LINEAR PROGRAMMING AND DYNAMIC PROGRAMMING APPLICATION TO WATER DISTRIBUTION NETWORK DESIGN

by
John C. Schaake, Jr.
and
Dennis Lai

HYDRODYNAMICS LABORATORY
Report No. 116

Prepared with the support of
New York City Bureau of the Budget
M.I.T. Sloan Basic Research Fund

July 1969

MIT

DEPARTMENT
OF
CIVIL
ENGINEERING

SCHOOL OF ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts 02139

R69-69

HYDRODYNAMICS LABORATORY
Department of Civil Engineering
Massachusetts Institute of Technology

LINEAR PROGRAMMING AND DYNAMIC PROGRAMMING APPLICATION TO WATER DISTRIBUTION NETWORK DESIGN

by
John C. Schaake, Jr.
Dennis Lai

July 1969

Report No. 116

The work upon which this publication is based was supported by funds provided by the New York City Bureau of the Budget and by the MIT Sloan Basic Research Fund.

ABSTRACT

The water distribution network design problem is to find the optimal set of investments in pipelines that are needed to satisfy water requirements. The strategy of this study has been first to define an optimality criterion for ranking alternative investment opportunities and then to formulate a mathematical programming model for solving the optimal investment problem. The least cost optimality criterion leads to a non-linear mathematical programming problem for which no computational methods exist that guarantee an optimal solution. Other existing techniques that yield "good" solutions are computationally inefficient.

The strategy taken in this study has been to modify the least cost problem so that linear programming could be applied to achieve a solution to the modified form of the problem. Variables were transformed to linearize the non-linear terms in the pipe flow formula. In this way, the non-linear flow phenomenon is represented exactly. The resulting linear programming model may be used to determine the pipe diameters of pipes that must be added to the system to satisfy given sets of water requirements that are expected to occur at a given future time.

Water requirements increase with increases in population and economic productivity. To meet these growing requirements, excess capacity must be provided. The problem of deciding how far into the future the system should be planned is known as a capacity expansion problem. The capacity expansion problem has been formulated as a dynamic programming problem and applied to the water distribution network expansion problem.

ACKNOWLEDGEMENT

The material presented here was developed in a program of applied research in engineering systems analysis into the primary water distribution system of the City of New York. The project was sponsored by the Bureau of the Budget of New York City in Connection with a proposed five stage, billion dollar addition to the existing network of deep rock large diameter tunnels. The purpose was to use the principles of engineering systems analysis to determine the characteristics of the most economically effective design for large-scale water distribution systems. Additional support was provided for developing new systems analysis techniques by an M.I.T. Sloan Basic Research Grant.

The project was executed in the M.I.T. Urban Systems Laboratory under the leadership of Dr. Richard de Neufville, Dr. Peter S. Eagleson, Dr. John C. Schaake, Jr., Dr. Joseph Stafford, and Dr. Frank Perkins, and supported by B. Bayer, J. Dubinsky, J. Hester, M. Hester, D. Lai, W. Maddaus, and D. Picard, as well as the assistance of R. Arnott, P. Messeri, N. Scanlan, K. Swartz and S. Vick.

The members of the project wish to express their appreciation for the support provided by Mr. David Grossman, Deputy Director, New York Bureau of the Budget; Professor C. L. Miller, Director, Urban Systems Laboratory; Commissioner Maurice Feldman, of the Department of Water, Sewers, Gas and Electricity; and Mr. Vincent Terenzio, of the New York City Board of Water Supply.

July 1969

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Chapter 1

Introduction

The estimated annual investment for water distribution systems in the United States is $1.5 billion or $7.50 per person (ASCE, 1969).[1] The most costly parts of most water supply systems are the distribution facilities, which include pipe networks as well as the pumping and storage components. These complex systems must be designed to satisfy a multitude of criteria imposed by many different water users, ranging from lawn sprinkling and fire fighting to the various industrial and domestic needs.

To design facilities to serve these diverse needs at minimum cost is a challenging goal. It is evident that the economically efficient allocation of resources to water distribution facilities is unlikely without systematic, objective, and computationally efficient design methodologies.

A complete consideration of a water distribution system design should consist of the following items: (i) diameters and head losses for all pipe elements (ii) multiple sources of supply (iii) pumping stations (iv) elevated and ground storage reservoirs. An adequate design, while giving least cost, should meet possible different demand patterns which represent different times of day, various fire flow requirements and special industrial uses.

Summary

A principal objective is to present a Linear Programming Formulation of the optimal network design problem. Since this problem may be shown to involve a concave nonlinear minimum cost objective function subject to linear constraints, many local optima exist making the global optimum difficult to find. For practical purposes, approximate formulations which eliminate the concave non-linearities should be useful. One major benefit is the insight into the design problem which may be gained. Another benefit is that "good" solutions may be found although they are not likely to be optimum according to the original objective function. Important information may also be supplied by the dual solution indicating the binding constraints on the system and giving the marginal costs associated with increasing the constraint levels.

In the next chapter, the linear programming model is presented. Since water systems are usually constructed to supply growing water demands, the time sequence of possible capacity expansions to meet growing demands is considered in Chapter 3 in the context of a Dynamic Programming application. This Dynamic Programming-Capacity Expansion model treats the system as a "lumped" system. In Chapter 4 the Linear Programming model is used as a suboptimization model to produce designs under various conditions specified by the Dynamic Programming Model. This represents an initial attempt to state the network design problem as a capacity expansion problem. Much more work is needed to test the limitations of the joint DP-LP model. Additional work also is needed to study more completely the properties of the Linear Programming model.

Literature Review

A brief review of the work of most of the previous investigations on water distribution system analysis and design has been given by Pitchai[(2)] in 1966, so the objective here is to extend that review to include investigations being reported since then. The following review of recent literature has attempted to identify contributions both to analysis and design aspects. Analysis and design are essentially different approaches. In the former the focus is on understanding or evaluating; whereas, in the latter, the focus is on making decisions.

In a relative narrow sense, we may call a distribution system a design problem if the pipe diameters are unknown and are to be determined. In such cases, there usually exist a number of solutions which satisfy the specified design criteria. The engineering practice is that the solution which gives the least cost (or maximal benefit) is chosen.

If the set of pipe diameters is given, then the distribution problem becomes an analysis problem. The analysis objective is to determine for each node, the pressure, and to determine for each pipe, the flow magnitude and direction. These flow conditions must satisfy the following physical laws of the network:

(i) the algebraic sum of head loss around each loop must be zero;
(ii) flow into each node must equal flow out of the node; and
(iii) the proper relation between head loss and discharge must be maintained for each pipe.

It has been demonstrated by Pitchai that there exists an unique solution to the analysis problem.

Contributions to Water Distribution System Analysis

The preponderance of past work on distribution system has concentrated upon solving the non-linear equations that describe their hydraulic behavior. The pipe diameters are given, and the problem is to solve for unknowns which in general may be pressure or consumption at the nodes or the resistance of the pipe. It can be shown that, for a specific network with known consumptions, the problem is exactly determined, i.e., there are as many equations as the unknowns. Because of the nonlinearity of the equations, the solution is achieved by successive iterations using a suitable scheme which achieves convergence. If the consumptions at the nodes or the resistances of the pipes are not all known, other information should be adequately supplied so that the problem is reduced to an exactly determined problem. In general, the methods of system analysis employed would depend upon the types of unknowns existing in the distribution system.

The most recent and significant work on water distribution system analysis was presented by Shamir and Howard(3) in 1968. They applied the Newton-Raphson method to balance networks under very general steady-state flow conditions. The Newton-Raphson technique is a root-finding process which finds new improvements or corrections to the values of the unknowns in each iteration. The improvements or corrections are computed from the linearlized Taylor Series expansion, evaluated at the present state of the solution. The network problem considered by Shamir and Howard may contain pipes, pumps, valves, etc., and unknowns may be combinations of pressures, consumptions or element resistances. Governing equations used are the continuity equation at each node, so there are N equations assuming there are N nodes. One can then solve for N unknowns.

The Newton-Raphson method deals with the whole network at the same time so that corrections are made simultaneously in order to account for the joint interaction of all corrections. This method takes into account the effect of changing any one variable (pressure, consumption or resistance) on the entire network. This latent sensitivity information mades the Newton-Raphson method particularly useful for design purposes also.

