONSIDERABLY MORE EXPENSIVE COMPUTATIONALLY.

BECAUSE OF THE NATURE OF THE PROBLEM IT IS NOT POSSIBLE TO GUARANTEE THAT THE SOLUTION OBTAINED IS NOT A LOCAL MINIMUM. EXPERIENCE INDICATES THAT SUCH DIFFICULTIES ARE MORE LIKELY WITH VERY SMALL NETWORKS.

TO PROVIDE FACILITIES FOR TESTING THE SOLUTION AND FOR TESTING THE SENSITIVITY OF THE SOLUTION TO CHANGES IN THE NETWORK TOPOLOGY AND PROPERTIES, THE ROUTINES ARE DESIGNED PRIMARILY FOR USE IN A TIME-SHARING MODE, AND PROVIDE OPPORTUNITIES FOR ALTERING SYSTEM PARAMETERS IN THE COURSE OF EXECUTION. IN EACH MAJOR SUBROUTINE A SERIES OF COMMANDS ARE ALLOWED AS SUMMARISED BELOW. THE RESPONSE TO EACH INVITATION TO ENTER A COMMAND IS CHECKED FOR VALIDITY AND FEASIBILITY AND OPPORTUNITY AFFORDED FOR RE-ENTRY IN THE EVENT OF ERRORS BEING DETECTED. THE PACKAGE CONTAINS FOUR MAIN SECTIONS, EACH OF WHICH IS DESCRIBED BY PROMPT STATEMENTS.

(A) SUBROUTINE CNSTIN:

THIS ROUTINE PROVIDES ONE MEANS OF SPECIFYING PARAMETERS REQUIRED IN THE CALCULATION OF COSTS. AN OPTION IS PROVIDED WHEREBY THE INPUTS OR ASSIGNMENTS MAY BE BY-PASSED ON THE SECOND OR SUBSEQUENT RUNS. THE PROMPT STATEMENT IS...

```
DO YOU WISH TO REDEFINE ANY CONSTANTS
                                (YES/NO)...
```

TO WHICH THE RESPONSE 'NO' MUST BE GIVEN FOR THE BODY OF THE ROUTINE TO BE BY-PASSED. ANY OTHER RESPONSE RESULTS IN THE ROUTINE BEING EXECUTED.

SEE SECTION (4B) FOR EXAMPLE OF USER DEFINED CODING IN THIS ROUTINE.

(B) SUBROUTINE QINIT:
THIS ROUTINE PROVIDES OPPORTUNITES TO EXAMINE THE CURRENT VALUE OF NONZERO FLOW VARIABLES, OR TO ALTER THESE VALUES, OR TO OBTAIN A FEASIBILITY CHECK AND COST SOLUTION USING THE CURRENT FLOW VALUES. UPON ENTRY TO THE ROUTINE THE AVAILABLE COMMANDS ARE SUMMARISED.

GO TO EXIT FROM THE ROUTINE AND PROCEED TO THE NEXT STAGE, I.E. ROUTINE MODIFY.

STOP TO TERMINATE THE RUN.

LIST TO LIST THE NONZERO FLOWRATES CURRENTLY STORED IN THE PROGRAM, E.G.

```
 U/S NODE   D/S NODE    FLOW
    2          12       123.45
    ETC....
```

ALTER TO CHANGE SOME OR ALL OF THE CURRENTLY SPECIFIED FLOW VALUES. INITIALLY THE USER IS GIVEN THE OPTION OF ZEROING ALL FLOW VARIABLES. THE FIRST INPUT REQUIRED IS THE NUMBER OF FLOWRATES TO BE CHANGED (FORMAT I3). THE USER IS THEN REQUESTED TO DEFINE THE FLOW VARIABLE BY U/S AND D/S NODE AND FLOW VALUE (FORMAT 2I3,F15.5), EG.

```
 CHANGE NUMBER 3...   2 12   123.45
 CHANGE NUMBER 4...   ETC...
```

EACH ENTRY THUS MADE IS CHECKED FOR FEASIBILITY AND A NEW ENTRY REQUESTED IF A NON-EXISTANT FLOW VARIABLE IS DEFINED.

SOLVE TO OBTAIN A COST CALCULATION AND ANY RELEVANT DESIGN VALUES FOR THE CURRENTLY DEFINED SET OF FLOW VARIABLES. FOLLOWING THIS COMMAND THE SLACK VARIABLES ARE RE-CALCULATED AND A CHECK MADE FOR AN INFEASIBLE (I.E. -VE) VALUE. A FURTHER CONTINUITY CHECK IS MADE AT EACH CONSUMER NODE AND ANY IMBALANCE REPORTED. FINALLY THE COST COEFFICIENTS ARE COMPUTED AND A REPORT OF PROCESSING AND TRANSPORTATION COSTS OUTPUT BY ROUTINE REPORT.

(C) SUBROUTINE MODIFY:
THIS ROUTINE ALLOWS THE USER TO DELETE OR RESTORE LINKS AND NODES IN THE NETWORK OR TO ALTER THE STIPULATIONS (I.E. PROCESSING CAPACITY OR CONSUMER LOAD) AT ANY NODE. ONCE AGAIN, AT ENTRY TO THE ROUTINE THE AVAILABLE COMMANDS ARE SUMMARIZED AND ENTRY OF A COMMAND INVITED REPEATEDLY UNTIL TERMINATED BY EXIT FROM THE ROUTINE.

GO TO PROCEED TO THE NEXT STAGE IN THE PROGRAM.

STOP TO TERMINATE THE RUN.

LINKS TO DELETE OR SUBSEQUENTLY RESTORE LINKS FROM THE INITIAL NETWORK. THE FIRST INPUT REQUESTED IS THE NUMBER OF LINK CHANGES TO BE MADE (FORMAT I3). IF THIS

IS FEASIBLE EACH CHANGE IS REQUESTED AS FOLLOWS:

```
 SUPPLY U/S AND D/S NODE NOS, (FORMAT 2I3)
 (-VE TO DELETE, +VE TO RESTORE)
 CHANGE NO.  1   -2 12
 CHANGE NO.  2  ETC...
```

EACH ENTRY IS CHECKED FOR FEASIBILITY AND COMPATIBILITY WITH PREVIOUS COMMANDS. THE EFFECT OF LINK DELETION IS TO ADD AN ARBITRARILY HIGH PENALTY TO THE COST OF THE CORRESPONDING FLOW VARIABLE.

NODES TO DELETE OR SUBSEQUENTLY RESTORE NODES IN THE NETWORK. THE FIRST INPUT REQUESTED IS THE NUMBER OF CHANGES TO BE MADE (FORMAT I3). IF FEASIBLE, EACH CHANGE IS REQUESTED AS FOLLOWS.

```
 SUPPLY NODE NO, (FORMAT I3)
 (-VE TO DELETE, +VE TO RESTORE)
 CHANGE NO.  1   -10
 CHANGE NO.  2   ETC....
```

EACH ENTRY IS CHECKED FOR FEASIBILITY AND COMPATIBILITY WITH PREVIOUS COMMANDS. THE EFFECT OF NODE DELETION IS TO REDUCE THE STIPULATION TO AN INSIGNIFICANT AMOUNT. THE NODE MAY THEN STILL SERVE AS A JUNCTION. RESTORATION OF A NODE RESETS THE STIPULATION TO THE INITIAL VALUE.

STIP TO ALTER THE STIPULATION AT ONE OR MORE NODES. ONCE AGAIN THE FIRST INPUT REQUESTED IS THE NUMBER OF CHANGES TO BE MADE (FORMAT I3). IF FEASIBLE EACH CHANGE IS REQUESTED AS FOLLOWS.

```
 SUPPLY NODE NO. AND NEW STIPULATION
                         (FORMAT I3,F15.5)
 CHANGE NO.  1    12  -500.0
```

THE SIGNIFICANCE OF THE SIGN OF THE STIPULATION IS DISCUSSED IN SECTION (4A).

STATE TO ALTER THE STATE AT ONE OR MORE NODES. THE NUMBER OF STATE CHANGES IS FIRST REQUESTED AND THEN EACH CHANGE IS REQUESTED AS FOLLOWS.

```
 SUPPLY NODE NO. AND NEW STATE
                         (FORMAT I3,15.5)
 CHANGE NO.   1 10  20.0
```

AS WITH THE STIP COMMAND THE NODAL NUMBERS ARE CHECKED FOR FEASIBILITY UPON INPUT.

BACK TO RE-DIRECT COMPUTATION TO THE QINIT SUBROUTINE. UPON SPECIFICATION THE APPROPRIATE FLAG VARIABLE IS SET TO BE USED IN SUBROUTINE NETSOL.

(D) SUBROUTINE SOLVE:

THE MAIN OPTION AVAILABLE IN THE SOLUTION ROUTINES IS TO OBTAIN A PRINTOUT OF THE FLOW VALUES AND

COSTS DURING THE ITERATIONS OF THE SOLUTION TECHNIQUE. THE PROMPT TAKES THE FORM:

```
 IN ROUTINE 'SOLVE'...DO YOU WANT DETAILS OF
 COSTS DURING ITERATIONS (YES/NO)...
```

FURTHER OPTIONS ARE AVAILABLE IN THE ILP ALGORITHM WHICH ARE SPECIALIZED TO THAT ROUTINE. THE USER IS GIVEN THE OPTION OF USING A ROUTINE (REPLCQ) TO CALCULATE AN AVERAGED FLOWRATE TO BE USED AS A FLOW REPLACEMENT VALUE IN THE COMPUTATION OF COST COEFFICIENTS, IN PLACE OF THE STANDARD DEFAULT PARAMETER. IT IS RECOMMENDED THAT THE STANDARD VALUE BE USED. A FURTHER OPTION IS AVAILABLE TO ALLOW THE USER TO EXTEND THE LIMIT ON THE ALLOWABLE NUMBER OF ITERATIONS. IT IS ANTICIPATED THAT AS FURTHER SOLUTION ROUTINES ARE INCORPORATED IN THE PACKAGE EACH ROUTINE WILL HAVE OPTIONS AVAILABLE WHICH ARE DESIGNED SPECIFICALLY FOR THAT ALGORITHM.

ALTHOUGH INTENDED PRIMARILY FOR TIME-SHARING USE THE PROGRAM MAY BE USED IN BATCH MODE. OBVIOSLY ALL INPUT COMMANDS AND ASSOCIATED DATA MUST BE ERROR FREE AND THE FACILITIES FOR DATA CHECKING AND RE-ENTRY SO USEFUL IN TIME SHARING MODE WILL ALMOST CERTAINLY RESULT IN FAILURE IF THE INPUT DECK CONTAINS ERRORS.

(3) PROGRAM

(A) DECK NAME: NETSOL

(B) CALLING SEQUENCE:

```
CALL NETSOL(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,STIP,
            XL,WA-WG,L,N,NCEN,NQ,NR1,NR2,NW)
```

WHERE

A = ARRAY OF SIZE (N,NQ) TO HOLD THE STRUTTURAL COEFFS. COMPUTED BY ROUTINE DEFINE

B = ARRAY OF SIZE (N) TO HOLD WORKING VALUES OF NODE STIPULATIONS.

C = ARRAY OF SIZE (NQ) TO HOLD COMPUTED VALUES OF COST COEFFICIENTS.

CPEN = ARRAY OF SIZE (NQ) TO HOLD COMPUTED VALUES OF PENALTY COST COEFFICIENTS.

NAMES = ARRAY OF SIZE (5,N) CONTAINING A NAME OF UP TO 20 CHARACTERS FOR EACH NODE.

NDS = ARRAY OF SIZE (NQ) DEFINING THE DOWNSTREAM NODE NO. FOR EACH FLOW VARIABLE (ZERO FOR LAST N SLACK VARIABLES)

NUS = ARRAY OF SIZE (NQ) DEFINING THE UPSTREAM NODE NO. FOR EACH FLOW VARIABLE. (ZERO FOR THE LAST N SLACK VARIABLES)

Q = ARRAY OF SIZE (NQ) CONTAINING ON ENTRANCE THE INITIAL APPROXIMATIONS AND ON EXIT

THE OPTIMAL VALUES OF THE FLOWRATES. (INCLUDING SLACK VARIABLES)

STATE = ARRAY OF SIZE (N) DEFINING STATE (E.G ELEVATION) OF EACH NODE FOR USE IN CALCULATING TRANSPORTATION COSTS.

STIP = ARRAY OF SIZE (N) DEFINING YIELD (+VE) OR ABSTRACTION (-VE) OF EACH NODE.

XL = ARRAY OF SIZE (NQ) DEFINING THE LINK LENGTH FOR EACH FLOW VARIABLE. (MAY DIFFER IN D/S AND U/S DIRECTION)

WA-WG = WORKING ARRAYS FOR SOLUTION ROUTINE.

L = INTEGER NO. OF LINKS IN SYSTEM.

N = INTEGER NO. OF NODES.

NCEN = INTEGER NO. OF PROCESSING NODES IN NETWORK SYSTEM.

NQ = 2*L+N = NO. OF FLOW + SLACK VARIABLES

NR1 = INPUT FILE DEVICE FOR NETWORK DATA.

NR2 = INPUT FILE DEVICE FOR COMMANDS.

NW = OUTPUT FILE DEVICE.

(C) OUTPUT

APART FROM PROMPT AND DIAGNOSTIC STATEMENTS THE MAIN OUTPUT OF RESULTS IS ORGANIZED AND CONTROLLED BY SUBROUTINE REPORT. THIS ROUTINE CONTAINS STATEMENTS TO PRINT PROCESSING COSTS, TRANSPORTATION COSTS AND TOTALS, BUT MAY BE AUGMENTED BY THE USER TO OUTPUT ADDITIONAL INFORMATION AS REQUIRED. DESIGN QUANTITIES EVALUATED IN THE COURSE OF CALCULATING COSTS MAY BE TRANSFERRED INTO ROUTINE REPORT BY LABELLED COMMON BLOCK. SEE EXAMPLE FOR ILLUSTRATION.

(D) RESTRICTIONS

THE USER MUST PROVIDE COST FUNCTIONS TO CALCULATE PROCESSING COSTS AND TRANSPORTATION COSTS. SEE EXAMPLE (SECTION 4) FOR DETAILS OF THE SPECIFICATION AND CODING OF THESE ROUTINES.

(E) OTHER DECKS REQUIRED

DEFINE CNSTIN QINIT SOLCST<br>
MODIFY SOLVE LNKCST REPLCQ<br>
CENCST COST1 COST2<br>
REPORT SIMP SIMPLE

OTHER ROUTINES REQUIRED BY ROUTINES COST1 OR COST2.

ROUTINES COST1 AND COST2 ARE USER PROVIDED (SEE SECTION 4(B)). ROUTINES CNSTIN AND REPORT MAY BE AUGMENTED BY THE USER.

(4) EXAMPLE

(A) INPUT

INPUT OF DATA DESCRIBING THE NETWORK IS FROM A FILE CONSTRUCTED ACCORDING TO THE FOLLOWING FORMATS.

| RECORD TYPE | VARIABLE NAME | DESCRIPTION | FORMAT |
|---|---|---|---|
| 1 | N | TOTAL NUMBER OF NODES | I5 |
| | NCEN | NUMBER OF CENTRAL PROCESSING NODES (1.LE.NCEN.LE.N) | I5 |
| | L | TOTAL NUMBER OF LINKS | I5 |
| | | (NODE DATA - REPEAT RECORD TYPE 2 FOR EACH NODE, N RECORDS REQUIRED) | |
| 2 | I | NODE NUMBER (1.LE.I.LE.N)(NOTE 1) | I3,3X |
| | NAMES(I) | NODE NAME OF UP TO 20 CHARS. | 5A4 |
| | STATE(I) | STATE VARIABLE SUCH AS HEAD OR ELEVATION WHICH MAY AFFECT TRANSPORTATION COST.(NOTE 2) | F10.1 |
| | STIP(I) | MEASURE OF DEMAND OR YIELD ASSOCIATED WITH THE NODE (+VE FOR INFLOW, -VE FOR ABSTRACTION) | F10.1 |
| | | (LINK DATA - REPEAT RECORD TYPE 3 FOR EACH LINK, L RECORDS REQUIRED) | |
| 3 | NUS( ) | NODE NUMBER AT 'UPSTREAM' END OF LINK (NOTE 3) | I5 |
| | NDS( ) | NODE NUMBER AT 'DOWNSTREAM' END OF LINK | I5 |
| | XL( ) | LINK LENGTH IN DOWNSTREAM DIRECTION (NOTE 4) | F10.1 |
| | XL( ) | LINK LENGTH IN UPSTREAM DIRECTION (NOTE 5) | F10.1 |
| | Q( ) | INITIAL FLOW APPROXIMATION IN DOWNSTREAM DIRN. (NOTE 6) | F10.1 |
| | Q( ) | INITIAL FLOW APPROXIMATION IN UPSTREAM DIRN. (NOTE 6) | F10.1 |

NOTE (1) THE PROCESSING NODES MUST BE NUMBERED BEFORE NONPROCESSING NODES. ALL NODE NUMBERS MUST BE CONSECUTIVE STARTING WITH 1. THE NODAL DATA CAN APPEAR IN THE DATA FILE IN ANY ORDER.

NOTE (2) THE ECONOMIC MODEL IS DIMENSIONALESS. HOWEVER, SPECIFIC UNITS MAY BE REQUIRED IN THE USER DEFINED COST ROUTINES.

NOTE (3) THE CHOICE OF UPSTREAM AND DOWNSTREAM DIRECTION IS ARBITRARY AND HAS NO BEARING ON THE FINAL SOLUTION.

NOTE (4) THE LENGTH MAY BE ANY APPROPRIATE MEASURE UPON WHICH TRANSPORTATION COST DEPENDS.

NOTE (5) LINK LENGTH WILL USUALLY BE THE SAME IN BOTH DIRECTIONS. A ZERO READING (I.E. EMPTY FIELD) FOR THE SECOND OR UPSTREAM LENGTH IS TAKEN TO MEAN THAT THE TWO ARE EQUAL.