Like any iteration procedure to solve nonlinear equations, the Newton-Raphson method may encounter convergence problems. In this case, the mathematical criteria for convergence for all possible combinations of unknowns have not been established. Therefore, it is not now possible to test, a priori, for convergence of hydraulic network analysis by the Newton-Raphson method. It has been observed by de Neufville et al.[(4)] (1969), that "divergence may occur if a particular pipe in a network is especially smaller than the others." As this particular small pipe was artificially made larger, the divergence problem was eliminated. It has been the experience of some investigators (Warga,[(5)] 1954; Pitchai,[(2)] 1966) that a good starting guess will usually lead to a solution.

Probably the method most commonly used for balancing a hydraulic network is the Hardy Cross method. The method is well suited for solution by hand and is easily adapted for machine computation. The method can be approached either by balancing flows or by balancing pressures. Both the Hardy Cross and Newton-Raphson methods solve the nonlinear equations by iterations. The Hardy Cross performs iterations on separate equations one at a time, which requires small amounts of computer storage but may need excessive computation time for a large network. Also, the Hardy Cross method may not converge when a network contains some large pipe of short length and relatively small flow (Dillingham,[(6)] 1967). Some procedures, such as using linear formula between discharge and head loss when the head loss is less than 1 foot, have been developed to handle this difficulty. But still, there is no guarantee of convergence. In addition, the Hardy Cross method does not readily provide a sensitivity analysis.

Contributions to Water Distribution System Design

There never has been a comprehensive study to develop methods for optimal design of pipe network, pumping and storage facilities. Efforts have been devoted either to the optimal design techniques of pipe network and pumping facilities (Pitchai,[1] 1966; Jacoby,[7] 1968) or to economical trade-offs between the booster pumping with ground storage reservoir and elevated storage (McPherson,[8] 1966). Since in the proposed method of approach, storage costs are not considered, review of the literature will be concentrated on work related to pipe network design, including pumping facilities.

A few of the many alternate methods of formulating a minimum cost design objective have been explored. A notable study was completed by Pitchai[1]. He formulated the design problem as a non-linear integer programming problem which he solved with a random search technique. Cost of pipes and annual cost of energy used are included in the objective function to be minimized.* Constraints may be imposed, such as: minimum permissible pipe sizes; maximum permissible head loss along a specified

---

*Not all energy costs appear to have been accounted for since the energy costs are taken as the sum of all of the energy losses in the network. This omits accounting for the energy released to the consumer as potential energy associated with the pressures at the demand nodes.

path; and operating pressures to coincide with characteristic curves of pumps. The constraint on maximum permissible head loss along a specified path was used to augment the objective function as a penalty function.

A Newton-Raphson method was used for balancing the network. The optimum was sought by a sequential, random sampling scheme. The processes began with an initial guess of design diameters which served as a so-called central design. That design was subsequently analyzed with the Newton-Raphson method. By the defined cost function, the system cost for that particular design was computed. The next step was to generate randomly a set of designs about the central design. Then, the corresponding design costs were determined, and the best design among them was selected to serve as the central design of next random cast. The results given show that the system cost decreases with the number of casts, but there is no proof that the global optimum is found. Large amounts of computer time are required. For example, the computer time required for a design with one demand pattern for a 25-loop network with only a single source of supply and no other pumps or reservoir was 8 minutes on an IBM 7094 computer. To study marginal sensitivity of design to the constraints, the constraints must be changed and new designs run.

A design with multiple demand patterns was also considered. The design procedures follow nearly the same way as when there is only one demand pattern; except that for any set of design pipes, there will be a system cost corresponding to each demand pattern. The largest one among the different demand patterns is chosen as the representative system cost. Moreover, the minimum cost among the alternate designs is taken to be the total system cost. The design from this minimax approach appears to be very sensitive to the penalty assigned to the violation of maximum allowable head loss along specified paths. If the penalty imposed is very large, i.e., no violation of maximum allowable head loss is allowed, one can anticipated that many trial and error processes are required to get a set of feasible design diameters. No discussion on this matter was given.

The second recent study (Jacoby,[7] 1968) is very similar to the previous study by Pitchai. One difference is that constraints were stated as inequalities by Jacoby, in contrast to Pitchai's equality constraints. The cost function and the constraints were combined to form a "merit function" from which Jacoby sought the optimum by a gradient-random search iteration method. After the continuous solutions were obtained, they were rounded to the nearest integer solutions. If these round-off results were not feasible, the Hardy Cross method was applied to eliminate this infeasibility. Because the objective function has many local optima, the technique does not assure that the global optimum will be found. The author of the method advises that "caution should be used to avoid local minima." To study sensitivity of the design to variation in either parameters or constraints requires changing these and again running the program. In the paper, information on the computational efficiency is not presented and an important question of handling multiple loadings is not considered.

One common characteristic of these two existing optimal design methodologies is that they use iterative search techniques to seek the optimum designs. A major disadvantage of these is the relatively high cost of the required computations compared to possible costs of other more direct methods and the method to be proposed here. Moreover, in the two methods proposed above, there is no assurance that better designs do not exist than the designs which are considered as optimum.

In another recent study, Karmeli et al. (1968)[9] studied a simple branched network, i.e., a network without loops, with only one source of supply, and fomulated the design problem as a linear programming problem. Because the network is branched, the discharge that each pipe will carry can be computed. The diameters taken into account for each pipe are determined in advance. As a result, the friction loss per unit length for each diameter to be considered can be computed. The decision variables are the piezometric head at the sources and the length of each predetermined diameter to be allocated to each branch of the tree-shaped network. The constraints are the total length of each pipe and the minimum allowable piezometric head at each node. The method does not allow for multiple demand patterns.

One other study was made by Smith (1966),[10] who used a random search-steepest descent method to begin to explore the response surface, followed by a linear programming procedure to guide the solution toward an optimum. The constraints were specified as linear equalities. The objective function was similar to that used by Pitchai and appears to have the same discrepancy with respect to energy costs. Multiple demand patterns were also accounted for.

Chapter 2

LINEAR PROGRAMMING PIPE NETWORK OPTIMIZATION

Introduction

It appears possible to formulate the minimum cost network design problem as a linear programming problem. This is an attractive approach due to its computational efficiency and because its solution promises to give valuable insights to the sensitivity analysis of demands to the total system cost. In this chapter, the linear programming problem will first be formulated for design of a new distribution system for a single demand pattern with possible multiple sources of supply. To this model, the capability will be added for designing new additions to an existing pipe network system. Finally, additional generalization to take care of the multiple demand pattern will be considered. Problems involving multiple demand patterns have been formulated, but no computational experience exists for multiple demand patterns at this time.

The functions of the LP computer programs will subsequently be described, and the results of an example using the LP program will be given. Assumptions will be stated when they are made. Listings of all computer programs are given in the appendix; it should be understood these are experimental programs which have evolved out of this research effort.

As this model is presented, it will clearly be shown that the network cost minimization problem is essentially a non-linear problem. By means of variable transformations, all constraints may be linearized, but the objective function remains non-linear. This could be dealt with conveniently if the non-linear cost functions were convex. Unfortunately, this function is concave so it becomes extremely difficult to find the global optimum since many local optima may be shown to exist in this case. No technique ever applied to the water distribution system optimization problem can be claimed to have "solved" this problem in the sense that the global optimum is given with certainty. For example, Pitchai[(1)] and Jacoby[(7)] both used random search techniques which do not necessarily lead to the global optimum.

Since there is no assurance the original non-linear problem is solved by any existing technique, it seems that other formulations of the problem may also have practical value. In many systems problems other than water distribution systems, it has often been worthwhile to create a linear version of the problem, even when the problem is not basically linear, in order to gain the insights that linear formulations are known to give. Thus, one of the important merits of a linear-programming-water-distribution-model is an improved understanding of water distribution design which may result from these insights. The value of these kinds of insights was appreciated by the noted mathematician, Hamming,[(11)] who prefaced his famous book on numerical analysis with the statement, "The purpose of computing is insight, not numbers." This might also be restated in terms of systems analysis or mathematical programming. Hopefully, the numerical results from linear programming models of water distribution systems may also be useful for many applications.