NOTE (6) THE INITIAL APPROXIMATIONS MAY BE LEFT AS ZERO (OR EMPTY FIELDS).

THE FOLLOWING DATA FILE IS SET UP FOR A DISTRIBUTION NETWORK OF 13 NODES (INCLUDING 4 PROCESSING CENTRES) AND 21 LINKS.

```
  13    4   21
 1   CLEARWATER LAKE       430.0      +10.70
 5   THIRSTVILLE           275.0       -5.00
 6   BRACKWELL             330.0       -0.73
 2   CRYSTAL CREEK         370.0       +4.30
10   FOSSETTVILLE          290.0       -2.00
11   DRYGULCH              395.0       -3.00
 4   DEAD MAN)S POND       360.0       +6.50
12   JUNCTION ONE          425.0        0.00
13   JUNCTION TWO          400.0        0.00
 3   SULPHUR SPRINGS       525.0       +3.00
 7   NEWTOWN               360.0       -8.00
 8   BOGTON PARK           385.0       -1.20
 9   SALTWELL FLATS        425.0       -1.50
   1   12   23750.0
  12    6   15850.0
   6    1   26400.0
  12   11   12670.0
  11    6   16370.0
  11    3   10030.0
   3    6   28500.0
   3   13   24290.0
  13    6   13200.0
   2   13   21120.0
  13    5    9000.0
   5    6   16900.0
   5    7   14800.0
   7    6   25870.0
   7    8   18000.0
   8    9   22700.0
   9    7   34300.0
   8   10   19000.0
  10    9   24800.0
   8    4   29000.0
   4   10   32700.0
```

(B) CODE

A CERTAIN AMOUNT OF CODING IS REQUIRED BY THE MODEL FOR EACH APPLICATION TO A NEW NETWORK PROBLEM. THE MODEL ARRAYS MUST BE DIMENSIONED ACCORDING TO THE NETWORK SIZE. THE CNSTIN ROUTINE MUST BE SET UP TO INPUT PARAMETERS FOR THE COST ROUTINE AND THE COST ROUTINES CONSTRUCTED. THE ROUTINES REQUIRING CODING BY THE USER ARE SHOWN BELOW.

```
C
C     MAIN PROGRAM
C     DRIVER PROGRAM FOR NETWORK OPTIMIZATION.
C     READS PROBLEM SIZE FROM TAPE NR1.
C     PRINTS OUT THE PROGRAM TITLE.
C      DIMENSIONED ACCORDING TO PROBLEM SIZE.
C      N    = THE TOTAL NUMBER OF NODES.
C      NCEN = THE NUMBER OF PROCESSING NODES.
C      L    = NUMBER OF LINKS.
C      NQ   = THE NUMBER OF FLOW PLUS SLACK VARIABLES.
C             (2*L+N)
C
C      ARRAY SIZE SPECIFICATION.....
C      (N,NQ)..A
C      (5,N)...NAMES
C      (NQ)....C,CPEN,NDS,NUS,Q,XL,
C      (N).....B,STATE,STIP
C
C     ARRAYS STANDARD TO NETWORK PACKAGE.
C
      DIMENSION A(13,55),B(13),C(55),CPEN(55),NAMES(5,13)
      DIMENSION NDS(55),NUS(55),Q(55),STATE(13),STIP(13),XL(55)
C
C     WORKING ARRAYS FOR OPTIMIZATION PACKAGE.
C
      DIMENSION WA(13,13),WB(13),WC(13),WD(13)
      DIMENSION WE(13),WF(13),WG(55)
      INTEGER WF
C
C     DEFINE INPUT AND OUTPUT FILE NUMBERS.
C      NR1 = DATA FILE
C      NR2 = KEYBOARD INPUT DEVICE
C      NW  = OUTPUT DEVICE
C
      NR1=1
      NR2=5
      NW=6
      REWIND NR1
      READ(NR1,10)N,NCEN,L
 10   FORMAT(3I5)
      NQ=2*L+N
      WRITE(NW,20)
 20   FORMAT(///10X,' NETWORK SYSTEM OPTIMIZATION PROGRAM'/
     +       10X,' -----------------------------------'/
     +20X,' BY  A A SMITH'/
     +20X,' AND R H TUFGAR'////)
      CALL NETSOL(A,B,C,CPEN,NAMES,NDS,NUS,Q,STATE,STIP,XL,
     +            WA,WB,WC,WD,WE,WF,WG,L,N,NCEN,NQ,NR1,NR2,NW)
      STOP
      END
      SUBROUTINE CNSTIN(NR2,NW,QLIMIT)
C
C    ***********************************************************
C     USER AUGMENTED SUBROUTINE.
C     THIS SUBROUTINE ENABLES THE USER TO INPUT ANY CONSTANTS
C     REQUIRED IN THE COST SUBROUTINE.
C
C     NR2    = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW     = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     QLIMIT = LIMITING FLOWRATE VALUE. VALUES LESS THAN
C              QLIMIT ARE INSIGNIFICANT TO THE PROBLEM.
C    ***********************************************************
C
C     INSERT ANY LABELLED COMMON BLOCK STATEMENTS HERE.....
C
      QLIMIT=0.001
      DATA YES/1HY/
C
C     START OF USER SUPPLIED CONSTANT EVALUATION.....
C       USER SUPPLIES CODING HERE TO DEFINE ANY CONSTANTS.
C     END OF USER SUPPLIED CONSTANT EVALUATON.....
C
C     OPTION TO BYPASS REDEFINITION OF ANY CONSTANTS.
C
      WRITE(NW,10)
 10   FORMAT(/' IN SUBROUTINE "CNSTIN" '/
     +        ' DO YOU WISH TO REDEFINE ANY CONSTANTS'/
     +        '                          (YES/NO)...')
      READ(NR2,20)ANS
 20   FORMAT(A1)
      IF(ANS.NE.YES)RETURN
C
C     START OF USER DEFINED CODING TO REDEFINE CONSTANTS.....
C       USER SUPPLIES CODING HERE IF CONSTANTS ARE TO BE CHANGED
C     END OF USER DEFINED CODING TO REDEFINE CONSTANTS.....
C
      RETURN
      END
      SUBROUTINE COST1(COST,I,QI,STDS,STUS,XLI)
C
C    ************************************************************
C     USER SUPPLIED COST ROUTINE*****
C     COMPUTES THE TOTAL TRANSPORTATION COST FOR THE (I)TH FLOW
C     VARIABLE.
C     ROUTINE SHOULD CONTAIN ANY RELEVANT LABELLED COMMON
C     BLOCK STORAGE USED IN ROUTINES "CNSTIN" AND "REPORT"
C     FOR EITHER INPUT OF PARAMETERS OR RETURN OF DESIGN
C     PARAMETERS.
```

```
C
C     COST    = COMPUTED TRANSPORTATION COST FOR Q(I).
C     I       = FLOW VARIABLE IDENTIFICATION NUMBER.
C     QI      = CURRENT VALUE OF FLOWRATE Q(I).
C     STDS    = STATE AT DOWNSTREAM NODE OF Q(I).
C     STUS    = STATE AT UPSTREAM NODE OF Q(I).
C     XLI     = LENGTH OF LINK FOR Q(I).
C   ************************************************************
C
C     START OF USER PROVIDED CODING*****
C     USER SUPPLIES THE CODING NEEDED TO PROVIDE THE COST OF
C     TRANSPORTATION FOR THE (I)TH FLOW VARIABLE HAVING THE
C     CURRENT VALUE QI. THIS MAY REQUIRE THE CODING OF DESIGN
C     COMPUTATIONS. ANY SPECIFIC CONSTANTS OR VARIABLES
C     REQUIRED FOR INPUT OR OUTPUT MAY BE TRANSFERRED THROUGH
C     COMMON BLOCK.
C
C     COST=.....
C
C     END OF USER PROVIDED CODING*****
      RETURN
      END
      SUBROUTINE COST2(COST,ICEN,QI)
C
C   ************************************************************
C     USER SUPPLIED SUBROUTINE*****
C     COMPUTES THE TOTAL PROCESSING COST FOR A SPECIFIED
C     PROCESSING CENTRE AND FLOW QI. ROUTINE SHOULD CONTAIN
C     ANY RELEVANT LABELLED COMMON BLOCK STORAGE USED IN ROUTINES
C     "CNSTIN" AND "REPORT" FOR EITHER INPUT OF PARAMETERS OR
C     RETURN OF DESIGN PARAMETERS.
C
C     COST    = COMPUTED PROCESSING COST FOR NODE ICEN.
C     ICEN    = NUMBER OF PROCESSING NODE BEING COSTED.
C     QI      = VOLUME RATE OF FLOW BEING PROCESSED.
C   ************************************************************
C
C     START OF USER PROVIDED CODING*****
C     THE USER PROVIDES CODING HERE TO PROVIDE THE COST OF
C     PROCESSING AT THE RATE, QI, FOR THE CENTRE, ICEN.
C     ANY RELEVANT DESIGN DATA SHOULD BE CALCULATED HERE AND
C     AND TRANSFERRED TO REPORT THROUGH COMMON BLOCK.
C
C     COST=.....
C
C     END OF USER PROVIDED CODING*****
      RETURN
      END
```

(C) OUTPUT

THE OUTPUT FROM A TYPICAL RUN USING THE ILP ALGORITHM AND THE DATA FILE SHOWN ABOVE IS INCLUDED.

```
                NETWORK SYSTEM OPTIMIZATION PROGRAM
                -----------------------------------
                          BY  A A SMITH
                          AND R H TUFGAR



IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                             (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO

IN ROUTINE "SOLVE"...DO YOU WANT DETAILS
OF COSTS DURING ITERATIONS. (YES/NO)...NO


DO YOU WISH TO USE THE "REPLCQ" SUBROUTINE
                               (YES/NO)...NO

ITERATION  1
ITERATION  2
ITERATION  3
```

```
                     PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                     FLOW        COST (MIL)

 CLEARWATER LAKE                  9.43           .54
 CRYSTAL CREEK                    4.30           .30
 SULPHUR SPRINGS                  3.00           .23
 DEAD MAN'S POND                  4.70           .32


  TRANSPORTATION COSTS
 NODE          ROUTE               FLOW      COST (MIL)

   1  CLEARWATER LAKE       TO
    6 BRACKWELL                    9.43          1.23
   3  SULPHUR SPRINGS       TO
   11 DRYGULCH                     3.00           .21
   2  CRYSTAL CREEK         TO
   13 JUNCTION TWO                 4.30           .76
  13  JUNCTION TWO          TO
    5 THIRSTYVILLE                 4.30           .20
   6  BRACKWELL             TO
    5 THIRSTYVILLE                  .70           .21
   6  BRACKWELL             TO
    7 NEWTON                       8.00          1.31
   8  BOGTON MOOR           TO
    9 SALTWELL FLATS               1.50           .46
   4  DEAD MAN'S POND       TO
    8 BOGTON MOOR                  2.70           .79
   4  DEAD MAN'S POND       TO
   10 FOSSETTVILLE                 2.00           .72


         PROCESS COST...... (MILLION)     1.384
         TRANSPN COST...... (MILLION)     5.882
         TOTAL SYSTEM ..... (MILLION)     7.266
  END OF REPORT



 IN SUBROUTINE "CNSTIN"
 DO YOU WISH TO REDEFINE ANY CONSTANTS
                                      (YES/NO)...NO
```

```
IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP

 END OF PROGRAM
:
```

(5) DISCUSSION

THIS PACKAGE IS UNDER CONTINUED DEVELOPMENT AND MAY HAVE ADDED FEATURES INTRODUCED. CONSULT THE LATEST DOCUMENTATION.

(6) SOURCE

ALAN A. SMITH AND
RAY H. TUFGAR
MCMASTER UNIVERSITY
HAMILTON ONTARIO.

# APPENDIX F

## EXAMPLE USING THE ITERATIVE LINEAR PROGRAMMING ALGORITHM

This example is presented to demonstrate a typical application of the model using the ILP optimization routine. The distribution network used consists of a 13 node regional water supply system with 4 processing (source) nodes, 7 consumer nodes and 2 null, or junction, nodes. The network layout is illustrated graphically in figure F.1. There are 21 feasible links connecting the nodal points. The problem objective is to determine from which sources the demands should be satisfied and which routes should be used to convey the water to provide an optimal system.

The nodal data is provided in table F.1. The state variables represent the pressure elevation available at the processing nodes or required at the consumer nodes. The stipulation values are the nodal processing capacity or demand values of the consumer nodes. Table F.2 supplies the link definitions and the length of each link. The corresponding data file for the network system is illustrated in the documentation of Appendix E (see page 333).

To apply the econometric model, it is necessary to construct cost routines to calculate the transportation and processing costs. For purposes of illustration, it is assumed that the processing costs are adequately described by an exponential relation of the form:

$$CP = F \cdot a\,Q^{b} \qquad (F.1)$$

where a and b are coefficients which yield a concave function, Q is the processing flowrate and the factor, F, defines the effect of staging or postponing plant construction. In this sample problem, the following relation is used for all processing nodes:

$$CP = 100000.0 \cdot Q^{0.75} \qquad (F.2)$$

(It is a simple matter to make the coefficient a and b dependent on the processing node being considered.)

The transportation costs are defined by an equally simple relation which is used in an earlier example (Equation 3.3).

$$CT = 15.0 \cdot XL \cdot Q^{\frac{1}{2}} + 200.0 \cdot Q(.004 \cdot XL + STDS - STUS) \qquad (F.3)$$

where: XL = length (ft.)

Q = flowrate (MGD)

STDS = downstream elevation (ft.)

STUS = upstream elevation (ft.)

The cost functions used in this example represent the costs adequately although they do not represent any real life network system. More complex design calculations would normally be required, involving additional variables and auxiliary subroutines.

Subroutines COST1 and COST2 are consructed to include the above cost functions. No additional input is required through CNSTIN, therefore no user supplied portion is added. The output from REPORT is not augmented and this routine also remains unchanged.

Two different computer runs are used in this problem, the difference lying in the specification of an initial solution. Figure F.2(a) includes the output without an initial solution and figure F.2 (b) illustrates a run with an initial solution specified. In each example, the minimum of options are exercised since the model is only tested to illustrate convergence properties and typical execution time.

The execution times are very similar in the two sample runs, it takes three iterations and 3.129 cp seconds (CDC 6400) to convergence when no initial solution is specified and two iterations and 2.330 cp seconds to convergence when an initial solution is specified. The model, unfortunately, does not converge to the same solution in each case but the cost difference is only 0.4% and both solutions are very close to the optimal solution as found later in Appendix J by a different nonlinear optimization technique.