Linear Programming Model for One Demand Pattern

Constraints for Each Node. The formulation follows from the observation that, at each node in the network the relation

$$\sum_i K_{ij} d_{ij}^{\ p} \leq -Q_j \qquad j=1,\ldots,m \qquad (2\text{-}1)$$

must be satisfied. The index j indentifies a specific node and i identifies a neighboring node; d_{ij} is the pipe diameter between nodes i and j; m is the total number of nodes; p is a constant whose value is approximately 2.5, and Q_j is the demand or supply rate at node j. The sign convention is that flows into the nodes are considered negative and flows out, as positive. The left hand side of Eq. 2-1 represents the algebraic sum of flows in pipes connecting to node j. K_{ij} is a measure of the potential for conveying water between the nodes i and j and can be expressed functionally as

$$K_{ij} = g(f, H_{ij}, L_{ij}) \qquad (2\text{-}2)$$

where the sign of K_{ij} is the same as that of H_{ij}, the head loss between the nodes i and j. The terms f and L_{ij} represent the friction coefficient and pipe length, respectively. The Hazen-Williams formula, which is commonly used for water distribution studies, was adopted to relate the pipe discharge and head loss. Accordingly,

$$p = 2.63$$

and

$$K_{ij} = 6.2 \times 10^{-4}\ C_{HW} \left(\frac{H_{ij}}{L_{ij}}\right)^{0.54} \qquad (2\text{-}3)$$

This gives discharge in cubic feet per second if d_{ij} is in inches and H_{ij}, L_{ij} are in feet.

For a given pattern of demands imposed on the system, there usually is a minimum pressure at each node that must be maintained. This depends on the topographic elevations of the distribution system service area and the residual energy that is required by codes imposed by fire insurance underwriters or by requirements for normal operation purposes. Determining these pressures requires sound engineering judgment and must be done as a step in the design process. The actual operating pressures will generally exceed these minimum pressures. Designing the system to give adequate operating pressures should consider the economics of allocating pressure losses throughout the system.

In this LP model, the operating pressure at each node must be specified in advance of computing the optimum pipe diameters. This must be done so that the quantity H_{ij} is defined and may be used to compute the magnitudes of the elements in the LP coefficient matrix. It follows, therefore, that the LP model does not explicitly yield the optimum operating pressures throughout the system and that the optimum diameters which are given are related to the specific pressure pattern associated with the set of values of H_{ij}.

Analysis of the sensitivity of the cost of water distribution systems to various parameters shows that cost is relatively insensitive to pressure loss so that some variation from the true optimal operating pressure should be acceptable. Moreover, near the optimal setting of any unconstrained decision variable, small changes may be made in the decision variables without affecting total costs. Operating pressures are often constrained by required pressures at the extremities and are essentially unconstrained in the interior of the system since the total head loss along a given path is fixed. To see if a given head loss distribution along different paths is near optimal, the LP model can be used successively for different distributions. If small changes in the pressure pattern have little effect on cost, the pattern is near optimal.

The economics of allocating pressure losses is discussed in an appendix. It provides an algorithm for allocating the pressure loss along a pipeline, given the total pressure losses between the source and the extremities. The total cost of pipelines alone is minimized, and the algorithm is applicable only for a network which does not have a loop path. The pressure head at each node may be determined by the algorithm and the whole network system may then be designed using the heads obtained. Proposed tree-shaped networks so designed will be optimal in the least cost sense.

The node equations, Eq. 2-1, represent a set of m constraints, assuming there are m nodes. Assume there are n pipes where n is usually greater than m. For $n > m$, the implicit function theorem states that the diameters of m of the pipes can be expressed in terms of sizes arbitrarily assigned to the remaining $n - m$ pipes. A unique solution, therefore, does not exist, so it is meaningful to seek a minimum cost solution.

Objective Function. Considering first the capital cost of the installed pipe, the cost per linear foot of pipe is approximately (Linaweaver *et al.*, 1964)[12]

$$C_{ij} = \alpha d_{ij}^{1.3} \qquad (2\text{-}4)$$

where $\alpha = .36$.

For tunnels, the cost per linear foot is approximately

$$C_{ij} = 1.1\, d_{ij}^{1.24} \tag{2-5}$$

The unit of d_{ij} is inches. The cost expressed by Eqs. 2-4 and 2-5 has considered the cost of the land, pipelines, and the costs of operation and maintenance. Eqs. 2-4 and 2-5 are the result of cost analysis over 50 oil, gas and water pipelines and about 20 tunnels. The costs given here are based on an ENR cost index = 877. In engineering optimization problems, estimating precise cost coefficients for each variable is usually difficult. Since it is felt that there is no other representative formula, Eqs. 2-4 and 2-5 are used throughout this report for pipe and tunnel costs. For clarity and convenience, from now on, the index i shall denote pipes and index j shall denote nodes. Moreover, there shall always be n pipes and m nodes. The total cost of all pipes in the network is then

$$C_p = \sum_{i=1}^{n} \alpha\, L_i d_i^{1.3} \tag{2-6}$$

Consider next the power cost. It seems clear that the cost of energy required for pumping, which may be accounted for partly as loss of head due to friction and partly as residual energy discharged as pressure energy to the user, may constitute an important component of the total system cost. This cost can be expressed as

$$C_e = a \left[\sum_{i}^{\text{pipes}} q_i h_i + \sum_{j}^{\substack{\text{demand}\\ \text{nodes}}} Q_j H_j \right] \tag{2-7}$$

where a is a constant to account for the price of a unit quantity of energy, the duration of pumping, units conversions, and pumping efficiency.

Also, q_i = flow in pipe i

h_i = head loss in pipe i

Q_j = demand at node j

H_j = residual energy head at node j

Expressing q_i in terms of d_i by the Hazen-Williams formula, Eq. 2-7 becomes

$$C_e = \sum_i^n a_i' d_i^P + \sum_j^{\text{demand nodes}} aQ_jH_j \tag{2-8}$$

where a_i' depends, in part, on a and, in part, on the other terms besides d_i in the Hazen-Williams Formula. Adding this pumping cost to the total capital cost for pipes, the objective function becomes

$$C = \sum_i^n \alpha L_i d_i^{1.3} + a_i' d_i^P + \sum_j^{\text{demand nodes}} aQ_jH_j \tag{2-9}$$

Because the required pressure H_j at node j is specified and is not a decision variable, the third term in Eq. 2-9 is a constant. It has no effect in obtaining an optimal solution. Thus, we can drop it during the optimization process but we should consider it to get the actual total system cost. If H_j were a decision variable, this term should remain in the objective function.

<u>Non-linear Programming Model</u>. The network design problem has now been formulated as the following non-linear programming model:

$$\text{Min } C = \sum_i^n \alpha L_i d_i^{1.3} + a_i' d_i^P \tag{2-10}$$

subject to

$$\sum_{i \in s_j} K_i d_i^{\,p} \leq -Q_j \qquad , \ j=1,\ldots,m \qquad (2\text{-}11)$$

where s_j is the set of pipes connecting to node j, and

$$d_i \geq 0 \qquad i=1,\ldots,n$$

Both the objective function, Eq. 2-10, and the constraints, Eq. 2-11, are nonlinear. It is to be noted that the pipe diameters, the decision variables, are continuous variables in this model. Future investigation should take the discrete set of available commercial diameters into cosideration.

Linear Programming Model via Variable Transformation. It appears possible to approximate the nonlinear optimization model by a linear programming model. Substituting the relation

$$x_i = d_i^{\,p} \qquad (2\text{-}12)$$

into Eq. 2-11, we obtain the linearized constraint equations

$$\begin{aligned} \sum_{i \in s_j} K_i X_i &\leq -Q_j \qquad j=1,\ldots,m \\ X_i &\geq 0 \qquad i=1,\ldots,n \end{aligned} \qquad (2\text{-}13)$$

The objective function, Eq. 2-10, can be rewritten as

$$\text{Min } C = \sum_{i}^{n} \alpha L_i d_i^{\,1.3/p} + a_i{}' X_i \qquad (2\text{-}14)$$

which is nonlinear because of the first terms. Since the first terms in Eq. 2-14 contain the only nonlinear terms remaining in the model, examine these in more detail.