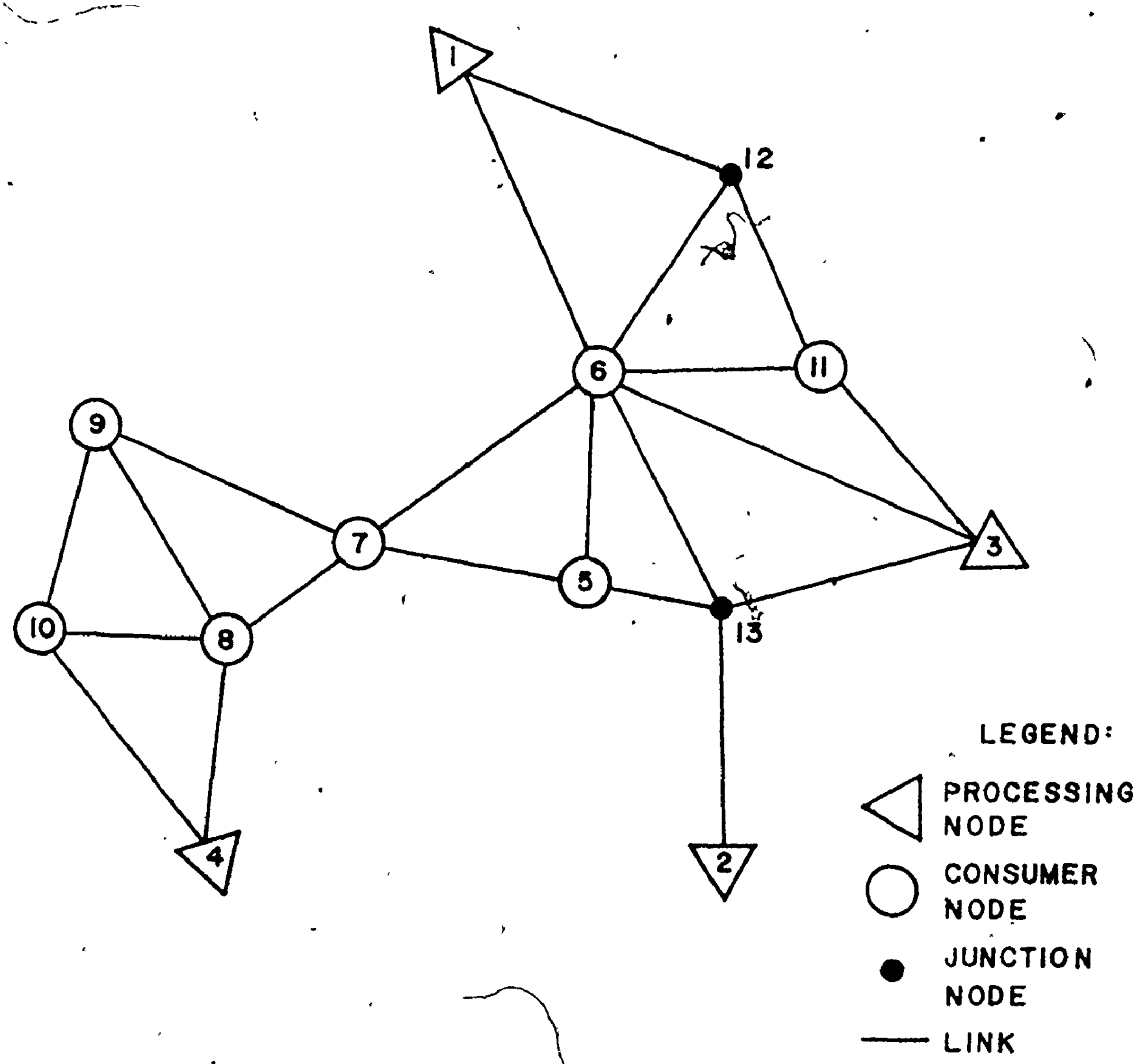
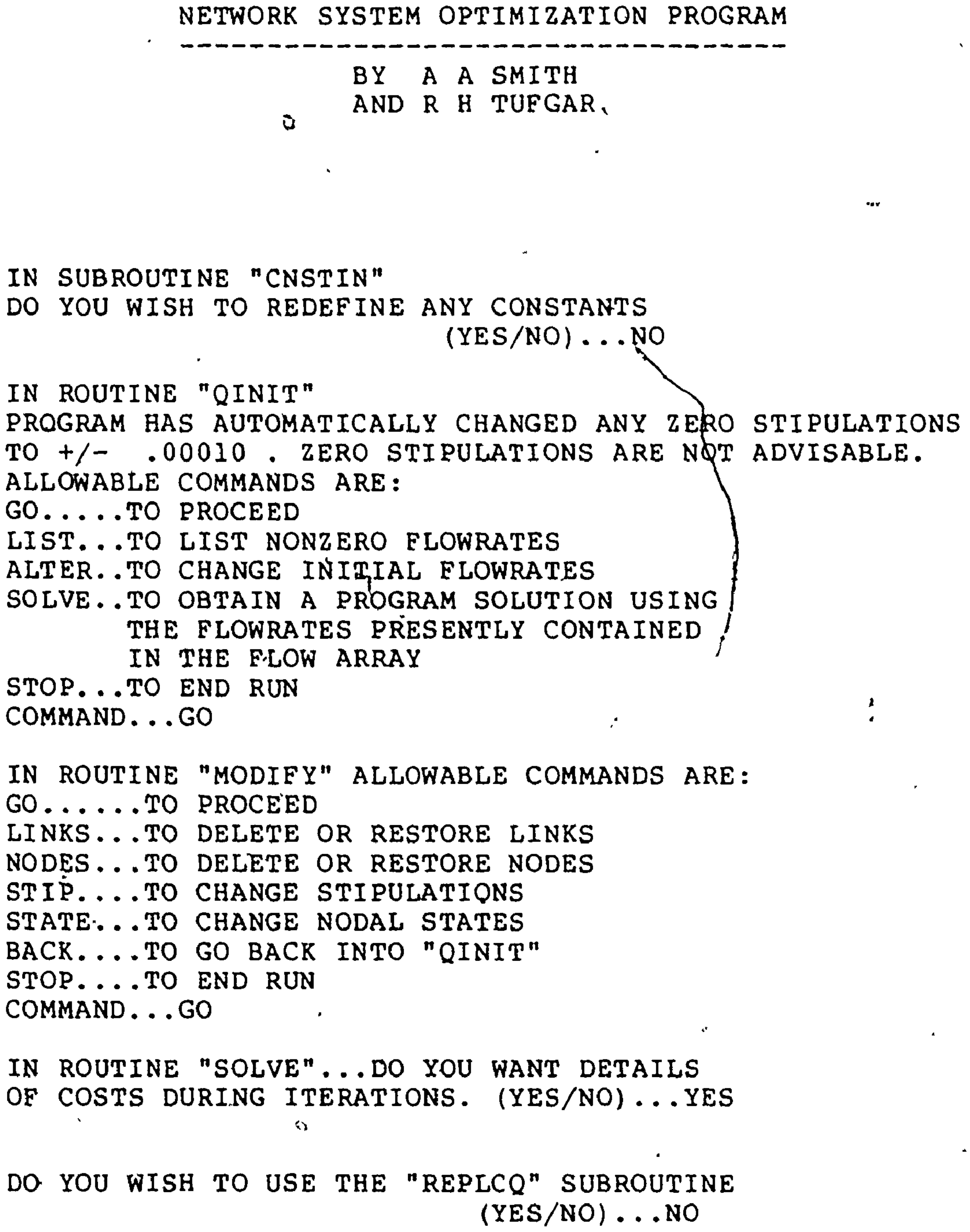


FIGURE F·I — DISTRIBUTION NETWORK FOR SAMPLE PROBLEM

FIGURE F·2(a) - SAMPLE COMPUTER RUN - NO INITIAL SOLUTION SPECIFIED



```
                    NETWORK SYSTEM OPTIMIZATION PROGRAM
                    -----------------------------------
                              BY  A A SMITH
                              AND R H TUFGAR



IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                         (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...GO

IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO

IN ROUTINE "SOLVE"...DO YOU WANT DETAILS
OF COSTS DURING ITERATIONS. (YES/NO)...YES


DO YOU WISH TO USE THE "REPLCQ" SUBROUTINE
                           (YES/NO)...NO
```

## FIGURE F·2(a) – (contd)

```
ITERATION  1
Q( 1) FROM NODES   1 TO  12         .0  11283618.0
Q( 2) FROM NODES  12 TO   1         .0  11285618.0
Q( 3) FROM NODES  12 TO   6         .0   7511997.0
Q( 4) FROM NODES   6 TO  12         .0   7549997.0
Q( 5) FROM NODES   6 TO   1         .0  12563742.0
Q( 6) FROM NODES   1 TO   6        7.6  12523742.0
Q( 7) FROM NODES  12 TO  11         .0   6014046.0
Q( 8) FROM NODES  11 TO  12         .0   6026046.0
Q( 9) FROM NODES  11 TO   6         .0   7765071.0
Q(10) FROM NODES   6 TO  11         .0   7791071.0
Q(11) FROM NODES  11 TO   3         .0   4791672.0
Q(12) FROM NODES   3 TO  11        3.0   4739672.0
Q(13) FROM NODES   3 TO   6         .0  13502542.0
Q(14) FROM NODES   6 TO   3         .0  13580542.0
Q(15) FROM NODES   3 TO  13         .0  11516194.0
Q(16) FROM NODES  13 TO   3         .0  11566194.0
Q(17) FROM NODES  13 TO   6         .0   6257871.0
Q(18) FROM NODES   6 TO  13         .0   6285871.0
Q(19) FROM NODES   2 TO  13        4.3  10040994.0
Q(20) FROM NODES  13 TO   2         .0  10028994.0
Q(21) FROM NODES  13 TO   5        4.3   4251276.0
Q(22) FROM NODES   5 TO  13         .0   4301276.0
Q(23) FROM NODES   5 TO   6         .0   8040895.0
Q(24) FROM NODES   6 TO   5         .7   8018895.0
Q(25) FROM NODES   5 TO   7         .0   7049098.0
Q(26) FROM NODES   7 TO   5         .0   7015098.0
Q(27) FROM NODES   7 TO   6         .0  12285916.0
Q(28) FROM NODES   6 TO   7        6.2  12297916.0
Q(29) FROM NODES   7 TO   8         .0   8557552.0
Q(30) FROM NODES   8 TO   7        1.8   8547552.0
Q(31) FROM NODES   8 TO   9        1.5  10793718.0
Q(32) FROM NODES   9 TO   8         .0  10777718.0
Q(33) FROM NODES   9 TO   7         .0  16284362.0
Q(34) FROM NODES   7 TO   9         .0  16310362.0
Q(35) FROM NODES   8 TO  10         .0   9008692.0
Q(36) FROM NODES  10 TO   8         .0   9046692.0
Q(37) FROM NODES  10 TO   9         .0  11810516.0
Q(38) FROM NODES   9 TO  10         .0  11756516.0
Q(39) FROM NODES   8 TO   4         .0  13774110.0
Q(40) FROM NODES   4 TO   8        4.5  13784110.0
Q(41) FROM NODES   4 TO  10        2.0  15523136.0
Q(42) FROM NODES  10 TO   4         .0  15551136.0
Q(43) FROM NODES   0 TO   0        3.1          .0
Q(44) FROM NODES   0 TO   0         .0          .0
Q(45) FROM NODES   0 TO   0         .0          .0
Q(46) FROM NODES   0 TO   0         .0          .0

OBJECTIVE FUNCTION VALUE=    377745984.0
```

# FIGURE F·2(a) — (contd)

```
ITERATION  2
Q( 1)  FROM  NODES    1  TO   12            .0   11283618.0
Q( 2)  FROM  NODES   12  TO    1            .0   11285618.0
Q( 3)  FROM  NODES   12  TO    6            .0    7511997.0
Q( 4)  FROM  NODES    6  TO   12            .0    7549997.0
Q( 5)  FROM  NODES    6  TO    1            .0   12563742.0
Q( 6)  FROM  NODES    1  TO    6           9.4     144480.8
Q( 7)  FROM  NODES   12  TO   11            .0    6014046.0
Q( 8)  FROM  NODES   11  TO   12            .0    6026046.0
Q( 9)  FROM  NODES   11  TO    6            .0    7765071.0
Q(10)  FROM  NODES    6  TO   11            .0    7791071.0
Q(11)  FROM  NODES   11  TO    3            .0    4791672.0
Q(12)  FROM  NODES    3  TO   11           3.0      68886.4
Q(13)  FROM  NODES    3  TO    6            .0   13502542.0
Q(14)  FROM  NODES    6  TO    3            .0   13580542.0
Q(15)  FROM  NODES    3  TO   13            .0   11516194.0
Q(16)  FROM  NODES   13  TO    3            .0   11566194.0
Q(17)  FROM  NODES   13  TO    6            .0    6257871.0
Q(18)  FROM  NODES    6  TO   13            .0    6285871.0
Q(19)  FROM  NODES    2  TO   13           4.3     175670.5
Q(20)  FROM  NODES   13  TO    2            .0   10028994.0
Q(21)  FROM  NODES   13  TO    5           4.3      47303.5
Q(22)  FROM  NODES    5  TO   13            .0    4301276.0
Q(23)  FROM  NODES    5  TO    6            .0    8040895.0
Q(24)  FROM  NODES    6  TO    5            .7     305488.8
Q(25)  FROM  NODES    5  TO    7            .0    7049098.0
Q(26)  FROM  NODES    7  TO    5            .0    7015098.0
Q(27)  FROM  NODES    7  TO    6            .0   12285916.0
Q(28)  FROM  NODES    6  TO    7           8.0     182540.6
Q(29)  FROM  NODES    7  TO    8            .0    8557552.0
Q(30)  FROM  NODES    8  TO    7            .0     210646.1
Q(31)  FROM  NODES    8  TO    9           1.5     304177.1
Q(32)  FROM  NODES    9  TO    8            .0   10777718.0
Q(33)  FROM  NODES    9  TO    7            .0   16284362.0
Q(34)  FROM  NODES    7  TO    9            .0   16310362.0
Q(35)  FROM  NODES    8  TO   10            .0    9008692.0
Q(36)  FROM  NODES   10  TO    8            .0    9046692.0
Q(37)  FROM  NODES   10  TO    9            .0   11810516.0
Q(38)  FROM  NODES    9  TO   10            .0   11756516.0
Q(39)  FROM  NODES    8  TO    4            .0   13774110.0
Q(40)  FROM  NODES    4  TO    8           2.7     233260.9
Q(41)  FROM  NODES    4  TO   10           2.0     358995.9
Q(42)  FROM  NODES   10  TO    4            .0   15551136.0
Q(43)  FROM  NODES    0  TO    0           1.3     -60168.1
Q(44)  FROM  NODES    0  TO    0            .0     -69447.8
Q(45)  FROM  NODES    0  TO    0            .0     -75989.9
Q(46)  FROM  NODES    0  TO    0           1.8     -62630.9

OBJECTIVE FUNCTION VALUE=      5817804.0
```

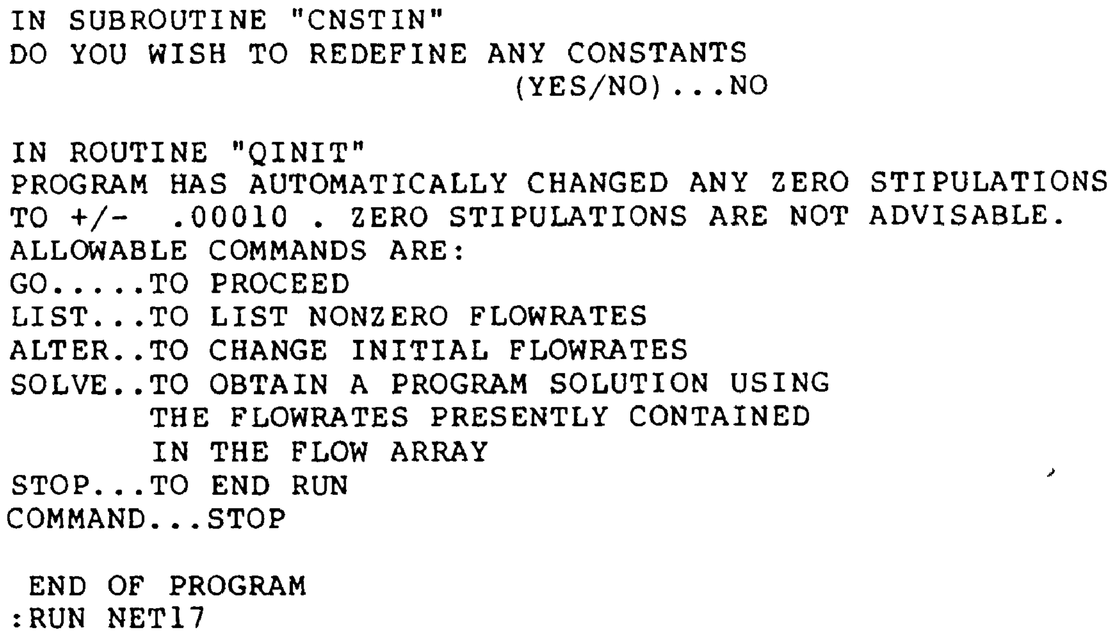
## FIGURE F·2(a) — (contd)

```
ITERATION   3
Q( 1) FROM NODES    1 TO  12          .0  11283618.0
Q( 2) FROM NODES   12 TO   1          .0  11285618.0
Q( 3) FROM NODES   12 TO   6          .0   7511997.0
Q( 4) FROM NODES    6 TO  12          .0   7549997.0
Q( 5) FROM NODES    6 TO   1          .0  12563742.0
Q( 6) FROM NODES    1 TO   6         9.4    130074.0
Q( 7) FROM NODES   12 TO  11          .0   6014046.0
Q( 8) FROM NODES   11 TO  12          .0   6026046.0
Q( 9) FROM NODES   11 TO   6          .0   7765071.0
Q(10) FROM NODES    6 TO  11          .0   7791071.0
Q(11) FROM NODES   11 TO   3          .0   4791672.0
Q(12) FROM NODES    3 TO  11         3.0     68886.4
Q(13) FROM NODES    3 TO   6          .0  13502542.0
Q(14) FROM NODES    6 TO   3          .0  13580542.0
Q(15) FROM NODES    3 TO  13          .0  11516194.0
Q(16) FROM NODES   13 TO   3          .0  11566194.0
Q(17) FROM NODES   13 TO   6          .0   6257871.0
Q(18) FROM NODES    6 TO  13          .0   6285871.0
Q(19) FROM NODES    2 TO  13         4.3    175670.5
Q(20) FROM NODES   13 TO   2          .0  10028994.0
Q(21) FROM NODES   13 TO   5         4.3     47303.5
Q(22) FROM NODES    5 TO  13          .0   4301276.0
Q(23) FROM NODES    5 TO   6          .0   8040895.0
Q(24) FROM NODES    6 TO   5          .7    305488.8
Q(25) FROM NODES    5 TO   7          .0   7049098.0
Q(26) FROM NODES    7 TO   5          .0   7015098.0
Q(27) FROM NODES    7 TO   6          .0  12285916.0
Q(28) FROM NODES    6 TO   7         8.0    163892.4
Q(29) FROM NODES    7 TO   8          .0   8557552.0
Q(30) FROM NODES    8 TO   7          .0   8547552.0
Q(31) FROM NODES    8 TO   9         1.5    304177.1
Q(32) FROM NODES    9 TO   8          .0  10777718.0
Q(33) FROM NODES    9 TO   7          .0  16284362.0
Q(34) FROM NODES    7 TO   9          .0  16310362.0
Q(35) FROM NODES    8 TO  10          .0   9008692.0
Q(36) FROM NODES   10 TO   8          .0   9046692.0
Q(37) FROM NODES   10 TO   9          .0  11810516.0
Q(38) FROM NODES    9 TO  10          .0  11756516.0
Q(39) FROM NODES    8 TO   4          .0  13774110.0
Q(40) FROM NODES    4 TO   8         2.7    292932.6
Q(41) FROM NODES    4 TO  10         2.0    358995.9
Q(42) FROM NODES   10 TO   4          .0  15551136.0
Q(43) FROM NODES    0 TO   0         1.3    -57065.0
Q(44) FROM NODES    0 TO   0          .0    -69447.8
Q(45) FROM NODES    0 TO   0          .0    -75989.9
Q(46) FROM NODES    0 TO   0         1.8    -67916.6

OBJECTIVE FUNCTION VALUE=      5688299.0
```