By the Hazen-Williams formula, p takes the value of 2.63. Eq. 2-4 can be written as

$$C_i = \alpha X_i^{1.3/p} \simeq \alpha X_i^{1/2} \tag{2-15}$$

where C_i is the cost per unit length of pipe with sizes d_i in inches. A linear approximation of Eq. 2-15 is

$$C_i = \alpha X_i^{1/2} \simeq \beta_i' + \beta_i X_i \tag{2-16}$$

where, as shown in Fig. 2-1, β_i' and β_i are respectively the intercept and the slope of the straight line which approximates the curve of Eq. 2-15 within the range of variables between X_i and X_{i+1}. Both β_i' and β_i are functions of X_i. Based on this linearization, the objective function is redefined as

$$\text{Min } C = \sum_i^n (\beta_i' L_i + L_i \beta_i X_i) + \sum_i^n a_i' X_i \tag{2-17}$$

Since the range of possible pipe sizes in a network may be too large to justify a single linear function in place of the non-linear cost function, a piece-wise linear function must be used. Any one pipe, however, is expected to fall into a certain class of pipe sizes before the design is run. On this basis, a single linear function for each individual pipe is used.

When the LP run is made, the classes for pipes are changed if a pipe does not fall in the proper range. The LP model then is rerun until the optimal solution shows that the classes of pipe sizes are correctly related to the pipe sizes. This procedure does not assure that the global optimum of the non-linear programming model is reached. In terms of the non-linear programming model, a local optimum may be reached by this procedure.

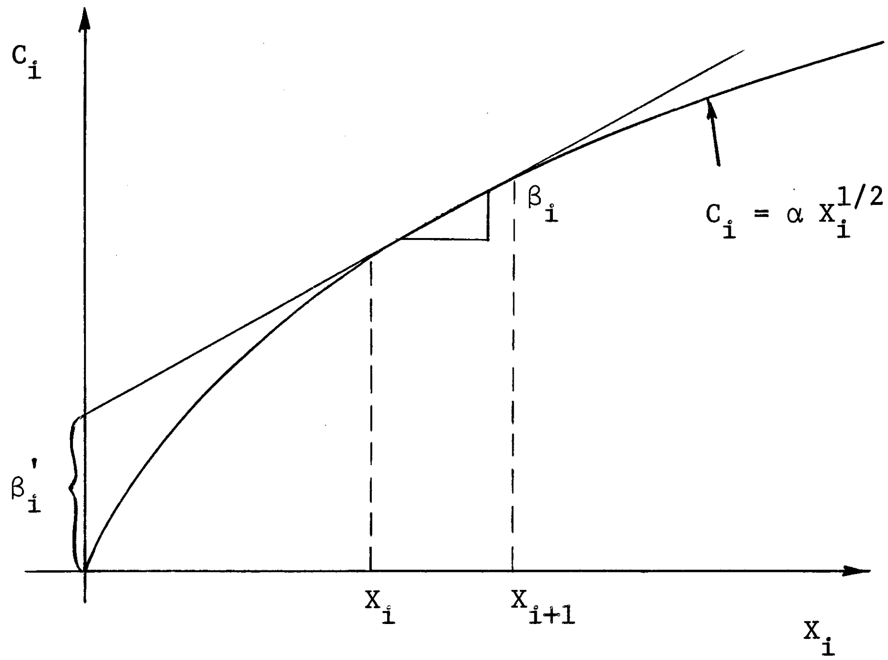


Figure 2-1

Linealization of Cost Function

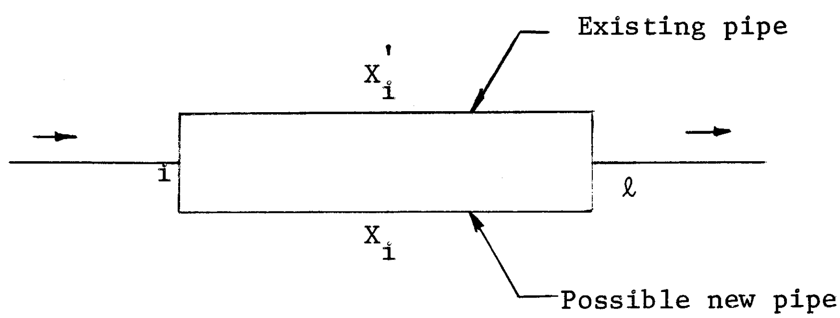


Figure 2-2

Consideration of Existing Pipe

Thus, the following linear programming problem has been developed:

$$\text{Min } C = \sum_{i=1}^{n} L_i \beta_i X_i + \sum_{i=1}^{n} a_i' X_i \tag{2-18}$$

S.T.

$$\sum_{i \varepsilon s_j} K_i X_i \leq -Q_j \quad , j=1,\ldots,m$$
$$X_i \geq 0 \quad i=1,\ldots,n \tag{2-19}$$

It does not necessarily seek the minimum cost design of the non-linear model. The resultant design, nevertheless, would appear to be a "good" design and conceivably could be more desirable from a practical point of view than the original non-linear minimum cost design. This is because there exists no algorithm which guarantees to obtain the global optimum of the original non-linear optimization problem. Very often, the so-called optimal design is only one of the local optima which may not give a design as good as the one by linear programming problem formulated. In addition, there always exists a wide range of uncertainty in designating cost coefficients. Since the non-linear cost function is concave, the linear programming model will actually tend to treat small pipes preferentially to larger pipes since economics of scale are neglected.

Another point worth mentioning is that a demand constraint at each node may be treated as an equality, rather than an inequality, constraint. Because the cost function is monotonically increasing with respect to the diameters, to supply more water than is needed will tend to increase pipe sizes of the system. It is then conceivable that the constraints corresponding to demand nodes will always be binding, i.e., will have equality constraints instead of inequalities. The constraints corresponding to supply nodes may or may not be binding depending on whether the total amount of supply capacity is equal to or greater than the total demands on the system.

It is well known for any non-degenerate basic feasible solution of a linear programming problem with m constraints and n decision variables, that only m of the n variables have non-zero values. Therefore, n-m pipe sizes must be zero so there are only m distinct pipes in the optimal network. Moreover, the problem is non-degenerate if the m constraints are linearly independent. Because the actual total amount of supply should equal the total amount of demands of the water systems and there are only m-1 independent node equations in a network of m nodes, it follows that only m-1 rather than m distinct pipes exist in the optimal network. Such a network can be proved to look like a tree, so there are no loops as actually occur in virtually all water systems.

As the result of having been able to specify the pressure head required at each node, the following equation must be satisfied:

$$\sum_{k=1}^{\text{supply node}} Q_k H_k = \sum_{i}^{\text{pipe}} q_i h_i + \sum_{j}^{\text{demand node}} Q_j H_j \qquad (2\text{-}20)$$

The right-hand sides of Eq. 2-20 and Eq. 2-7 have the same meaning, so we may write Eq. 2-7 in the form

$$C_p = a \sum_{k=1}^{\substack{\text{supply}\\ \text{node}}} Q_k H_k \tag{2-21}$$

Q_k is the amount to be supplied from the supply node K and H_k is the head that this supply would be pumped against. For multiple sources of supply, we may consider Q_k as a decision variable. Additional constraints would limit the allocation of the resources Q_k so that the amount supplied is less than or equal to the actual amount of available supply. For a network system which has a single source of supply like New York City primary distribution water supply system, the pumping cost expressed by Eq. 2-21 is constant and can be omitted from the objective function. Consequently, for this particular case, Eq. 2-18 can be written instead as

$$\text{Min } C = \sum_{i=1}^{n} L_i \beta_i X_i \tag{2-22}$$

The computer program attached in the appendix has used Eq. 2-22 instead of Eq. 2-18.

Consideration of Existing Pipe Network. It seems clear that existing as well as proposed pipes can be included in the network. As shown in Fig. 2-2, two decision variables, X_i and X_i', are assigned to each branch where there is an existing pipe; X_i' denotes the amount of the existing pipe capacity, measured in terms of pipe diameter, that is needed in the optimal network. Thus, the constraint

$$X_i' \leq d_i \tag{2-23}$$

where d_i is the existing diameter for pipe i, is added to limit the maximum size of the decision variable X_i' corresponding to this existing pipe. The unit cost of X_i' could be taken as zero or a fraction of unit costs of proposed pipes. Since the existing pipe may not be large enough, additional capacity may be needed. The size of any additional pipe is given by X_i.