FIGURE F·2(a) — (contd)

```
                 PROBLEM SOLUTION
 PROCESSING CENTER COSTS
     CENTER                      FLOW       COST (MIL)

CLEARWATER LAKE                   9.43            .54
CRYSTAL CREEK                     4.30            .30
SULPHUR SPRINGS                   3.00            .23
DEAD MAN'S POND                   4.70            .32


 TRANSPORTATION COSTS
NODE         ROUTE                 FLOW     COST (MIL)

  1  CLEARWATER LAKE       TO
   6 BRACKWELL                     9.43           1.23
  3  SULPHUR SPRINGS       TO
  11 DRYGULCH                      3.00            .21
  2  CRYSTAL CREEK         TO
  13 JUNCTION TWO                  4.30            .76
 13  JUNCTION TWO          TO
   5 THIRSTYVILLE                  4.30            .20
  6  BRACKWELL             TO
   5 THIRSTYVILLE                   .70            .21
  6  BRACKWELL             TO
   7 NEWTON                        8.00           1.31
  8  BOGTON MOOR           TO
   9 SALTWELL FLATS                1.50            .46
  4  DEAD MAN'S POND       TO
   8 BOGTON MOOR                   2.70            .79
  4  DEAD MAN'S POND       TO
  10 FOSSETTVILLE                  2.00            .72


        PROCESS COST...... (MILLION)     1.384
        TRANSPN COST...... (MILLION)     5.882
        TOTAL SYSTEM ..... (MILLION)     7.266
 END OF REPORT
```

FIGURE F·2(a) — (contd)



```
IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                          (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP

 END OF PROGRAM
:RUN NET17
```

## FIGURE F·2(b) – SAMPLE COMPUTER RUN – INITIAL SOLUTION SPECIFIED

```
          NETWORK SYSTEM OPTIMIZATION PROGRAM
          -----------------------------------
                   BY  A A SMITH
                   AND R H TUFGAR




IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                           (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...LIST


INITIAL FLOW ASSUMPTIONS
U/S NODE   D/S NODE   FLOW(MGD)
   1          0       10.70000
   2          0        4.30000
   3          0        3.00000
   4          0        6.50000
   5          0        5.00000
   6          0         .73000
   7          0        8.00000
   8          0        1.20000
   9          0        1.50000
  10          0        2.00000
  11          0        3.00000
  12          0         .00010
  13          0         .00010
COMMAND...ALTER


THE FLOW ARRAY CONTAINS PREVIOUSLY ASSIGNED
OR CALCULATED FLOW VALUES. DO YOU WISH TO
ZERO THE FLOW ARRAY ELEMENTS? (YES/NO)...YES
```

FIGURE F·2(b) — (contd)

```
SUPPLY NUMBER OF FLOW CHANGES,
                (FORMAT I3)... 13

SUPPLY UPSTREAM NODE NUMBER, DOWNSTREAM NODE
NUMBER AND FLOW VARIABLE VALUE , (FORMAT I3,I3,F15.5)

CHANGE NUMBER   1...  1  6  10.7

CHANGE NUMBER   2...  6  7  8.0

CHANGE NUMBER   3...  6 12  0.0001

CHANGE NUMBER   4...  6 11  1.2698

CHANGE NUMBER   5...  6  5  0.7001

CHANGE NUMBER   6...  3 11  1.7302

CHANGE NUMBER   7...  2 13  4.3

CHANGE NUMBER   8... 13  5  4.2999

CHANGE NUMBER   9...  4  8  1.2

CHANGE NUMBER  10...  4 10  3.5

CHANGE NUMBER  11... 10  9  1.5

CHANGE NUMBER  12...  3  0  1.2698

CHANGE NUMBER  13...  4  0  1.8
COMMAND...GO
```

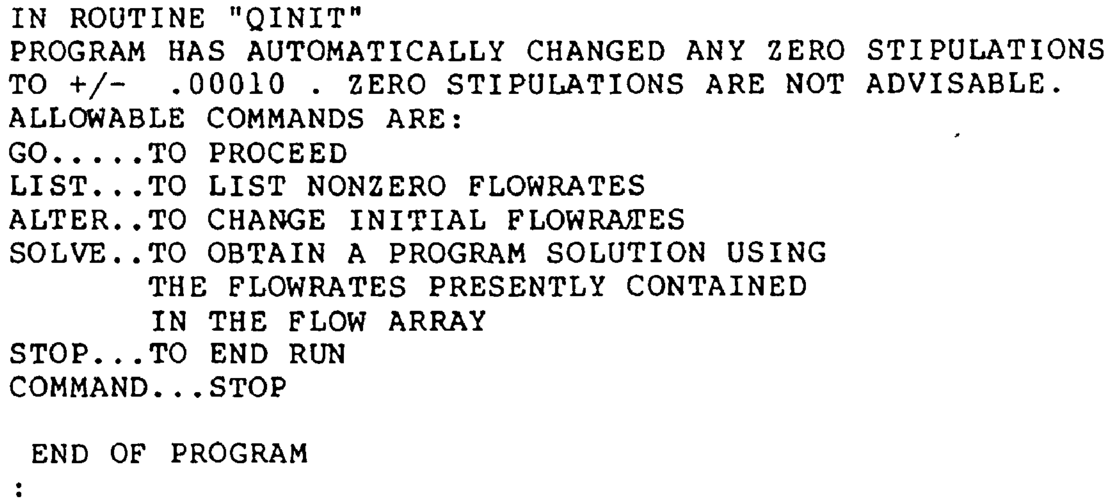
FIGURE F·2(b) — (contd)

```
IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO

IN ROUTINE "SOLVE"...DO YOU WANT DETAILS
OF COSTS DURING ITERATIONS. (YES/NO)...NO


DO YOU WISH TO USE THE "REPLCQ" SUBROUTINE
                           (YES/NO)...NO

ITERATION  1
ITERATION  2
```

## FIGURE F·2(b) – (contd)

```
                      PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                         FLOW        COST (MIL)

 CLEARWATER LAKE                     9.43            .54
 CRYSTAL CREEK                       4.30            .30
 SULPHUR SPRINGS                     3.00            .23
 DEAD MAN'S POND                     4.70            .32


  TRANSPORTATION COSTS
 NODE         ROUTE                    FLOW        COST (MIL)

   1  CLEARWATER LAKE       TO
    6 BRACKWELL                        9.43            1.23
   3  SULPHUR SPRINGS       TO
   11 DRYGULCH                         3.00             .21
   2  CRYSTAL CREEK         TO
   13 JUNCTION TWO                     4.30             .76
  13  JUNCTION TWO          TO
    5 THIRSTYVILLE                     4.30             .20
   6  BRACKWELL             TO
    5 THIRSTYVILLE                      .70             .21
   6  BRACKWELL             TO
    7 NEWTON                           8.00            1.31
  10  FOSSETTVILLE          TO
    9 SALTWELL FLATS                   1.50             .53
   4  DEAD MAN'S POND       TO
    8 BOGTON MOOR                      1.20             .51
   4  DEAD MAN'S POND       TO
   10 FOSSETTVILLE                     3.50             .96


         PROCESS COST...... (MILLION)     1.384
         TRANSPN COST...... (MILLION)     5.914
         TOTAL SYSTEM ..... (MILLION)     7.297
  END OF REPORT


IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                         (YES/NO)...NO
```

FIGURE F·2(b) - (contd)



```
IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP

 END OF PROGRAM
:
```

# TABLE F·I — NODAL DEFINITIONS

| NODE NO. | NODAL NAME | STATE (ft.) | STIPULATION (MGD) |
|---|---|---|---|
| | | | (supply) |
| 1 | Clearwater Lake | 430.0 | 10.70 |
| 2 | Crystal Creek | 370.0 | 4.30 |
| 3 | Sulphur Springs | 525.0 | 3.00 |
| 4 | Dead Man's Pond | 360.0 | 6.50 |
| | | | (demand) |
| 5 | Thirstyville | 275.0 | 5.00 |
| 6 | Brackwell | 330.0 | 0.73 |
| 7 | Newton | 360.0 | 8.00 |
| 8 | Bogton Moor | 385.0 | 1.20 |
| 9 | Saltwell Flats | 425.0 | 1.50 |
| 10 | Fossettville | 290.0 | 2.00 |
| 11 | Drygulch | 395.0 | 3.00 |
| 12 | Junction One | 425.0 | 0.00 |
| 13 | Junction Two | 400.0 | 0.00 |

## TABLE F·2 – LINK DEFINITIONS

| U/S NODE | D/S NODE | LENGTH (ft.) |
|---|---|---|
| 1 | 12 | 23750.0 |
| 12 | 6 | 15850.0 |
| 6 | 1 | 26400.0 |
| 12 | 11 | 12670.0 |
| 11 | 6 | 16370.0 |
| 11 | 3 | 10030.0 |
| 3 | 6 | 28500.0 |
| 3 | 13 | 24290.0 |
| 13 | 6 | 13200.0 |
| 2 | 13 | 21120.0 |
| 13 | 5 | 9000.0 |
| 5 | 6 | 16900.0 |
| 5 | 7 | 14800.0 |
| 7 | 6 | 25870.0 |
| 7 | 8 | 18000.0 |
| 8 | 9 | 22700.0 |
| 9 | 7 | 34300.0 |
| 8 | 10 | 19000.0 |
| 10 | 9 | 24800.0 |
| 8 | 4 | 29000.0 |
| 4 | 10 | 32700.0 |

# APPENDIX G

## SIMPLEX ALGORITHM

The search technique used in the SIMPLEX algorithm to move from one basic solution to another is similar to the nonlinear vertex-to-vertex solution technique presented in Chapter 5. Unfortunately, when applied to network problems with null nodes, the vertex-to-vertex search technique encounters difficulties. As shown in Chapter 5 (see page 164), the algorithm may not bring in flow variables joined to a null node. In this Appendix, a network problem is solved by hand, using the SIMPLEX method, to demonstrate the difference between the two solution methods and determine if the SIMPLEX algorithm encounters difficulties with null nodes. The cost functions used are, of course, linear, suitable for use in the SIMPLEX method. The sample problem would, therefore, represent a single iteration in the ILP algorithm (see Chapter 4, section 4.2.7).

A 5 node distribution network used for a sample problem is illustrated in figure G.1. The nodal stipulations are as follows:

| Node | Stipulation |
|---|---|
| 1 | 10.0 |
| 2 | 5.0 |
| 3 | 6.0 |
| 4 | 0.0 |
| 5 | 7.0 |

The linear cost coefficients (see figure G.1) are chosen to yield an optimal policy which involves the transmission of flow through the junction node, 4.

The cost coefficient array, structural matrix and stipulation array are written to form the mathematical statement of the linear problem.

| $C_i$ | 20 | 22 | 12 | 30 | 10 | 10 | 8 | 8 | 12 | 12 | 0 | 0 | 100 | 100 | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | STIP |
| 1 | 1 | 1 | | | | | | | | | 1 | | | | | 10 |
| 2 | | | 1 | 1 | | | | | | | | 1 | | | | 5 |
| 3 | 1 | | | | 1 | -1 | | | -1 | 1 | | | 1 | | | 6 |
| 4 | | 1 | 1 | | -1 | 1 | -1 | 1 | | | | | | 1 | | 0 |
| 5 | | | | 1 | | | 1 | -1 | 1 | -1 | | | | | 1 | 7 |

$$\text{The objective is:}\quad \underset{\bar{Q}^*}{\text{MIN}}\ Z \;=\; \sum_{i=1}^{NQ} C_i Q_i$$

The solution to this example is obtained by means of the SIMPLEX algorithm which is described in any text on linear programming (e.g. Aguilar 1973, reference 6). To determine if the solution algorithm will introduce variables joined to a null node into the solution, an initial solution is specified without node 4 being joined to another node (see figure G.2). The initial solution is also a degenerate basic feasible solution (ref. Chapter 5, page 163).

The structural matrix is first reduced to conform to the initial solution through the use of row operations. To use the SIMPLEX method to solve the problem by hand, the reduced structural matrix is set up in a tabular form, called the SIMPLEX Tableau (included in the following pages), with the modified stipulation array along the right hand side, cost coefficients along the top and the cost coefficients of the solution variables along the left hand side (their position is dictated by the location of the elements of the incidence matrix). Along the bottom of the tableau, the Ƶ and C-Ƶ arrays are located. The Ƶ array elements comprise the sum of the cost coefficients along the left side of the tableau multiplied by the structural coefficient of the same row for each column. The C-Ƶ array elements are formed by taking the difference between the cost coefficient for the comumn and the Ƶ array element.

Having obtained the SIMPLEX tableau in this manner, each iteration in the course of solution involves two steps.

(1) The variable entering the solution is selected from the values of $(C-Z)_i$, i=1,NQ, choosing the lowest value for minimization (say $(C-Z)_k$).

(2) To maintain a basic feasible solution, a variable is selected to be discarded on the basis of the θ Rule as described in Chapter 5 (i.e. Min $\frac{b_i}{a_{ik}}$ s.t. $a_{ik} > 0$, i=1,N).

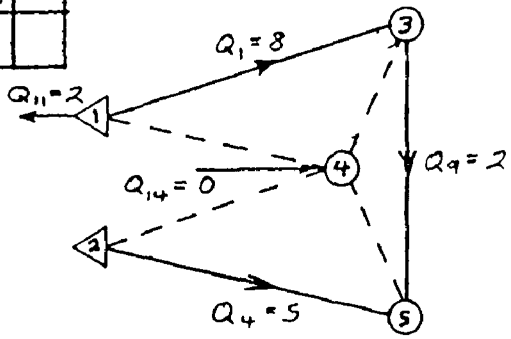
Upon choosing the variables to be transferred, a new reduced structural matrix is formed by row operations to use in the SIMPLEX Tableau for the next iteration. When all of the C-Z elements are non-negative, the optimal solution is found. In the working sheets on the pages following, the solution for the sample problem is determined. The SIMPLEX Tableau and diagram of the solution are shown for each iteration. The structural elements circled in the tableaux are the incidence matrix elements.

# WORKING SHEET 1

## ITERATION 1

| | $C_i$ | 20 | 22 | 12 | 30 | 10 | 10 | 8 | 8 | 12 | 12 | 0 | 0 | 100 | 100 | 100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $Q_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | B | θ | |
| c<br>0 | Q<br>11 | | 1 | 1 | | -1 | 1 | -1 | 1 | | | ① | 1 | -1 | | -1 | 2 | 2 | |
| 30 | 4 | | | 1 | ① | | | | | | | | 1 | | | | 5 | 5 | |
| 20 | 1 | ① | | -1 | | 1 | -1 | 1 | -1 | | | | -1 | 1 | | 1 | 8 | - | |
| 190 | 14 | | 1 | 1 | | -1 | 1 | -1 | 1 | | | | | | ① | | 0 | 0 | ← OUT |
| 12 | 9 | | | -1 | | | | 1 | -1 | ① | -1 | | -1 | | | 1 | 2 | - | |
| Z | | 20 | 100 | 98 | 30 | -80 | 80 | -68 | 68 | 12 | -12 | 0 | -2 | 20 | 100 | 32 | 334 | | |
| C-Z | | 0 | -78 | -86 | 0 | 90 | -70 | 76 | -60 | 0 | 94 | 0 | 2 | 80 | 0 | 68 | | | |

↑
IN

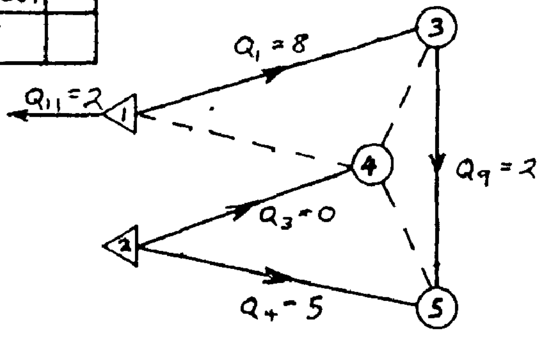


row 1 = row 1 - row 4
row 2 = row 2 - row 4
row 3 = row 3 + row 4
row 5 = row 4 + row 5

## ITERATION 2

| | $C_L$ | 20 | 22 | 12 | 30 | 10 | 10 | 8 | 8 | 12 | 12 | 0 | 0 | 100 | 100 | 100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $Q_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | B | θ | |
| c<br>0 | q<br>11 | | | | | | | | | | | ① | 1 | -1 | -1 | -1 | 2 | - | |
| 30 | 4 | | -1 | | ① | 1 | -1 | 1 | -1 | | | | 1 | | -1 | | 5 | 5 | ← OUT |
| 20 | 1 | ① | 1 | | | | | | | | | | -1 | 1 | 1 | 1 | 8 | - | |
| 12 | 3 | | 1 | ① | | -1 | 1 | -1 | 1 | | | | | | 1 | | 0 | - | |
| 12 | 9 | | 1 | | | -1 | 1 | | | ① | -1 | | -1 | | | 1 | 2 | - | |
| Z | | 20 | 14 | 12 | 30 | 6 | -6 | -18 | -18 | 12 | -12 | 0 | -2 | 20 | 2 | 32 | 334 | | |
| C-Z | | 0 | 8 | 0 | 0 | 4 | 16 | -10 | 26 | 0 | 24 | 0 | 2 | 80 | 98 | 68 | | | |

↑
IN



# WORKING SHEET 2

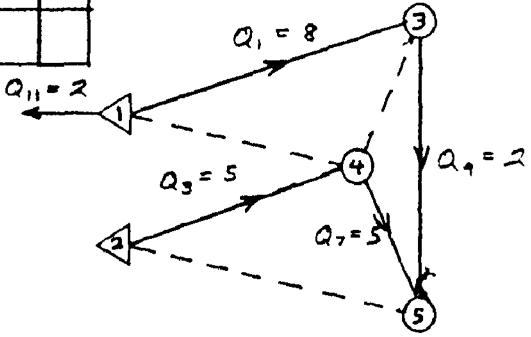
## ITERATION 3

row 4 = row 2 + row 4

| | Ci | 20 | 22 | 12 | 30 | 10 | 10 | 8 | 8 | 12 | 12 | 0 | 0 | 100 | 100 | 100 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qi | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | B | θ |
| c<br>0 | Q<br>11 | | | | | | | | | | | ① | 1 | -1 | -1 | -1 | 2 | - |
| 8 | 7 | | -1 | | 1 | 1 | -1 | ① | -1 | | | | 1 | | -1 | | 5 | - |
| 20 | 1 | ① | 1 | | | | | | | | | | -1 | 1 | 1 | 1 | 8 | 8 |
| 12 | 3 | | | ① | 1 | | | | | | | | 1 | | | | 5 | - |
| 12 | 9 | | 1 | | | -1 | 1 | | | ① | -1 | | -1 | | | | 2 | 2 |
| Z | | 20 | 24 | 12 | 20 | -4 | 4 | 8 | -8 | 12 | -12 | 0 | -12 | 20 | 12 | 32 | 284 | |
| C-Z | | 0 | -2 | 0 | 10 | 14 | 6 | 0 | 16 | 0 | 24 | 0 | 12 | 80 | 88 | 68 | | |

← OUT

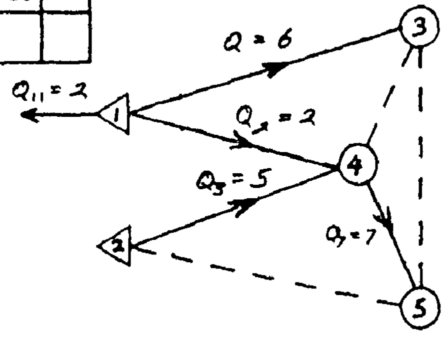
↑
IN



## ITERATION 4

row 2 = row 2 + row 5
row 3 = row 3 - row 5

| | Ci | 20 | 22 | 12 | 30 | 10 | 10 | 8 | 8 | 12 | 12 | 0 | 0 | 100 | 100 | 100 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qi | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | B | θ |
| c<br>0 | Q<br>11 | | | | | | | | | | | ① | 1 | -1 | -1 | -1 | 2 | |
| 8 | 7 | | | | 1 | | | ① | -1 | 1 | -1 | | | | -1 | 1 | 7 | |
| 20 | 1 | ① | | | | 1 | -1 | | | -1 | 1 | | | 1 | 1 | | 6 | |
| 12 | 3 | | | ① | 1 | | | | | | | | 1 | | | | 5 | |
| 22 | 2 | | ① | | | -1 | 1 | | | 1 | -1 | | -1 | | | 1 | 2 | |
| Z | | 20 | 22 | 12 | 20 | -2 | 2 | 8 | -8 | 10 | -10 | 0 | -10 | 20 | 12 | 30 | 280 | |
| C-Z | | 0 | 0 | 0 | 10 | 12 | 8 | 0 | 16 | 2 | 22 | 0 | 10 | 80 | 88 | 70 | | |



All of the elements of the C-Ƶ vector in iteration 4 are nonnegative implying that the optimal solution has been located. The first iteration brings in the variable, $Q_3$, which is joined to the null node demonstrating that the SIMPLEX method is stable when applied to a linear network problem exhibiting degeneracy due to null nodes. It is of interest to notice that although $Q_3$ is in the solution in the second iteration, it has a value of zero. Consequently, the system cost does not change from the first to second iteration (the last element of the Ƶ array provides the solution cost). In the nonlinear vertex-to-vertex search method, the solution in iteration 2 would not be considered as an initial solution for a following iteration since it does not yield an improvement to the objective. In the SIMPLEX method, however, the second iteration provides a "stepping stone" to reach the solution in iteration 3 where $Q_7$ is then introduced. In this solution, both $Q_3$ and $Q_7$ have nonzero values and there is a definite improvement in the solution cost. This then leads to the detection of the optimal solution shown in iteration 4.