Considerations of Multiple Demand Patterns

As this linear programming model was formulated, only a single demand pattern was considered, but other demand patterns, which represent various times of the day and various fire flow requirements, are equally as important. As before, each pattern specifies the demands and the operating pressures when that pattern occurs. Assuming that there are γ demand patterns, the constraints of the linear programming model would become

$$\sum_{i_\ell} K_{i_\ell} X_{i_\ell} \leq -Q_{j_\ell} \qquad \ell=1,\ldots,\gamma$$

$$i_\ell = (\ell-1)n+1, (\ell-1)n+2, \ldots, \ell n \qquad (2\text{-}24)$$

$$j_\ell = (\ell-1)m+1, (\ell-1)m+2, \ldots, \ell m$$

and

$$X_{i_\ell} \leq X_{d_i} \qquad (2\text{-}25)$$

where X_{d_i} denotes the design pipe capacity for pipe i, and X_{i_ℓ} denotes the pipe capacity required in branch i during demand pattern ℓ. Eq. 2-25 states that the pipe capacity used for each demand pattern may not exceed the design pipe capacity. This assures that the designed network will work satisfactorily under different demand situations. The objective function for multiple demand patterns is

$$\text{Min } C = \sum_{i=1}^{n} (L_i \beta_i + a_i{}') X_{d_i} + \sum_{\ell=1}^{\gamma} \underline{0}\, \underline{X}_\ell \tag{2-26}$$

where $\underline{X}_\ell$ represents a set of feasible diameters to satisfy the ℓ^{th} demand pattern and β_i and $a_i{}'$ are as defined previously. For illustration, consider a network design problem with two demand patterns and with existing pipes. The constraints are

$$\begin{array}{lllllll} \underline{A}_1\underline{X}_1 & & & + (\underline{A}_1)_{ex}\underline{X}_{ex} & \leq & -\underline{Q}_1 \\ & \underline{A}_2\underline{X}_2 & & + (\underline{A}_2)_{ex}\underline{X}_{ex} & \leq & -\underline{Q}_2 \\ \underline{X}_1 & & -\underline{X}_d & & \leq & \underline{0} \\ & \underline{X}_2 & -\underline{X}_d & & \leq & \underline{0} \\ & & & \underline{X}_{ex} & \leq & (\underline{d}_{ex})^p \end{array} \tag{2-27}$$

In vector form, the objective function is

$$\text{Min } C = \underline{0}\,\underline{X}_1 + \underline{0}\,\underline{X}_2 + \underline{C}\,\underline{X}_d + \underline{0}\,\underline{X}_{ex} \tag{2-28}$$

where

$\underline{A}_1$, $\underline{A}_2$ = constraint matrix for demand patterns 1 and 2 with dimension (m x n). n = no. of nodes, n = no. of pipes.

$\underline{X}_1$, $\underline{X}_2$ = pipe capacities used for patterns 1 and 2. [Note that they have zero cost coefficients, and that both have dimensions (n x 1).]

$\underline{X}_d$ = set of actual design pipe capacities, which have non-zero cost coefficients. Dimension (n x 1)

$(\underline{A}_1)_{ex}$, $(\underline{A}_2)_{ex}$ = subset of constraint matrix $\underline{A}_1$, $\underline{A}_2$ containing columns corresponding to the existing pipes (m x NEP), NEP = no. of existing pipes.

$\underline{X}_{ex}$ = set of decision variables corresponding to the existing pipes. (NEP x 1)

$\underline{Q}_1$, $\underline{Q}_2$ = demand vector for pattern 1 and 2. (m x 1)

$\underline{0}$ = zero vector. (n x 1)

$\underline{d}_{ex}$ = vector representing the known diameters of existing pipes. (NEP x 1)

It is useful to know the dimensions of the constraint coefficients matrix identified commonly by the symbol $\underline{A}$. For a network of m nodes, n pipes, NEP existing pipes and P demand patterns, the number of rows is

$$Pm + pn + NEP = P(m+n) + NEP \; ; \tag{2-29}$$

and the number of columns is

$$Pn + n + NEP + [P(m+n) + NEP] \tag{2-30}$$

The quantity in brackets is associated with the slack variables. $\underline{A}$ contains mostly zero elements, but its dimensions could become too large, even for large-scale computers, for moderate system designs with just a few demand patterns. Therefore, to reduce the size of $\underline{A}$ matrix by partitioning is desirable and may be possible. Future research is required to find the best way to decompose large LP multiple demand pattern distribution system models.

Computational experience is, thus far, limited to single demand patterns. The formulation proposed for multiple demand patterns requires further programming and investigations. For a single demand pattern, the optimal design is a tree-shaped network without a loop. However, for multiple demand patterns, loops may optimally occur as actually found in practice.

Some Features of the LP Formulation

This LP problem appears to have advantages over the original non-linear problem in that the theory of linear programming has been well developed and is computationally very efficient. It eliminates the need to analyze numerous solutions in search of the optimum. In addition, the economic interpretation of the dual solution has latent value for improving existing design methodology. For example, some of the binding constraints will represent fire flow requirements, and the dual solution will indicate reduction in system cost that would attach to a unit reduction in the fire flow requirement. Non-linear programming models usually do not provide such convenient and straightforward sensitivity analysis.

Computer Programs

A computer program has been developed and tested for the case of single demand pattern with existing pipe network. The listings of the program can be found in the appendix. The descriptions of the programs, their use and data formats for input and output information are given below. The flow chart is shown in Fig. 2-3.

There are five subprograms in the LP pipe network optimizer, namely MAIN program and subroutines NCOST, ORGLP, LPROG and SIMPLX. Their functions can be briefly described as follows:

(i) MAIN

It reads in all necessary input data for the computation. The order of input is:

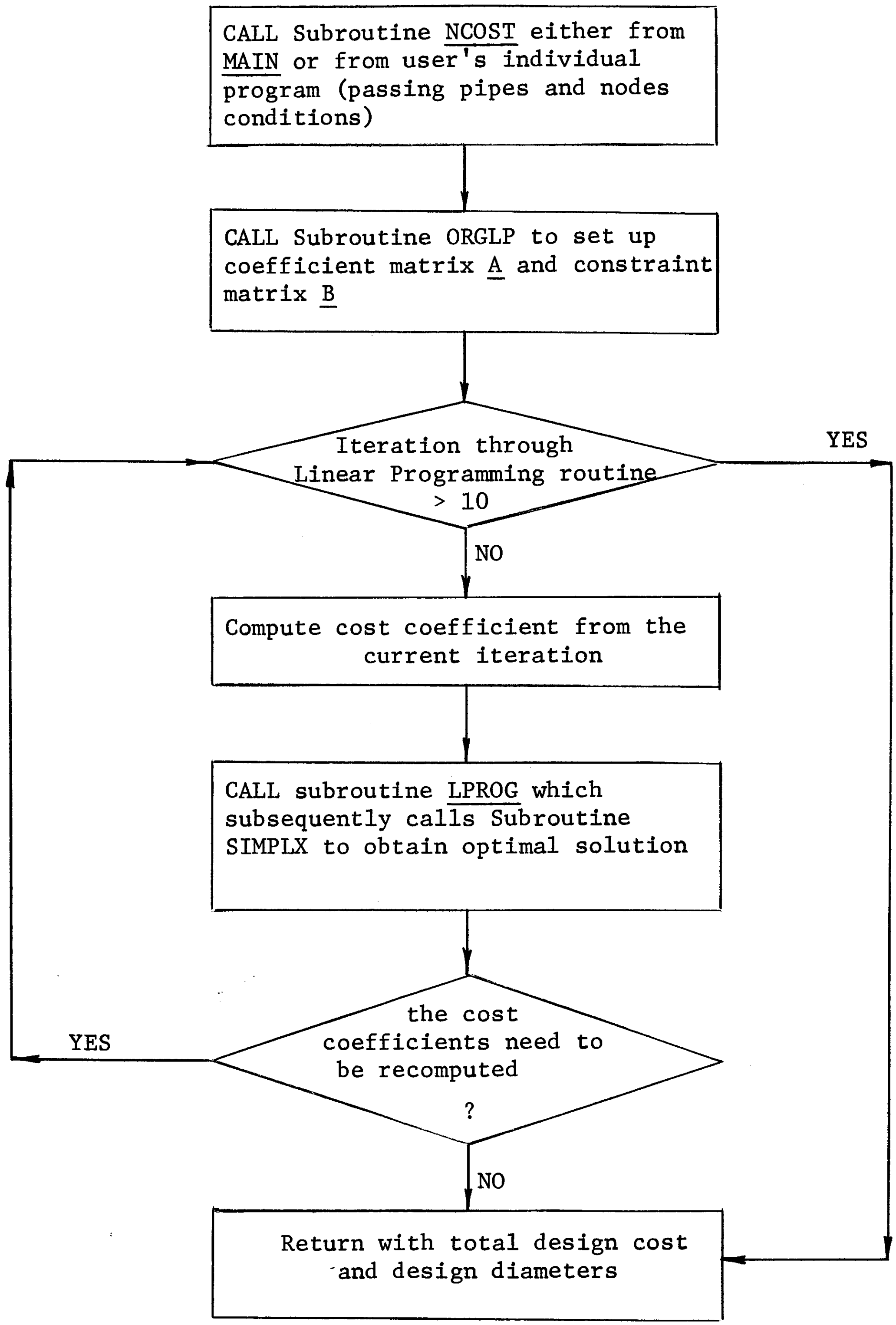


FIGURE 2-3: Flow Chart of LP Optimizer

(a) XNP, XNN, HWC, FACTOR (4F 10.1)

XNP = number of pipes (or tunnels)

XNN = no. of nodes

HWC = Hazen-Williams Coefficient

FACTOR = a scaling factor to scale the constraint coefficients A and requirement matrix B so that they have approximately the same orders of magnitude.

(b) XIP (I), XJN (I), XKN (I), FL (I), EXD (I), ESDIA (I), (6F 10.1)

I = pipe index

XIP (I) = identification number for pipe I

XJN (I) = upstream node for pipe I

XKN (I) = downstream node for pipe I

FL (I) = length of pipe I in feet

EXD (I) = existing I^{th} pipe diameter in inches

ESDIA (I) = identifier for estimated design diameter of pipe I (piecewise linearization of cost function)

Let D = estimated pipe diameter in inches

then

ESDIA (I)	= 1	if $0 \le D \le 60$
	= 2	if $60 < D \le 120$
	= 3	if $120 < D \le 180$
	= 4	if $180 < D \le 240$
	= 5	if $240 < D \le 300$
	= 6	if $300 < D$

Note that the identifier ESDIA (I) given above is especially designed for tunnel design for the New York City water supply tunnel system. As a result of that, the tunnel cost function, Eq. 2-5, is used for obtaining the total tunnel system cost.

(c) XIN(I), Q(I), H(I) (3F 10.1)

I = node index

XIN(I) = identification number of node I

Q(I) = demand or supply at node I if a demand node, use positive sign, otherwise, negative

H(I) = energy head at node I

Note: All variables in input data are real numbers for convenience in setting up data cards. If subroutine NCOST is called directly without going through the main program, all variables except HWC, FL(I), EXD(I), H(I), should be integers.

MAIN program also writes out total system cost, the design diameters, and the portion of existing pipe diameters used for that particular design.

(ii) Subroutine NCOST

This subroutine serves as a monitor program for the linear programming optimizer. It calls subroutine ORGLP to set up proper A and B matrices and then calls subroutine LPROG which subsequently calls subroutine SIMPLX to solve the LP problem. Eventually, it returns the desired design information to the MAIN program. The calling sequence is:

```
CALL NCOST (NN, NP, IN, IP, JN, KN, FL, EXD, Q, H, HWC,
            TCOST, DIANEW, DIAUSE, KESDIA, OBJ, FLOW, FACTOR)
```

in which NN, NP, IN, IP, JN, KN, FL, EXD, Q, H, HWC, KESDIA are input from the MAIN program. NN is the integer equivalent of XNN in MAIN program, and

DIANEW = New design diameters

DIAUSE = the portion of existing pipe diameter used.

OBJ = value of the optimal objective function from linear programming routine. It is not equal to the total system cost because the cost coefficients in LP routine are not the actual cost coefficients. Actual unit cost formula (2-4) or (2-5) should be used to compute the total system cost after the design diameter is determined.

(iii) Subroutine ORGLP

The function of this subroutine is essentially to set up an augmented constraint matrix A and an augmented requirement vector B. Here the term "augmented" is used because the first row of the A matrix contains the coefficients in the objective function.

(iv) Subroutines LPROG and SIMPLX

Subroutine LPROG together with subroutine SIMPLX will solve a linear programming problem of the form:

Minimize the objective function $\underline{C}\ \underline{X}$

Subject to the constraints $\underline{A}\ \underline{X} = \underline{B}$

$$\underline{X} \geq 0$$

where $\underline{C}$ and $\underline{B}$ are given 1 x n and m x 1 matrices respectively, $\underline{A}$ is a given m x n matrix and $\underline{X}$ is a variable of n x 1 matrix

The calling sequence of subroutine LPROG is

```
CALL LPROG (ME, M , N, A, B, Z, DIA, OBJ)
```

in which

ME = is the row dimension in the calling program of the augmented matrix of coefficient, A.

M = the number of constraint equations plus 1 i.e. P (m+n) + NEP + 1 (eq. 2-29)

N = number of variables (eq. 2-30)

A = augmented matrix of constraints coefficients

B = augmented matrix of requirements

Z = variable matrix containing the solution to the linear programming problem after execution of the subroutine

DIA = variable matrix containing the solution to the primal problem

OBJ = value of the objective function

Examples

The network used in the example is the New York City primary water distribution tunnel system which is shown in Figure 2-4. Input data are shown in Figure 2-5. Output results are respectively shown and partially tabulated in Figures 2-6 and 2-7.

The computation was done on the M.I.T. Urban Systems Laboratory IBM System 360/37 time sharing system. The table at Figure 2-7 indicates how much of the existing capacity has been used. If the capacity of existing tunnels is not adequate, the size of a new additional tunnel is indicated. For example the existing 180 inch capacity of pipe 1 was needed, as well as a new addition of 52 inch diameter. For pipe 9, no new pipe is needed since only the capacity of 106 inches out of the existing capacity of 180 inches is actually required.

The computation seems to be very efficient. It takes about 5 sec of C.P.U. time to solve a problem with a constraint matrix dimensioned (40 x 60). The size of matrix which can be handled with the existing program is estimated to be about (100 x 100). In other words, it can handle approximately two demand patterns for the example given.