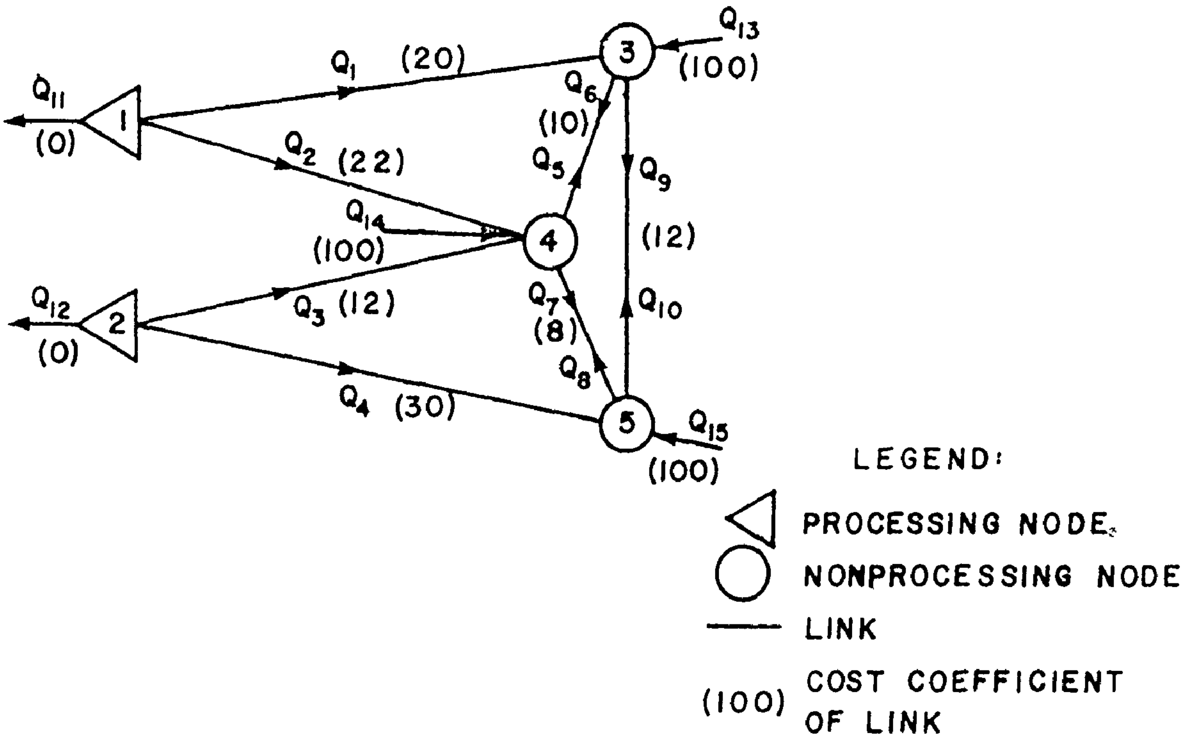


FIGURE G·1 — DISTRIBUTION NETWORK

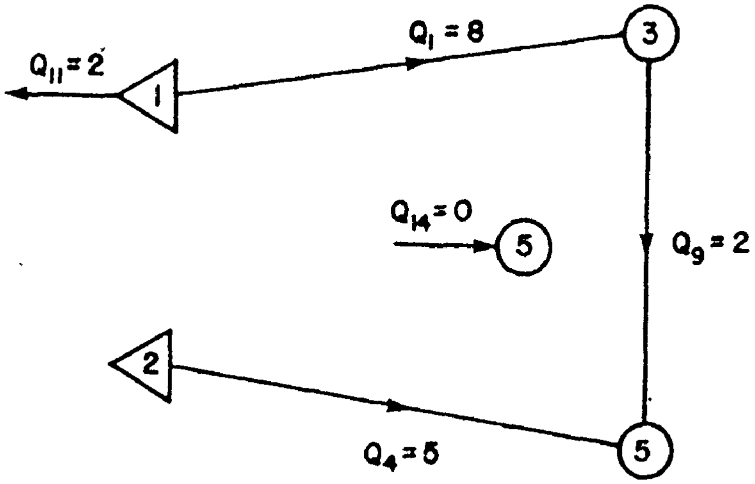


FIGURE G·2 — STARTING SOLUTION

# APPENDIX H

# LISTING OF THE HYVRST OPTIMIZATION ROUTINES

Subroutines included

SOLVE

HYVRST

INTFES

```
      SUBROUTINE SOLVE(A,B,C,CPEN,NDS,NUS,Q,STATE,XL,WA,WB,WC,WD,WE,
     +                 WF,WG,WH,COST,IFLAG,ITYPE,N,NCEN,NQ,NR2,NW)
C
C   ************************************************************
C       SUBROUTINE TO CALL THE OPTIMIZATION ROUTINE, HYVRST AND
C       DEFINE THE REQUIRED PARAMETERS
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C               DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     WA-WH   = WORKING ARRAYS FOR THE OPTIMIZATION ALGORITHMS.
C     COST    = OPTIMAL SOLUTION COST.
C     IFLAG   = ERROR FLAG. SET=1 IF ERROR ENCOUNTERED.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. TYPE=1 FOR A DISTRIBUTION NETWORK,
C                         TYPE=0 FOR A COLLECTION NETWORK.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C     NR2     = PERIPH. DEVICE NUMBER FOR KEYBOARD INPUT.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C
C     USES ROUTINES HYVRST, INTFES, SOLCST, LNKCST, CENCST.
C     USER DEFINED ROUTINES COST1, COST2.
C   ************************************************************
C
C       DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NDS(NQ),NUS(NQ)
      DIMENSION Q(NQ),STATE(N),XL(NQ)
C
C       WORKING ARRAYS.
C
      DIMENSION WA(N,NQ),WB(N),WC(N),WD(N)
      DIMENSION WE(N),WF(N),WG(N),WH(N,N)
      INTEGER WC,WD,WF,WH
C
      DATA YES/1HY/
C
```

```
C     SET NUMBER OF VARIABLES USED IN SEARCH.
C
      NVARS=NQ-N+NCEN
C
C     SET ITERATION LIMITS.
C
      MITER1=40
      MITER2=N
      MITER3=5
      IPRINT=0
      WRITE(NW,10)
 10   FORMAT(/' DO YOU WISH AN INTERMEDIATE PRINTOUT'/
     +        ' IN THE SOLUTION ROUTINE,  (YES/NO)...')
      READ(NR2,20)ANS
 20   FORMAT(A1)
      IF(ANS.EQ.YES)IPRINT=1
      CALL HYVRST(A,B,C,CPEN,NDS,NUS,Q,STATE,XL,WA,WB,WC,
     +            WD,WE,WF,WG,WH,COST,IFLAG,IPRINT,ITYPE,
     +            MITER1,MITER2,MITER3,N,NCEN,NQ,NVARS,NW)
      RETURN
      END
      SUBROUTINE HYVRST(A,B,C,CPEN,NDS,NUS,Q,STATE,XL,WA,WB,WC,
     +                  WD,WE,WF,WG,WH,COST,IFLAG,IPRINT,ITYPE,
     +                  MITER1,MITER2,MITER3,N,NCEN,NQ,NVARS,NW)
C
C     ************************************************************
C      OPTIMIZATION ALGORITHM DESIGNED FOR USE WITH THE NETWORK
C      OPTIMIZATION PROGRAM. FINDS THE MINIMUM OF A NONLINEAR
C      CONCAVE OBJECTIVE FUNCTION, DEFINED BY SOLCST, SUBJECT TO
C      LINEAR CONSTRAINTS, DEFINED BY MATRICES A AND B.
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C               DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C               STIPULATIONS. (WORKING ARRAY OF STIP)
C     C       = ONE DIMENSIONAL ARRAY OF COST COEFFICIENTS FOR
C               THE DESIGN VARIABLES.
C     CPEN    = ONE-DIMENSIONAL ARRAY OF PENALTY COST COEFFICIENTS
C               FOR THE DESIGN VARIABLES.
C     NDS     = ONE-DIMENSIONAL ARRAY CONTAINING THE DOWNSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     NUS     = ONE-DIMENSIONAL ARRAY CONTAINING THE UPSTREAM
C               NODAL NUMBERS OF THE DESIGN VARIABLES.
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     STATE   = ONE-DIMENSIONAL ARRAY CONTAINING THE STATE VALUES
C               OF THE NODAL POINTS.
C     XL      = ONE-DIMENSIONAL ARRAY CONTAINING THE LENGTH VALUE
C               ASSOCIATED WITH EACH DESIGN VARIABLE.
C     WA-WH   = WORKING ARRAYS FOR THE OPTIMIZATION ALGORITHMS.
C     COST    = OPTIMAL SOLUTION COST.
C     IFLAG   = ERROR FLAG. SET=1 IF ERROR ENCOUNTERED.
C     IPRINT  = INTERMEDIATE PRINTOUT FLAG. SET=1 FOR PRINTOUT.
C     ITYPE   = FLAG TO DENOTE TYPE OF NETWORK PROBLEM BEING
C               ANALYSED. TYPE=1 FOR A DISTRIBUTION NETWORK,
C                         TYPE=0 FOR A COLLECTION NETWORK.
C     MITER1  = MAXIMUM NUMBER OF ITERATIONS ALLOWED TO
C               LOCATE AN OPTIMUM IN THE INITIAL SEARCH.
C     MITER2  = MAXIMUM NUMBER OF ITERATIONS ALLOWED TO
C               LOCATE A SOLUTION LESS COSTLY THAN THE
C               OPTIMUM (SECONDARY SEARCH)(N RECOMMENDED).
C     MITER3  = MAXIMUM NUMBER OF PRIMARY AND SECONDARY
C               SEARCHES ALLOWED (USUALLY NO MORE THAN 3
C               REQUIRED).
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NCEN    = NUMBER OF PROCESSING NODES IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C               ARTIFICIAL SLACK VARIABLES.
C     NVARS   = NUMBER OF VARIABLES WHICH MAY BE BROUGHT INTO THE
C               SOLUTION. COUNT STARTS AT THE FIRST VARIABLE.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C
C     USES ROUTINES INTFES, SOLCST, LNKCST, CENCST.
C     USER SUPPLIED ROUTINES COST1, COST2.
C     ************************************************************
C
```

```
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),C(NQ),CPEN(NQ),NDS(NQ),NUS(NQ)
      DIMENSION Q(NQ),STATE(N),XL(NQ)
C
C     WORKING ARRAYS.
C
      DIMENSION WA(N,NQ),WB(N),WC(N),WD(N)
      DIMENSION WE(N),WF(N),WG(N),WH(N,N)
      INTEGER WC,WD,WF,WH
C
C      CALL INTFES TO ESTABLISH INITIAL BASIC FEASABLE SOLUTION.
C        WA = MODIFIED STRUCTURAL MATRIX.
C        WB = MODIFIED STIPULATIONS.
C        WC = INTEGER ARRAY IDENTIFYING VARIABLES IN SOLUTION.
C
      CALL INTFES(A,B,Q,WA,WB,WC,IFLAG,N,NQ,NW)
C
C      CALCULATE COST OF INITIAL SOLUTION.
C
      IF(IFLAG.EQ.1)GO TO 220
      CALL SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
     +            COST1,ITYPE,N,NCEN,NQ)
      IF(IPRINT.NE.1)GO TO 5
      WRITE(NW,1020)
 1020 FORMAT(//' SUBROUTINE HYVRST'/' INTERMEDIATE SOLUTIONS'/
     +' VARIABLE NO.     VALUE'/)
      WRITE(NW,1030)(WC(I),WB(I),I=1,N)
 1030 FORMAT(//' INITIAL SOLUTION'/(I7,5X,F15.5))
      WRITE(NW,1040)COST1
 1040 FORMAT(/' COST= ',F15.2//)
 5    CONTINUE
      COST=COST1
      ITER1=0
C
C     EXTERNAL ITERATIVE LOOP.
C
 250  CONTINUE
      ITER1=ITER1+1
      IF(ITER1.LE.MITER3)GO TO 2
      WRITE(NW,1025)
 1025 FORMAT(/' IN ROUTINE HYVRST'/
     +' MITER3 EXCEEDED')
      IFLAG=1
      GO TO 220
 2    CONTINUE
      ITER=0
C
C     PRIMARY ITERATIVE LOOP.
C
 200  CONTINUE
      NEWS=0
      ITER=ITER+1
      IF(ITER.LE.MITER1)GO TO 3
      WRITE(NW,1070)
 1070 FORMAT(/' IN ROUTINE HYVRST'/
     +' MITER1 EXCEEDED')
      IFLAG=1
      GO TO 220
 3    CONTINUE
C
C     CHECK FOR LINEAR DEPENDANCY IN EQUATIONS.
C
      DO 7 I=1,N
      DO 8 J=1,NQ
      IF(ABS(WA(I,J)).GT.0.9)GO TO 7
 8    CONTINUE
      WRITE(NW,1000)
 1000 FORMAT(/' IN ROUTINE HYVRST'/
     +' EQUATIONS ARE LINEARLY DEPENDANT')
      IFLAG=1
      GO TO 220
 7    CONTINUE
C
C     SEARCH FOR MORE ECONOMICAL SOLUTION.
C
      DO 10 J=1,NVARS
C
C     CHECK FOR VARIABLE J BEING IN PRESENT SOLUTION.
C
      DO 15 I=1,N
      IF(WC(I).EQ.J)GO TO 10
 15   CONTINUE
C
C     CHECK FOR LOOPING SOLUTION.
C
      DO 30 I=1,N
      IF(WA(I,J).GT.0.9)GO TO 35
 30   CONTINUE
      GO TO 10
 35   CONTINUE
C
C     CHECK FOR INFEASABLE SOLUTION.
C
      DO 40 I=1,N
      IF(WB(I).LT.1.0E-10.AND.WA(I,J).GT.0.9)GO TO 10
 40   CONTINUE
C
C     DETERMINE VARIABLE DELETED FROM SOLUTION.
C
      THETA=10.0**12
      DO 50 I=1,N
      IF(WA(I,J).LT.0.9)GO TO 50
      IF(WB(I).GE.THETA)GO TO 50
      THETA=WB(I)
      IQOUT=WC(I)
 50   CONTINUE
      IQIN=J
```

```
C
C      FIND TRIAL SOLUTION AND COST.
C
       DO 60 I=1,NQ
 60    Q(I)=0.0
       Q(IQIN)=THETA
       DO 70 I=1,N
       IQ=WC(I)
       Q(IQ)=WB(I)-WA(I,J)*THETA
 70    CONTINUE
       CALL SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
      +            COST2,ITYPE,N,NCEN,NQ)
C
C      STORE TRIAL SOLUTION IF THERE IS AN IMPROVEMENT.
C
       IF(COST2.GE.COST)GO TO 10
       NEWS=1
       COST=COST2
       DO 80 I=1,N
       WD(I)=WC(I)
       IF(WD(I).EQ.IQOUT)WD(I)=IQIN
       IQ=WD(I)
       WE(I)=Q(IQ)
 80    CONTINUE
 10    CONTINUE
C
C      IF NO IMPROVEMENT HAS BEEN FOUND DURING THE LAST ITERATION
C      THE OPTIMAL SOLUTION HAS BEEN FOUND.
C
       IF(NEWS.EQ.0)GO TO 210
C
C      RECONSTRUCT MATRIX FOR NEXT ITERATION.
C
C      FIND PIVOTAL POINT.
C
       DO 90 I=1,N
       IF(WD(I).NE.WC(I))GO TO 100
 90    CONTINUE
 100   IPIVOT=I
       IQIN=WD(IPIVOT)
C
C      REDEFINE MATRIX VALUES.
C
       DO 110 I=1,N
       WB(I)=WE(I)
       WC(I)=WD(I)
       IF(I.EQ.IPIVOT)GO TO 110
       IF(WA(I,IQIN).LT.-0.9)GO TO 120
       IF(WA(I,IQIN).GT.0.9)GO TO 130
       GO TO 110
 120   DO 125 J=1,NQ
 125   WA(I,J)=WA(I,J)+WA(IPIVOT,J)
       GO TO 110
 130   DO 135 J=1,NQ
```

```
  135 WA(I,J)=WA(I,J)-WA(IPIVOT,J)
  110 CONTINUE
      IF(IPRINT.NE.1)GO TO 200
      WRITE(NW,1080)ITER,(WC(I),WB(I),I=1,N)
 1080 FORMAT(/' ITER ',I5/(I7,5X,F15.5))
      WRITE(NW,1090)COST
 1090 FORMAT(/' COST= ',F15.2//)
      GO TO 200
C
  210 CONTINUE
C
C     OPTIMAL SOLUTION FOUND.
C
      DO 260 I=1,N
      WF(I)=WD(I)
      WG(I)=WE(I)
  260 CONTINUE
C
C     CHECK FURTHER FOR OPTIMALITY.
C
      IF(IPRINT.EQ.1)WRITE(NW,1050)
 1050 FORMAT(/' OPTIMAL SOLUTION FOUND,'/
     +          ' SEARCH CONTINUED AS A CHECK')
      COST3=COST
      DO 265 I=1,N
      DO 266 II=1,N
  266 WH(I,II)=0
  265 CONTINUE
      ITER=0
C
C     SECONDARY ITERATIVE LOOP.
C
  500 CONTINUE
      NEWS=0
      ITER=ITER+1
      IF(ITER.LE.MITER2)GO TO 300
      GO TO 510
  300 CONTINUE
C
C     CHECK FOR LINEAR DEPENDANCY IN EQUATIONS.
C
      DO 320 I=1,N
      DO 330 J=1,NQ
      IF(ABS(WA(I,J)).GT.0.9)GO TO 320
  330 CONTINUE
      WRITE(NW,1000)
      IFLAG=1
      GO TO 220
  320 CONTINUE
C
C     SEARCH FOR MORE ECONOMICAL SOLUTION.
C
      INIT=1
      DO 340 J=1,NVARS
```

```
C
C      CHECK FOR VARIABLE J BEING IN PRESENT SOLUTION.
C
      DO 350 I=1,N
      IF(WC(I).EQ.J)GO TO 340
 350  CONTINUE
C
C     CHECK FOR LOOPING SOLUTION.
C
      DO 360 I=1,N
      IF(WA(I,J).GT.0.9)GO TO 370
 360  CONTINUE
      GO TO 340
 370  CONTINUE
C
C     CHECK FOR INFEASIBLE SOLUTION.
C
      DO 380 I=1,N
      IF(WB(I).LT.1.0E-10.AND.WA(I,J).GT.0.9)GO TO 340
 380  CONTINUE
C
C     DETERMINE VARIABLES DELETED FROM SOLUTION.
C
      THETA=10.0**12
      DO 390 I=1,N
      IF(WA(I,J).LT.0.9)GO TO 390
      IF(WB(I).GE.THETA)GO TO 390
      THETA=WB(I)
      IQOUT=WC(I)
 390  CONTINUE
      IQIN=J
C
C     FIND TRIAL SOLUTION AND COST
C
      DO 400 I=1,NQ
 400  Q(I)=0.0
      Q(IQIN)=THETA
      DO 410 I=1,N
      IQ=WC(I)
      Q(IQ)=WB(I)-WA(I,J)*THETA
 410  CONTINUE
      IF(ITER.EQ.1)GO TO 418
      ITERM1=ITER-1
      DO 415 JJ=1,ITERM1
      DO 416 I=1,N
      DO 417 II=1,N
      IQ=WC(I)
      IF(IQ.EQ.IQOUT)IQ=IQIN
      IF(IQ.EQ.WH(JJ,II))GO TO 416
 417  CONTINUE
      GO TO 415
 416  CONTINUE
      GO TO 340
```

```
  418 CONTINUE
      CALL SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
     +           COST2,ITYPE,N,NCEN,NQ)
C
C     IF TRIAL SOLUTION COST IS LESS THAN PREVIOUS OPTIMAL COST
C     THEN THE SEARCH BEGINS AGAIN.
C     STORE NEXT LOWEST SOLUTION COST IN WH( ).
C
      IF(INIT.EQ.1)GO TO 419
      IF(ABS(COST2-COST3).LT.0.01)GO TO 340
      IF(COST2.GE.COST)GO TO 340
  419 INIT=2
      IF(COST2.LT.COST3)NEWS=1
      COST=COST2
      DO 420 I=1,N
      WD(I)=WC(I)
      IF(WD(I).EQ.IQOUT)WD(I)=IQIN
      IQ=WD(I)
      WE(I)=Q(IQ)
  420 CONTINUE
  340 CONTINUE
C
C     RECONSTRUCT MATRIX FOR NEXT ITERATION.
C
C     STORE SOLUTION FOR COMPARISON DURING NEXT ITERATION.
C     FIND PIVOTAL POINT.
C
      DO 425 I=1,N
      WH(ITER,I)=WD(I)
  425 CONTINUE
      DO 430 I=1,N
      IF(WD(I).NE.WC(I))GO TO 440
  430 CONTINUE
  440 IPIVOT=I
      IQIN=WD(IPIVOT)
C
C     REDEFINE MATRIX VALUES.
C
      DO 450 I=1,N
      WB(I)=WE(I)
      WC(I)=WD(I)
      IF(I.EQ.IPIVOT)GO TO 450
      IF(WA(I,IQIN).LT.-0.9)GO TO 460
      IF(WA(I,IQIN).GT.0.9)GO TO 480
      GO TO 450
  460 DO 470 J=1,NQ
  470 WA(I,J)=WA(I,J)+WA(IPIVOT,J)
      GO TO 450
  480 DO 490 J=1,NQ
  490 WA(I,J)=WA(I,J)-WA(IPIVOT,J)
  450 CONTINUE
      IF(IPRINT.NE.1)GO TO 495
      WRITE(NW,1080)ITER,(WC(I),WB(I),I=1,N)
      WRITE(NW,1090)COST
  495 CONTINUE
      IF(NEWS.EQ.1)GO TO 250
      GO TO 500
C
C     ASSIGN SOLUTION TO Q ARRAY, CALCULATE THE COST COEFFICIENTS
C
  510 CONTINUE
      IF(IPRINT.EQ.1)WRITE(NW,1100)
 1100 FORMAT(//' OPTIMAL SOLUTION FOUND'/)
      DO 230 J=1,NQ
  230 Q(J)=0.0
      DO 240 I=1,N
      IQ=WF(I)
      Q(IQ)=WG(I)
  240 CONTINUE
      CALL SOLCST(B,C,CPEN,NDS,NUS,Q,STATE,XL,
     +            COST,ITYPE,N,NCEN,NQ)
  220 CONTINUE
      RETURN
      END
      SUBROUTINE INTFES(A,B,Q,WA,WB,WC,IFLAG,N,NQ,NW)
C
C     ***********************************************************
C     SUBROUTINE TO DETERMINE AN INITIAL FEASIBLE SOLUTION FOR
C     HYVRST. IF Q ARRAY CONTAINS AN INITIAL SOLUTION WA AND WB
C     ARE MODIFIED TO CONTAIN THE SOLUTION. IF THERE IS NO INITIAL
C     STARTING POINT THEN THE SLACK AND ARTIFICIAL SLACK VARIABLES
C     ARE SET EQUAL TO THE STIPULATIONS IN B.
C     IF ANY STIPULATIONS ARE NEGATIVE THEIR SIGN IS REVERSED
C     ALONG WITH THE SIGNS OF THE CORRESPONDING ROW OF THE
C     STRUCTURAL MATRIX WA( ).
C
C     A       = TWO-DIMENSIONAL ARRAY OF STRUCTURAL COEFFICIENTS
C                DEFINING THE PROBLEM CONSTRAINTS.
C     B       = ONE-DIMENSIONAL ARRAY CONTAINING THE CONSTRAINT
C                STIPULATIONS. (WORKING ARRAY OF STIP)
C     Q       = ONE-DIMENSIONAL ARRAY OF THE DESIGN VARIABLES.
C     WA-WC   = WORKING ARRAYS FOR THE OPTIMIZATION ALGORITHMS.
C     IFLAG   = ERROR FLAG. SET=1 IF ERROR ENCOUNTERED.
C     N       = NUMBER OF NODAL POINTS IN THE NETWORK.
C     NQ      = NUMBER OF FLOWRATE VARIABLES PLUS SLACK AND
C                ARTIFICIAL SLACK VARIABLES.
C     NW      = PERIPH. DEVICE NUMBER FOR OUTPUT.
C     ***********************************************************
C
C     DIMENSION ARRAY STORAGE.
C
      DIMENSION A(N,NQ),B(N),Q(NQ)
C
C     WORKING ARRAYS.
C
      DIMENSION WA(N,NQ),WB(N),WC(N)
      INTEGER WC
C
      IFLAG=0
C
C     ZERO WC, SET WA TO A, SET WB TO B
C
      DO 10 I=1,N
      DO 15 J=1,NQ
 15   WA(I,J)=A(I,J)
      WB(I)=B(I)
 10   WC(I)=0
C
C     CHECK THAT THE STIPULATIONS ARE NONNEGATIVE.
C
      DO 20 I=1,N
      IF(WB(I).GE.0.0)GO TO 20
      DO 25 J=1,NQ
      IF(ABS(WA(I,J)).LT.0.9)GO TO 25
      WA(I,J)=-WA(I,J)
 25   CONTINUE
      WB(I)=-WB(I)
 20   CONTINUE
```

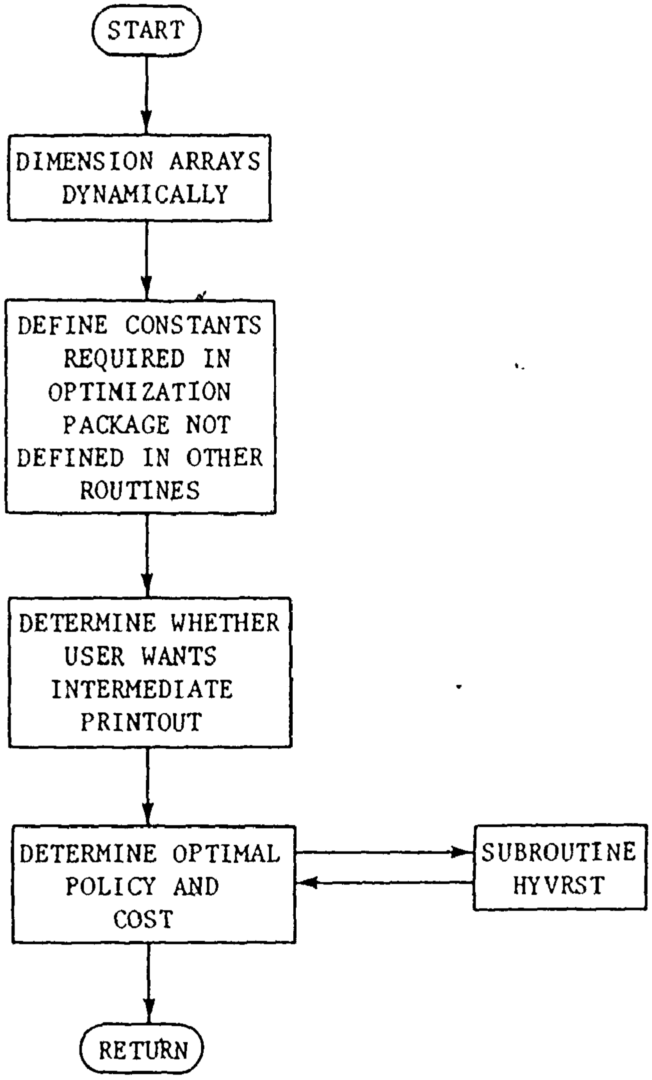
```
C
C     CHECK IF AN INITIAL SOLUTION IS SPECIFIED.
C
      L2=NQ-N
      DO 30 J=1,L2
      IF(Q(J).GT.1.0E-10)GO TO 100
 30   CONTINUE
C
C     FIND INITIAL BASIC FEASIBLE SOLUTION USING SLACK VARIABLES.
C
C     CHECK THAT THERE ARE N SLACK OR ARTIFICAL SLACK VARIABLES.
C
      DO 40 I=1,N
      ISLACK=L2+I
      DO 45 II=1,N
      IF(II.EQ.I)GO TO 45
      IF(ABS(WA(II,ISLACK)).GT.0.9)GO TO 50
 45   CONTINUE
      GO TO 60
 50   WRITE(NW,1000)ISLACK
 1000 FORMAT(' IN ROUTINE INTFES'/
     +' SLACK VARIABLE',I10,' IN MORE THAN ONE EQUATION')
      IFLAG=1
      GO TO 300
C
C     SET SLACK VARIABLE EQUAL TO STIPULATIONS.
C
 60   CONTINUE
      Q(ISLACK)=WB(I)
      WC(I)=ISLACK
 40   CONTINUE
      GO TO 300
C
C     RECONSTRUCT STRUCTURAL MATRIX AND STIPULATIONS FOR GIVEN
C     INITIAL SOLUTION.
C
C     CHECK FOR NEGATIVE FLOWRATES AND NONBASIC SOLUTION.
C
 100  CONTINUE
      ICOUNT=0
      DO 105 J=1,NQ
      IF(Q(J).LT.-1.0E-10)GO TO 106
      IF(Q(J).GT.1.0E-10)ICOUNT=ICOUNT+1
 105  CONTINUE
      IF(ICOUNT.LE.N)GO TO 107
      WRITE(NW,1025)
 1025 FORMAT(/' IN ROUTINE INTFES'/
     +' INITIAL SOLUTION IS NONBASIC, DECREASE NUMBER OF'/
     +' SOLUTION VARIABLES TO LESS THAN OR EQUAL TO THE'/
     +' NUMBER OF NODES.')
      IFLAG=1
      GO TO 300
 106  WRITE(NW,1030)
 1030 FORMAT(/' IN ROUTINE INTFES'/
     +' NEGATIVE FLOWRATES HAVE BEEN FOUND')
      IFLAG=1
      GO TO 300
 107  CONTINUE
C
C     CHECK FOR CONTINUITY.
C
      DO 110 I=1,N
      SUM=0.0
      DO 120 J=1,NQ
 120  SUM=SUM+WA(I,J)*Q(J)
      IF(ABS(SUM-WB(I)).GT.1.0E-10)GO TO 130
 110  CONTINUE
      GO TO 140
 130  WRITE(NW,1010)I
 1010 FORMAT(/' IN ROUTINE INTFES'/
     +' CONTINUITY NOT ESTABLISHED AT NODE',I10)
      IFLAG=1
      GO TO 300
 140  CONTINUE
C
C     RECONSTRUCT MATRICES SO THAT THEY CONFORM TO THE
C     INITIAL SOLUTION.
 250  CONTINUE
      NEWS=0
      DO 150 I=1,N
      IF(WC(I).GT.(NQ-N))GO TO 154
      IF(WC(I).NE.0)GO TO 150
 154   CONTINUE
      DO 155 J=1,NQ
      IF(ABS(Q(J)-WB(I)).GT.1.0E-10)GO TO 155
      IF(WA(I,J).LT.0.9)GO TO 155
      IF(WB(I).LT.1.0E-10.AND.J.LE.(NQ-N))GO TO 155
      IF(WC(I).EQ.J)GO TO 150
      GO TO 160
 155   CONTINUE
      GO TO 150
 160  CONTINUE
      DO 190 II=1,N
      IF(II.EQ.I)GO TO 190
      IF(WA(II,J).LT.-0.9)GO TO 200
      IF(WA(II,J).GT.0.9)GO TO 210
      GO TO 190
 200  DO 205 JJ=1,NQ
 205  WA(II,JJ)=WA(II,JJ)+WA(I,JJ)
      WB(II)=WB(II)+WB(I)
      GO TO 190
 210  DO 215 JJ=1,NQ
 215  WA(II,JJ)=WA(II,JJ)-WA(I,JJ)
      WB(II)=WB(II)-WB(I)
      IF(WB(II).GE.0.0)GO TO 190
      DO 220 JJ=1,NQ
      IF(ABS(WA(II,JJ)).LT.0.9)GO TO 220
      WA(II,JJ)=-WA(II,JJ)
```

```
220  CONTINUE
     WB(II)=-WB(II)
190  CONTINUE
     NEWS=1
     GO TO 230
150  CONTINUE
230  WC(I)=J
     IF(NEWS.EQ.1)GO TO 235
     IFLAG=1
     WRITE(NW,1020)
1020 FORMAT(/' IN ROUTINE INTFES'/
    +' INITIAL FEASIBLE SOLUTION CANNOT BE FOUND')
     GO TO 300
235  CONTINUE
     DO 240 I=1,N
     IF(WC(I).EQ.0)GO TO 250
240  CONTINUE
300  CONTINUE
     RETURN
     END
```

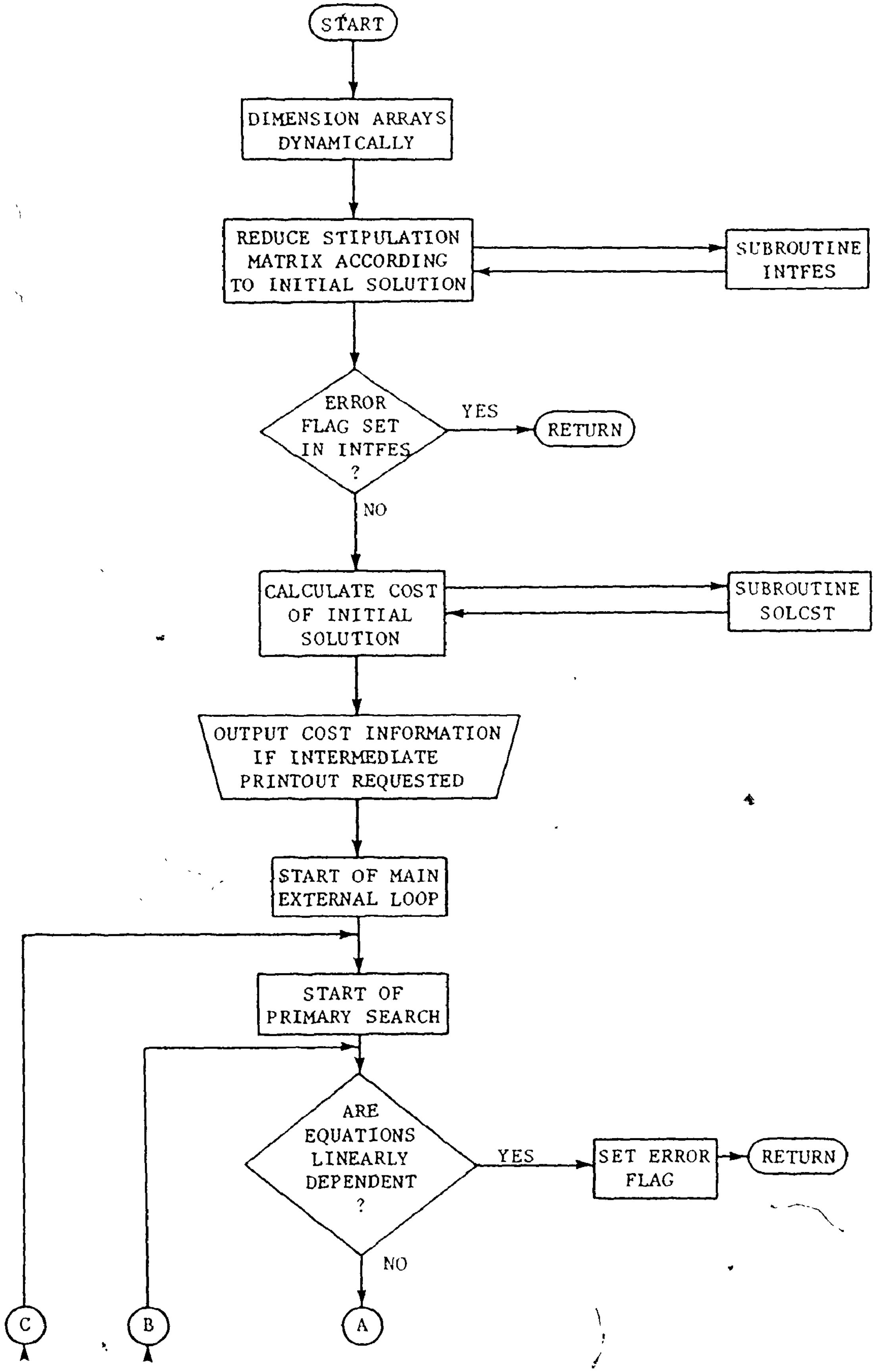
# APPENDIX I

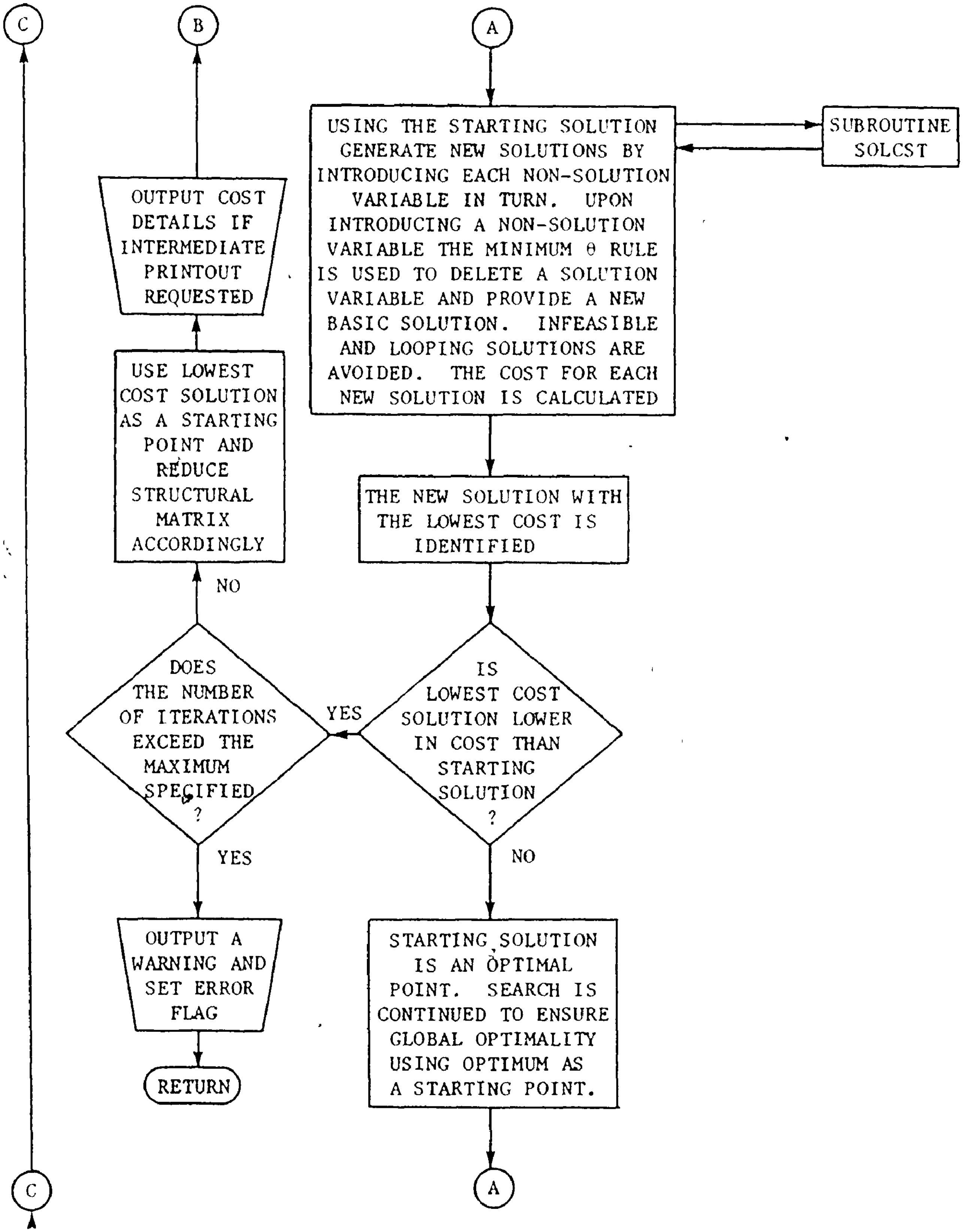
## FLOWCHARTS FOR THE HYVRST OPTIMIZATION ROUTINES

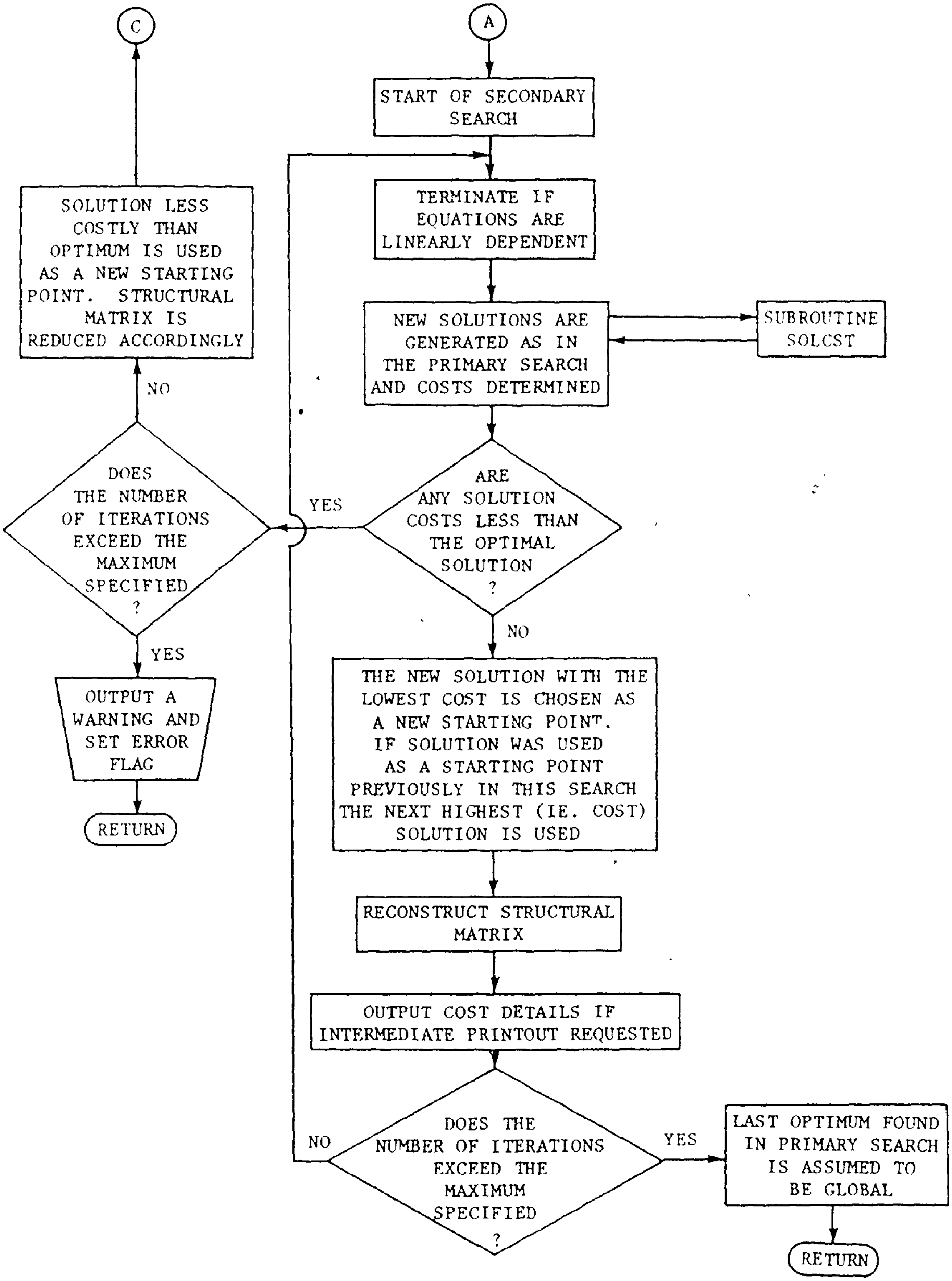
SUBROUTINE SOLVE


START
DIMENSION ARRAYS DYNAMICALLY
DEFINE CONSTANTS REQUIRED IN OPTIMIZATION PACKAGE NOT DEFINED IN OTHER ROUTINES
DETERMINE WHETHER USER WANTS INTERMEDIATE PRINTOUT
DETERMINE OPTIMAL POLICY AND COST
SUBROUTINE HYVRST
RETURN

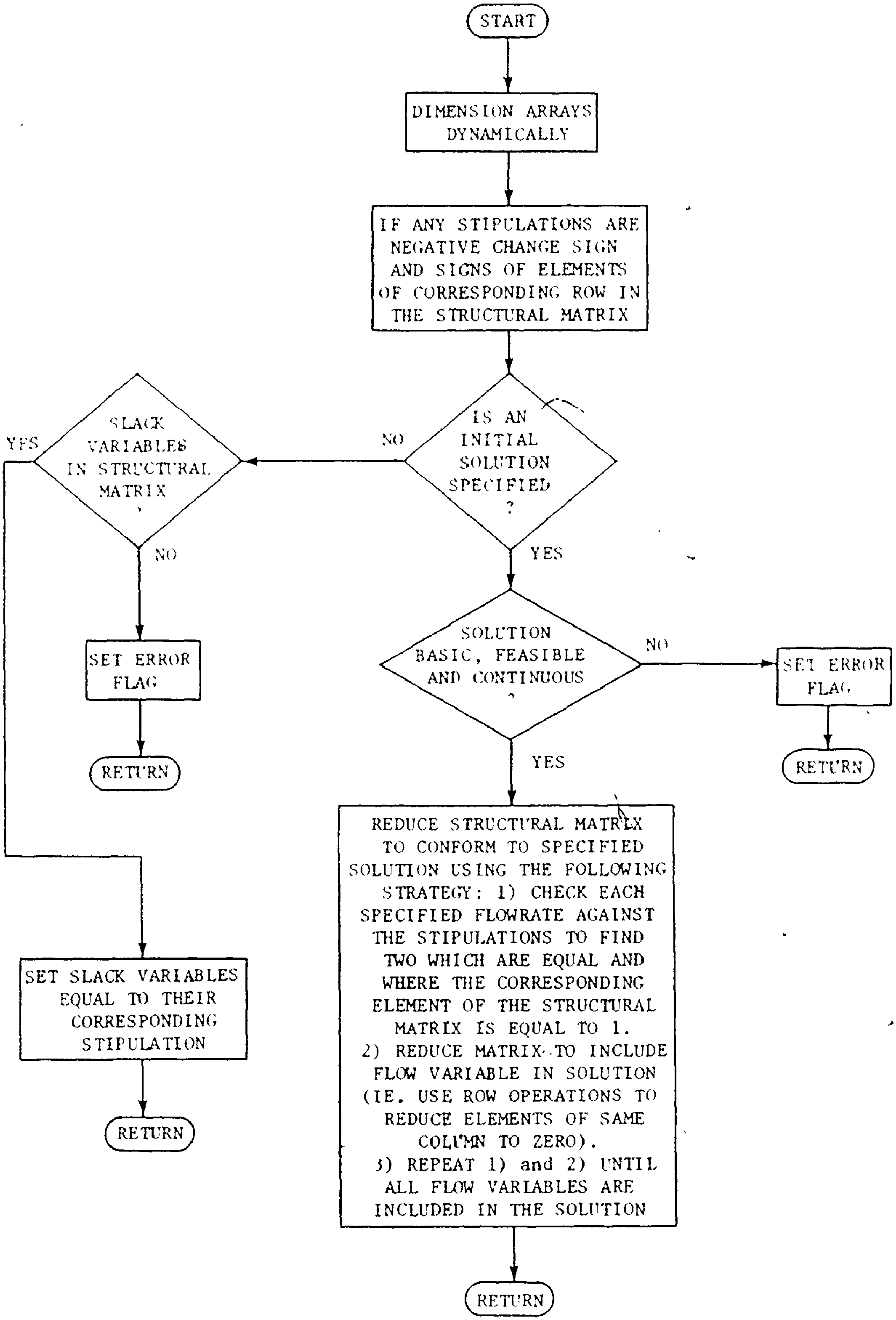
SUBROUTINE HYVRST


START
DIMENSION ARRAYS DYNAMICALLY
REDUCE STIPULATION MATRIX ACCORDING TO INITIAL SOLUTION
SUBROUTINE INTFES
ERROR FLAG SET IN INTFES ?
YES
RETURN
NO
CALCULATE COST OF INITIAL SOLUTION
SUBROUTINE SOLCST
OUTPUT COST INFORMATION IF INTERMEDIATE PRINTOUT REQUESTED
START OF MAIN EXTERNAL LOOP
START OF PRIMARY SEARCH
ARE EQUATIONS LINEARLY DEPENDENT ?
YES
SET ERROR FLAG
RETURN
NO
C
B
A


C
B
A
USING THE STARTING SOLUTION GENERATE NEW SOLUTIONS BY INTRODUCING EACH NON-SOLUTION VARIABLE IN TURN. UPON INTRODUCING A NON-SOLUTION VARIABLE THE MINIMUM θ RULE IS USED TO DELETE A SOLUTION VARIABLE AND PROVIDE A NEW BASIC SOLUTION. INFEASIBLE AND LOOPING SOLUTIONS ARE AVOIDED. THE COST FOR EACH NEW SOLUTION IS CALCULATED
SUBROUTINE SOLCST
OUTPUT COST DETAILS IF INTERMEDIATE PRINTOUT REQUESTED
USE LOWEST COST SOLUTION AS A STARTING POINT AND REDUCE STRUCTURAL MATRIX ACCORDINGLY
THE NEW SOLUTION WITH THE LOWEST COST IS IDENTIFIED
NO
DOES THE NUMBER OF ITERATIONS EXCEED THE MAXIMUM SPECIFIED ?
YES
IS LOWEST COST SOLUTION LOWER IN COST THAN STARTING SOLUTION ?
YES
NO
OUTPUT A WARNING AND SET ERROR FLAG
RETURN
STARTING SOLUTION IS AN OPTIMAL POINT. SEARCH IS CONTINUED TO ENSURE GLOBAL OPTIMALITY USING OPTIMUM AS A STARTING POINT.
C
A