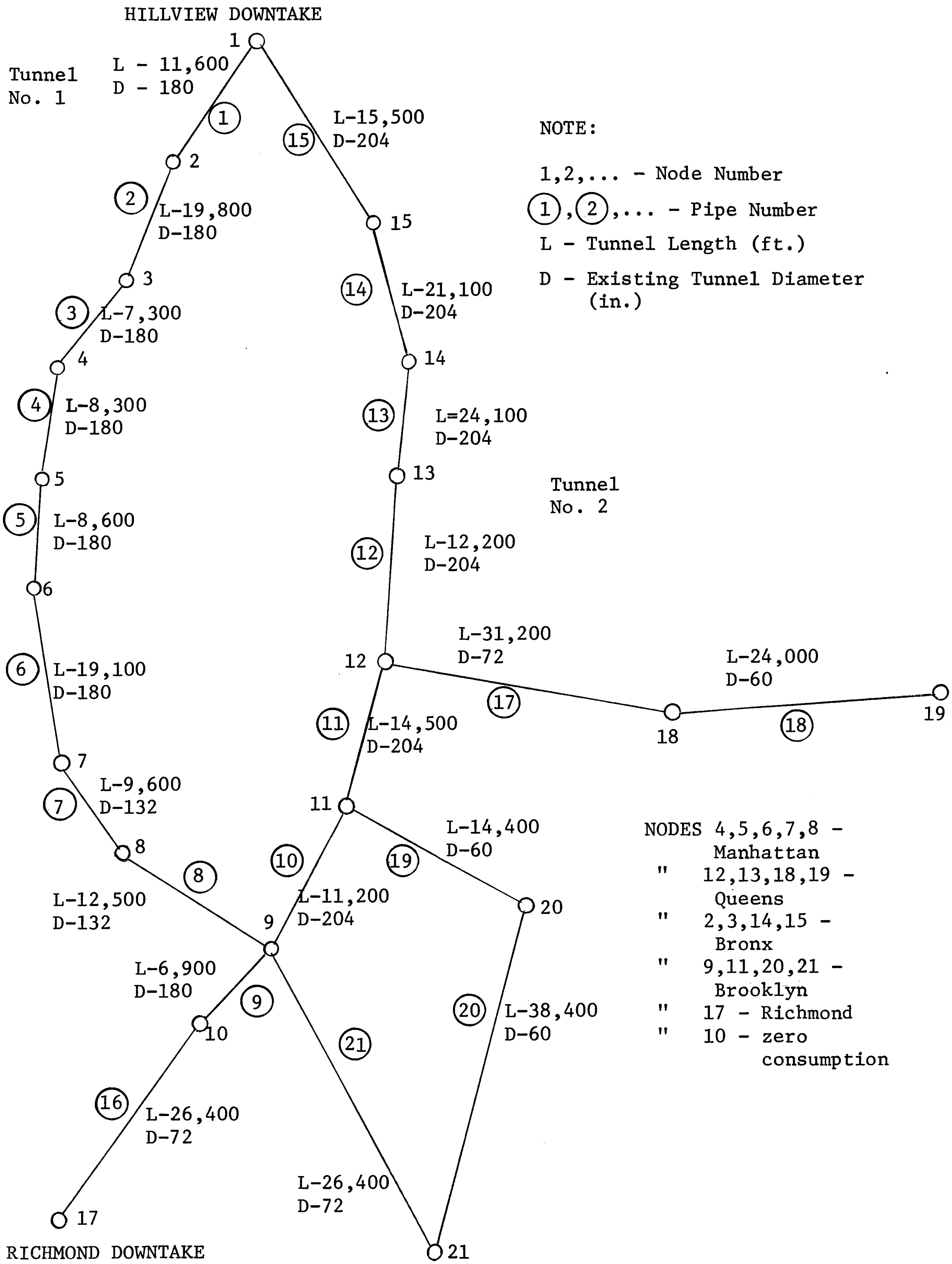


Figure 2 -4: Existing System

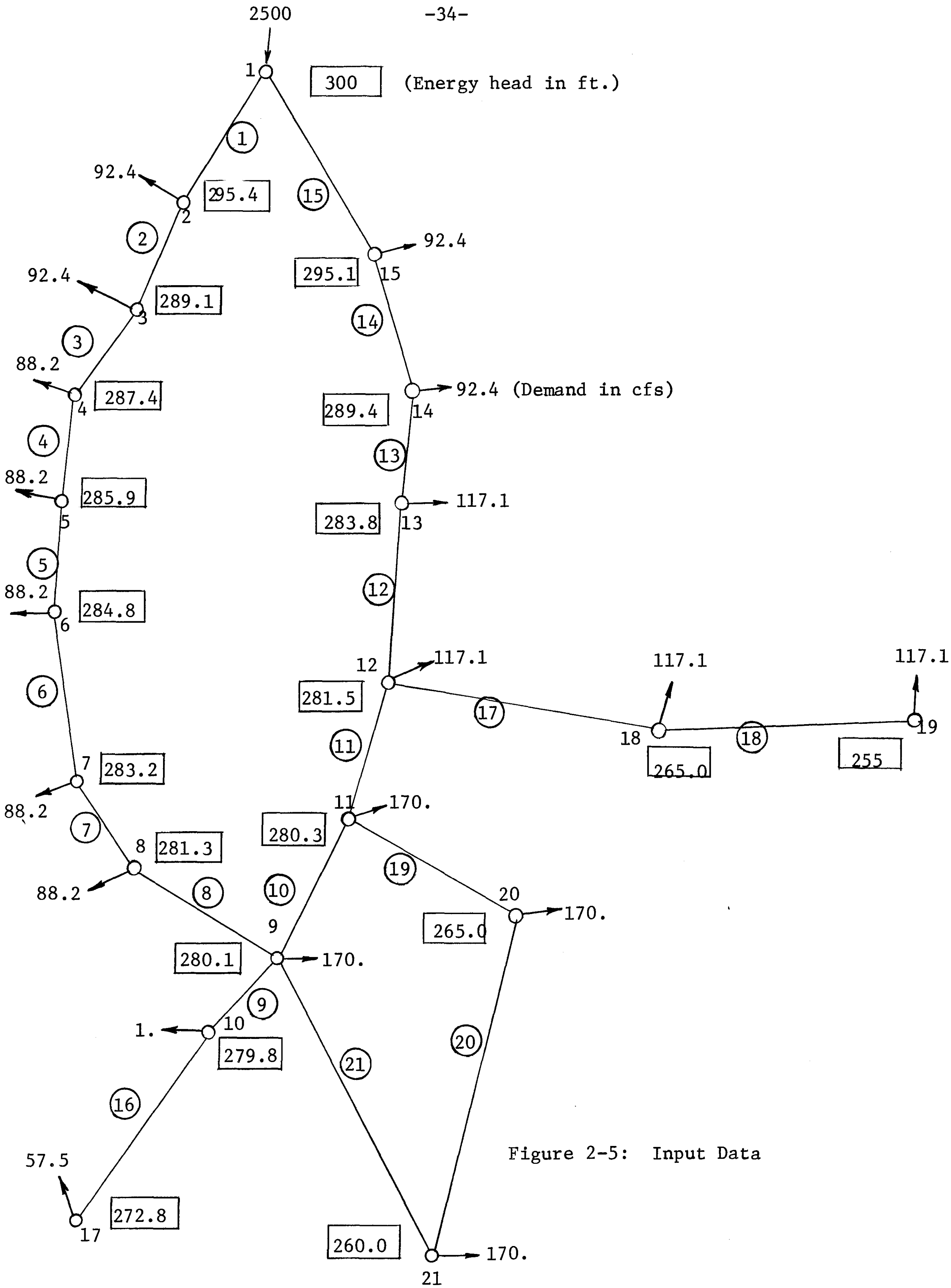


Figure 2-5: Input Data

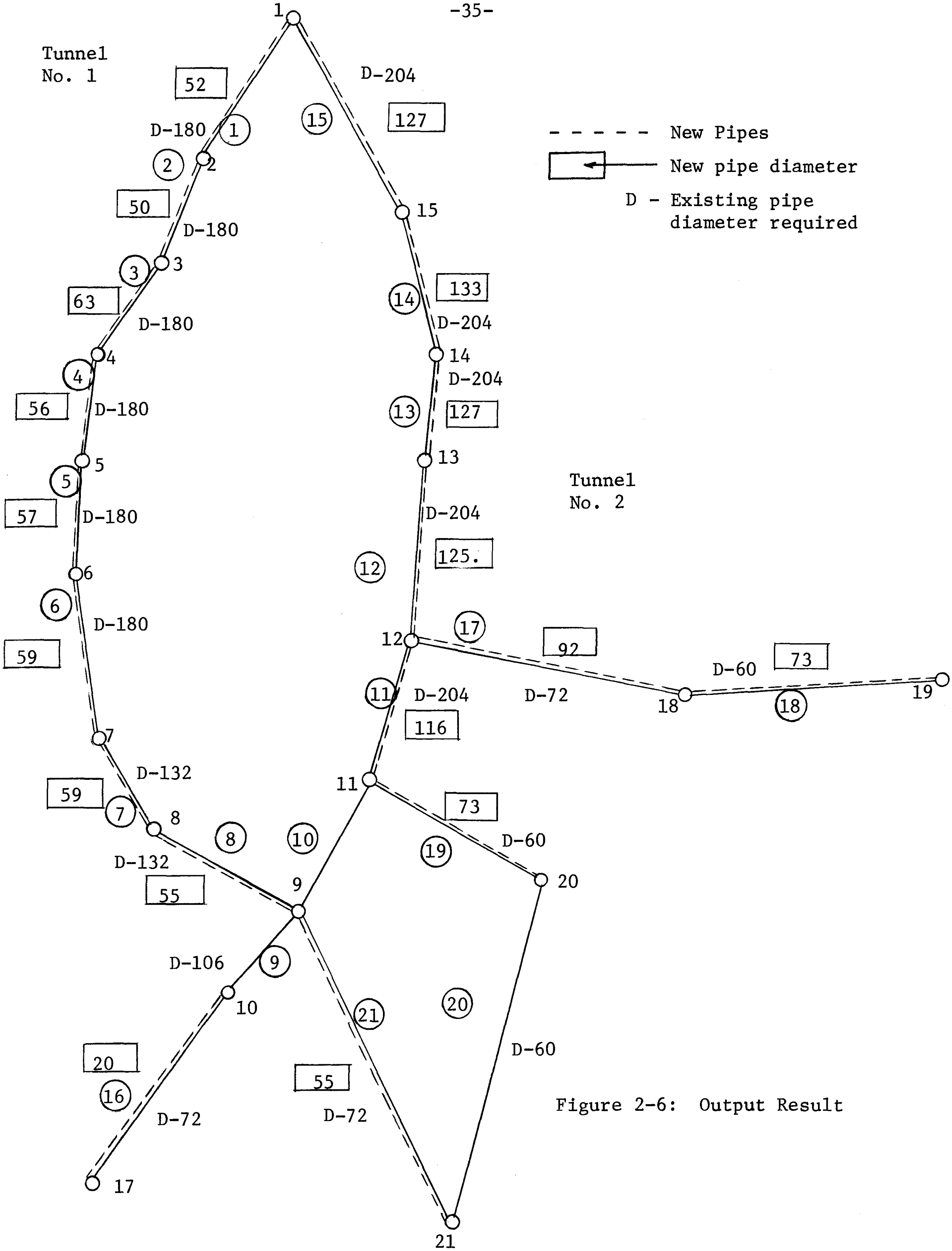


Figure 2-6: Output Result

TOTAL COST = 78084928.00 DOLLARS

PIPE	NEW DIAMETER	EXISTING DIAMETER	USED DIAMETER
1	52.02	180.00	180.00
2	49.90	180.00	180.00
3	63.41	180.00	180.00
4	55.59	180.00	180.00
5	57.25	180.00	180.00
6	59.19	180.00	180.00
7	59.06	132.00	132.00
8	54.95	132.00	132.00
9	0.0	180.00	106.33
10	0.0	204.00	204.00
11	116.21	204.00	204.00
12	125.25	204.00	204.00
13	126.87	204.00	204.00
14	133.07	204.00	204.00
15	126.52	204.00	204.00
16	19.52	72.00	72.00
17	91.83	72.00	72.00
18	72.76	60.00	60.00
19	72.61	60.00	60.00
20	0.0	60.00	60.00
21	54.82	72.00	72.00

Figure 2-7: Output Example

Chapter 3

APPLICATION OF DYNAMIC PROGRAMMING TO CAPACITY EXPANSION

Introduction

In the preceding chapter, a method is presented for water distribution system design. This method considers the flow and pressure conditions that may be typical for some particular period of time, but this directly addresses the fact that system demands tend to increase with time in response to population and economic growth. In other words, the previous method takes a representative snapshot of the system over a certain period of time. For those conditions a system may be designed which would behave according to the design criteria. Since the demand may be growing with time, there arises the problem of how to make investments over a period of time. This is called a "capacity expansion problem." The solution should indicate when to build extra capacity, how much to build and where to build. The investment problem is as complicated and as complex as the water distribution system analysis problem. The optimal time phasing of resource allocation is a central problem of design. It is the purpose of this chapter to define the capacity expansion problem and to show that the method of dynamic programming may be applied to its solution.

Dynamic Programming Formulation

Dynamic programming is an important technique in non-linear constrainted optimization problems. It can be applied to capacity expansion problems in more than one way so there are possible other formulations than the one presented here.

A basic assumption of economic analysis of engineering projects is there exists an economic time horizon, T, beyond which there is no value to future economic activity. This should not be confused with the useful life of a particular component, such as a pipe or a pump, which may be shorter than the length of the economic time horizon. The actual value used for T is immaterial to the problem formulation. On the other hand, T is assumed to exist and it takes on a finite value, however large.

The total period, T, may be partitioned into a number of sub-periods, say N of them, possibly of unequal length. These sub-periods, of length t_i, may be called design periods; and it is the period of time for which the capacity expansion, made at the beginning of the period, will be adequate. The first design period may not actually begin until existing capacity is exhausted. Then, additional capacity is required; and it, together with the existing capacity, should be adequate for the next t_i years. At that time, additional capacity again will be required.