C
A
START OF SECONDARY SEARCH
TERMINATE IF EQUATIONS ARE LINEARLY DEPENDENT
NEW SOLUTIONS ARE GENERATED AS IN THE PRIMARY SEARCH AND COSTS DETERMINED
SUBROUTINE SOLCST
ARE ANY SOLUTION COSTS LESS THAN THE OPTIMAL SOLUTION ?
YES
NO
THE NEW SOLUTION WITH THE LOWEST COST IS CHOSEN AS A NEW STARTING POINT. IF SOLUTION WAS USED AS A STARTING POINT PREVIOUSLY IN THIS SEARCH THE NEXT HIGHEST (IE. COST) SOLUTION IS USED
RECONSTRUCT STRUCTURAL MATRIX
OUTPUT COST DETAILS IF INTERMEDIATE PRINTOUT REQUESTED
DOES THE NUMBER OF ITERATIONS EXCEED THE MAXIMUM SPECIFIED ?
NO
YES
LAST OPTIMUM FOUND IN PRIMARY SEARCH IS ASSUMED TO BE GLOBAL
RETURN
DOES THE NUMBER OF ITERATIONS EXCEED THE MAXIMUM SPECIFIED ?
NO
YES
SOLUTION LESS COSTLY THAN OPTIMUM IS USED AS A NEW STARTING POINT. STRUCTURAL MATRIX IS REDUCED ACCORDINGLY
OUTPUT A WARNING AND SET ERROR FLAG
RETURN

# SUBROUTINE INTFES


START
DIMENSION ARRAYS DYNAMICALLY
IF ANY STIPULATIONS ARE NEGATIVE CHANGE SIGN AND SIGNS OF ELEMENTS OF CORRESPONDING ROW IN THE STRUCTURAL MATRIX
IS AN INITIAL SOLUTION SPECIFIED ?
NO
SLACK VARIABLES IN STRUCTURAL MATRIX ?
YES
NO
SET ERROR FLAG
RETURN
SET SLACK VARIABLES EQUAL TO THEIR CORRESPONDING STIPULATION
RETURN
YES
SOLUTION BASIC, FEASIBLE AND CONTINUOUS ?
NO
SET ERROR FLAG
RETURN
YES
REDUCE STRUCTURAL MATRIX TO CONFORM TO SPECIFIED SOLUTION USING THE FOLLOWING STRATEGY: 1) CHECK EACH SPECIFIED FLOWRATE AGAINST THE STIPULATIONS TO FIND TWO WHICH ARE EQUAL AND WHERE THE CORRESPONDING ELEMENT OF THE STRUCTURAL MATRIX IS EQUAL TO 1. 2) REDUCE MATRIX TO INCLUDE FLOW VARIABLE IN SOLUTION (IE. USE ROW OPERATIONS TO REDUCE ELEMENTS OF SAME COLUMN TO ZERO). 3) REPEAT 1) and 2) UNTIL ALL FLOW VARIABLES ARE INCLUDED IN THE SOLUTION
RETURN

# APPENDIX J

## EXAMPLE USING THE HYVRST ALGORITHM

The model application presented in this Appendix is similar to Appendix F except that the HYVRST routine is used in place of the ILP routine for optimization. The same water supply network is used, which is illustrated in figure F.1 and the corresponding data file is included in Appendix E (see page 133). The same cost routines are also used in this model application so that a comparison can be made in the solutions converged upon by the two optimization routines.

Only one model run is performed in this application, illustrated in figure J.1. An initial solution is specified which ~~is~~ identical to the one used in figure F.2(b). In this case, the solution converged upon is the global optimum which is confirmed through testing. The cost of finding the optimum is relatively high, 6.903 cp seconds when compared to the computer time used by the iterative linear programming algorithm (i.e. 2.33 cp seconds). The time differences in execution times may be more significant with more complicated cost subroutines. The optimal solution found in figure F.2(b) is 12% demonstrating that the solution found by the ILP approach is close to being optimal. The optimal solution cost is 7.207 x $10^6$ which is only 1.2% lower than the solution in figure F.2 (b) demonstrating that the solution found by the ILP approach is close to being optimal.

## FIGURE J·I — SAMPLE COMPUTER RUN — INITIAL SOLUTION SPECIFIED

```
              NETWORK SYSTEM OPTIMIZATION PROGRAM
              -----------------------------------
                        BY  A A SMITH
                        AND R H TUFGAR




IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                           (YES/NO)...NO

IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 , ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...LIST


INITIAL FLOW ASSUMPTIONS
U/S NODE   D/S NODE    FLOW
   1          0       10.70000
   2          0        4.30000
   3          0        3.00000
   4          0        6.50000
   5          0        5.00000
   6          0         .73000
   7          0        8.00000
   8          0        1.20000
   9          0        1.50000
  10          0        2.00000
  11          0        3.00000
  12          0         .00010
  13          0         .00010
COMMAND...ALTER


THE FLOW ARRAY CONTAINS PREVIOUSLY ASSIGNED
OR CALCULATED FLOW VALUES. DO YOU WISH TO
ZERO THE FLOW ARRAY ELEMENTS? (YES/NO)...YES
```

FIGURE J·I — (contd)

```
SUPPLY NUMBER OF FLOW CHANGES,
                (FORMAT I3)... 13


SUPPLY UPSTREAM NODE NUMBER, DOWNSTREAM NODE
NUMBER AND FLOW VARIABLE VALUE , (FORMAT I3,I3,F15.5)

CHANGE NUMBER   1...  1  6  10.7


CHANGE NUMBER   2...  6  8  8.0


LINK   6 TO   8 DOES NOT EXIST
PLEASE RESPECIFY

CHANGE NUMBER   2...  6  7  8.0


CHANGE NUMBER   3...  6 12  0.0001


CHANGE NUMBER   4...  6 11  1.2698


CHANGE NUMBER   5...  6  5  0.7001


CHANGE NUMBER   6...  3 11  1.7302


CHANGE NUMBER   7...  2 13  4.3


CHANGE NUMBER   8... 13  5  4.2999


CHANGE NUMBER   9...  4  8  1.2


CHANGE NUMBER  10...  4 10  3.5


CHANGE NUMBER  11... 10  9  1.5


CHANGE NUMBER  12...  3  0  1.2698


CHANGE NUMBER  13...  4  0  1.8

COMMAND...GO
```

FIGURE J·I — (contd)

```
IN ROUTINE "MODIFY" ALLOWABLE COMMANDS ARE:
GO......TO PROCEED
LINKS...TO DELETE OR RESTORE LINKS
NODES...TO DELETE OR RESTORE NODES
STIP....TO CHANGE STIPULATIONS
STATE...TO CHANGE NODAL STATES
BACK....TO GO BACK INTO "QINIT"
STOP....TO END RUN
COMMAND...GO



DO YOU WISH AN INTERMEDIATE PRINTOUT
IN THE SOLUTION ROUTINE,  (YES/NO)...YES



SUBROUTINE HYVRST
INTERMEDIATE SOLUTIONS
VARIABLE NO.     VALUE



INITIAL SOLUTION
     6           10.70000
    19            4.30000
    45            1.26980
    46            1.80000
    24             .70010
    10            1.26980
    28            8.00000
    40            1.20000
    37            1.50000
    41            3.50000
    12            1.73020
     4             .00010
    21            4.29990

COST=       7626857.00
```

FIGURE J·1 — (contd)

```
ITER       1
     6               9.43020
    19               4.30000
    43               1.26980
    46               1.80000
    24                .70010
    10                .00000
    28               8.00000
    40               1.20000
    37               1.50000
    41               3.50000
    12               3.00000
     4                .00010
    21               4.29990

COST=       7299787.00


ITER       2
     6               9.43020
    19               4.30000
    43               1.26980
    46               1.80000
    24                .70010
    10                .00000
    28               8.00000
    40               2.70000
    31               1.50000
    41               2.00000
    12               3.00000
     4                .00010
    21               4.29990

COST=       7268537.00


ITER       3
     6               9.43020
    19               4.30000
    43               1.26980
    46               1.80000
    24                .70010
    10                .00000
    28               8.00000
    40               4.70000
    31               1.50000
    35               2.00000
    12               3.00000
     4                .00010
    21               4.29990

COST=       7230676.00
```

FIGURE J·I — (contd)

```
ITER      4
       6                9.43020
      19                4.30000
      43                1.26980
      46                1.80000
      18                 .70010
      10                 .00000
      28                8.00000
      40                4.70000
      31                1.50000
      35                2.00000
      12                3.00000
       4                 .00010
      21                5.00000

COST=         7209137.00


OPTIMAL SOLUTION FOUND,
SEARCH CONTINUED AS A CHECK

ITER      1
       6                9.43010
      19                4.30000
      43                1.26980
      46                1.80000
      18                 .70010
      10                 .00000
      28                8.00000
      40                4.70000
      31                1.50000
      35                2.00000
      12                3.00000
       1                 .00010
      21                5.00000

COST=         7210314.00
```

FIGURE J·1 – (contd)

```
ITER       2
        6              9.43020
       19              4.30000
       43              1.26980
       46              1.80000
       18               .70010
       10               .00010
       28              8.00000
       40              4.70000
       31              1.50000
       35              2.00000
       12              3.00000
        8               .00010
       21              5.00000

COST=        7211116.00


ITER       3
        6              9.43010
       19              4.30000
       43              1.26980
       46              1.80000
       18               .70010
        1               .00010
       28              8.00000
       40              4.70000
       31              1.50000
       35              2.00000
       12              3.00000
        8               .00000
       21              5.00000

COST=        7210314.00


ITER       4
        6              9.43020
       19              4.30000
       43              1.26980
       46              1.80000
       18               .70020
       16               .00010
       28              8.00000
       40              4.70000
       31              1.50000
       35              2.00000
       12              3.00010
        8               .00010
       21              5.00000

COST=        7212323.00
```

FIGURE J·I — (contd)

```
ITER       5
      6                 9.43020
     19                 4.30000
     43                 1.26980
     46                 1.80000
     18                  .70010
      4                  .00010
     28                 8.00000
     40                 4.70000
     31                 1.50000
     35                 2.00000
     12                 3.00000
      8                  .00000
     21                 5.00000

COST=        7209137.00


ITER       6
      6                 9.43020
     19                 4.30000
     43                 1.26980
     46                 1.80000
     18                  .70010
     14                  .00010
     28                 8.00000
     40                 4.70000
     31                 1.50000
     35                 2.00000
     12                 3.00010
      8                  .00010
     21                 5.00000

COST=        7212942.00


ITER       7
      6                 9.43020
     19                 4.30000
     43                 1.26980
     46                 1.80000
     24                  .70010
     14                  .00010
     28                 8.00000
     40                 4.70000
     31                 1.50000
     35                 2.00000
     12                 3.00010
      8                  .00010
     21                 4.29990

COST=        7234481.00
```

# FIGURE J·I — (contd)

```
ITER      8
     6                 9.43020
    19                 4.30000
    43                 1.26980
    46                 1.80000
    24                  .70010
     4                  .00010
    28                 8.00000
    40                 4.70000
    31                 1.50000
    35                 2.00000
    12                 3.00000
     8                  .00000
    21                 4.29990

COST=      7230676.00


ITER      9
     6                 9.43010
    19                 4.30000
    43                 1.26980
    46                 1.80000
    24                  .70010
     1                  .00010
    28                 8.00000
    40                 4.70000
    31                 1.50000
    35                 2.00000
    12                 3.00000
     8                  .00000
    21                 4.29990

COST=      7231853.00


ITER     10
     6                 9.43020
    19                 4.30000
    43                 1.26980
    46                 1.80000
    24                  .70010
    10                  .00010
    28                 8.00000
    40                 4.70000
    31                 1.50000
    35                 2.00000
    12                 3.00000
     8                  .00010
    21                 4.29990

COST=      7232655.00
```

## FIGURE J·I — (contd)

```
 ITER      11
           6              9.43020
          19              4.30000
          43              1.26980
          46              1.80000
          24               .70020
          16               .00010
          28              8.00000
          40              4.70000
          31              1.50000
          35              2.00000
          12              3.00010
           8               .00010
          21              4.29980

COST=         7233862.00


 ITER      12
           6              9.43020
          19              4.30000
          43              1.26980
          46              1.80000
          26               .70020
          16               .00010
          28              8.70020
          40              4.70000
          31              1.50000
          35              2.00000
          12              3.00010
           8               .00010
          21              4.29980

COST=         7267843.00


 ITER      13
           6              9.43020
          19              4.30000
          43              1.26980
          46              1.80000
          26               .70010
           4               .00010
          28              8.70010
          40              4.70000
          31              1.50000
          35              2.00000
          12              3.00000
           8               .00000
          21              4.29990

COST=         7264651.00
```

FIGURE J·1 — (contd)

```
OPTIMAL SOLUTION FOUND



                   PROBLEM SOLUTION
  PROCESSING CENTER COSTS
      CENTER                    FLOW        COST (MIL)

 CLEARWATER LAKE                 9.43            .54
 CRYSTAL CREEK                   4.30            .30
 SULPHUR SPRINGS                 3.00            .23
 DEAD MAN'S POND                 4.70            .32


  TRANSPORTATION COSTS
 NODE        ROUTE                FLOW      COST (MIL)

   1  CLEARWATER LAKE      TO
    6 BRACKWELL                   9.43          1.23
   3  SULPHUR SPRINGS      TO
   11 DRYGULCH                    3.00           .21
   6  BRACKWELL            TO
   13 JUNCTION TWO                 .70           .18
   2  CRYSTAL CREEK        TO
   13 JUNCTION TWO                4.30           .76
  13  JUNCTION TWO         TO
    5 THIRSTYVILLE                5.00           .21
   6  BRACKWELL            TO
    7 NEWTON                      8.00          1.31
   8  BOGTON MOOR          TO
    9 SALTWELL FLATS              1.50           .46
   8  BOGTON MOOR          TO
   10 FOSSETTVILLE                2.00           .40
   4  DEAD MAN'S POND      TO
    8 BOGTON MOOR                 4.70          1.08


         PROCESS COST...... (MILLION)     1.384
         TRANSPN COST...... (MILLION)     5.823
         TOTAL SYSTEM ..... (MILLION)     7.207
  END OF REPORT




IN SUBROUTINE "CNSTIN"
DO YOU WISH TO REDEFINE ANY CONSTANTS
                            (YES/NO)...NO
```

FIGURE J·1 — (contd)

```
IN ROUTINE "QINIT"
PROGRAM HAS AUTOMATICALLY CHANGED ANY ZERO STIPULATIONS
TO +/-  .00010 . ZERO STIPULATIONS ARE NOT ADVISABLE.
ALLOWABLE COMMANDS ARE:
GO.....TO PROCEED
LIST...TO LIST NONZERO FLOWRATES
ALTER..TO CHANGE INITIAL FLOWRATES
SOLVE..TO OBTAIN A PROGRAM SOLUTION USING
       THE FLOWRATES PRESENTLY CONTAINED
       IN THE FLOW ARRAY
STOP...TO END RUN
COMMAND...STOP

 END OF PROGRAM
:
```

# APPENDIX K

## SUMMARY OF NOTATION

This appendix summarizes the notation utilized throughout the previous chapters and appendices. Generally, the definition of each variable is unique, however, a small number of variables have similar meaning. The majority of variables are included except for a small number which appear in the text on a one time only basis and have interpretations that are self explanatory.

$\bar{A}$ = Structural matrix containing the coefficients of the flowrate variables in the constraint equations.

$a$ = Element of the structural matrix $\bar{A}$.

$\bar{B}$ = Vector of stipulations from the constraint equations.

$b$ = Element of the stipulation array $\bar{B}$.

C( ) = Cost function for computing solution costs.

C = Specified or computed cost value, or a cost coefficient of a given flowrate Q.

CP( ) = Cost function for computing the costs for a specific processing facility.

CP = Specific or computed processing cost.

CPEN( ) = Penalty cost function.

CT( ) = Cost function for computing the costs for transportation.

CT = Specific or computed transportation cost.

$\bar{D}$ = Revised structural matrix (columns reordered).

D = Stipulation value associated with a nonprocessing node (demand).

DHP = Design horsepower of a pumping installation.

DIA = Pipeline diameter.

Eff = Underutilization ratio applied to the processing rate of processing facilities.

G( ) = Function representing a constraint equation.

G = Number of subgraphs or nonconnected networks.

$h_f$ = Head losses in a pipeline due to friction.

$h_p$ = Pumping head introduced from a pump installation.

IHP = Installed horsepower of a pumping installation.

L = Number of conveyance or transportation links.

$\ell$ = Number of violated constraints.

M = Direction vector component.

m = Accuracy limitation on flowrate Q.

MILEAGE = Transportation vehicle miles travelled.

N = Number of nodal points in a system.

NCEN = Number of processing nodes.

NQ = Number of design flowrate variables = $2 . L + N$.

$\bar{P}$ = Selected column vector from the structural matrix $\bar{A}$.

$\bar{Q}$ = Vector of flowrate variables (solution vector).

Q = Design flowrate variable representing the conveyance of material between two nodes.

QDES = Design flowrate modified to reflect the underutilization of a processing facility.

QI = Operating flowrate (per year).

R = Region containing the feasible solution set.

S = Stipulation associated with a processing node (supply).

STDS = Nodal state at the downstream end of a transportation link.

STUS = Nodal state at the upstream end of a transportation link.

T = The amount of material transshipped through a node.

TONNAGE = Tonnage rating of a fleet of vehicles for conveying material.

VEL = Conveyance velocity in a transportation link.

X = Transformed flowrate variable.

XL = Representative length value of a transportation link.

Ƶ( ) = Objective function (total system costs).

Ƶ = Objective function value (total system costs).

$\alpha$ = Coefficient with a value between 0 and 1.

$\varepsilon$ = EPS = Perturbation value representing a flowrate insignificant to a network problem.

$\gamma$ = Specific weight of water (62.4 $\text{lbs/ft}^3$).

$\theta$ = Variable used to represent the value of a nonsolution variable entering the solution.