The cost of an additional unit of capacity is assumed to remain unchanged over the economic time horizon so the economic analysis is done on a "constant dollar" basis. However, a dollar of cost incurred at different points in time are not economically equivalent so that some adjustment must be made to compare alternative expansion plans where costs are incurred at various times over the economic time horizon. The proper adjustment is to correct future costs to present costs. So-called present costs are measured in dollars, and the sum of the present costs of each expansion capacity gives the total present cost of the entire project. The present cost of a future expenditure is the amount of money that could be invested now at interest rate i to yield an amount equal to the expenditure at that future time. This may be interpreted as a function of the interest rate, so the value used for i is of special concern.

The appropriate value for i for public investment projects should represent the social time preference for money. It is a measure of how much a dollar must yield, in addition to its own value, over the period of one year for any typical year. It is not likely to be equal to the market interest rate at which money can be borrowed because that interest rate includes a "hedge" against inflation which is needed to assure that the initial dollar invested will return its own value. In other words, the market interest rate may be assumed to represent the sum of the social cost of capital plus an allowance for anticipated monetary depreciation (Hirshleifer, et al., 1963).[13] Presumably the value of i, interpreted as the social cost of capital only, should not vary as price levels change. It may be argued that this, the <u>real</u> marginal productivity of capital, is essentially independent of price levels.

Returning to the capacity expansion problem, the best sequence of expansion is that one which gives the minimum total present cost. The Dynamic Programming objective, then, is to find the minimum cost sequence.

Let the optimum number of design periods in the economic time horizon be n. Also, let t_j be the length of the j^{th} design period. One constraint is that

$$t_1 + t_2 + \ldots + t_n = T \qquad (3\text{-}1)$$

Let the present cost of the expansion made during design period j be $r_j(t_j)$. This is a function of the length of the period, t_j, because larger expansions are usually needed to satisfy longer periods. This also is a function of j because the cost must be discounted to present value, and discounting depends on the time of investment which depends, indirectly, on the period. In other words, if $c(t_j)$ is the actual cost of an expansion which is adequate, together with the existing capacity, for the next t_j years; the present cost, $r_j(t_j)$ is

$$r_j(t_j) = \frac{c(t_j)}{(1 + i)^{t_1 + t_2 + \ldots + t_{j-1}}} \tag{3-2}$$

The total present cost is

$$F_n = r_1(t_1) + r_2(t_2) + \ldots + r_n(t_n) \tag{3-3}$$

so the objective is to minimize $F_n(T)$.

In terms of Dynamic Programming, the times at which decisions must be made are known as stages and the decision moves the process from one so-called state to another. Schematically, this is illustrated in Figure 3-1.

In terms of the capacity expansion problem, the stage is associated with the design period (i.e. j denotes the stage). More specifically, j denotes the number of design periods remaining until the end of the economic time horizon.

The decision to be made at stage j is the value of t_j. This moves the decision process from state s_{j-1} to state s_j. The state of the system is the time between the present time (i.e. t = 0) and the future time

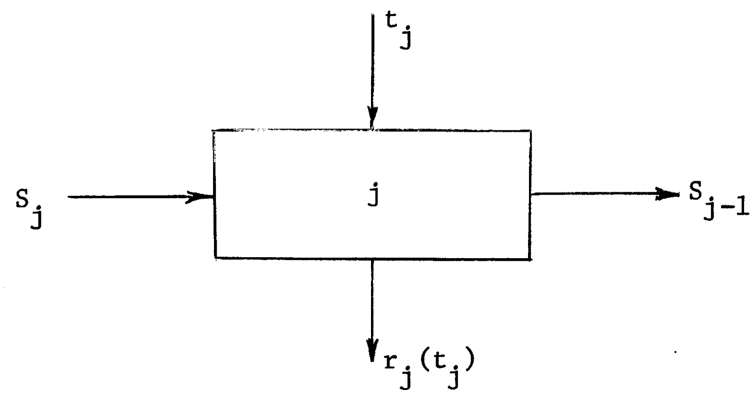


FIGURE 3-1: Typical Dynamic Programming Stage

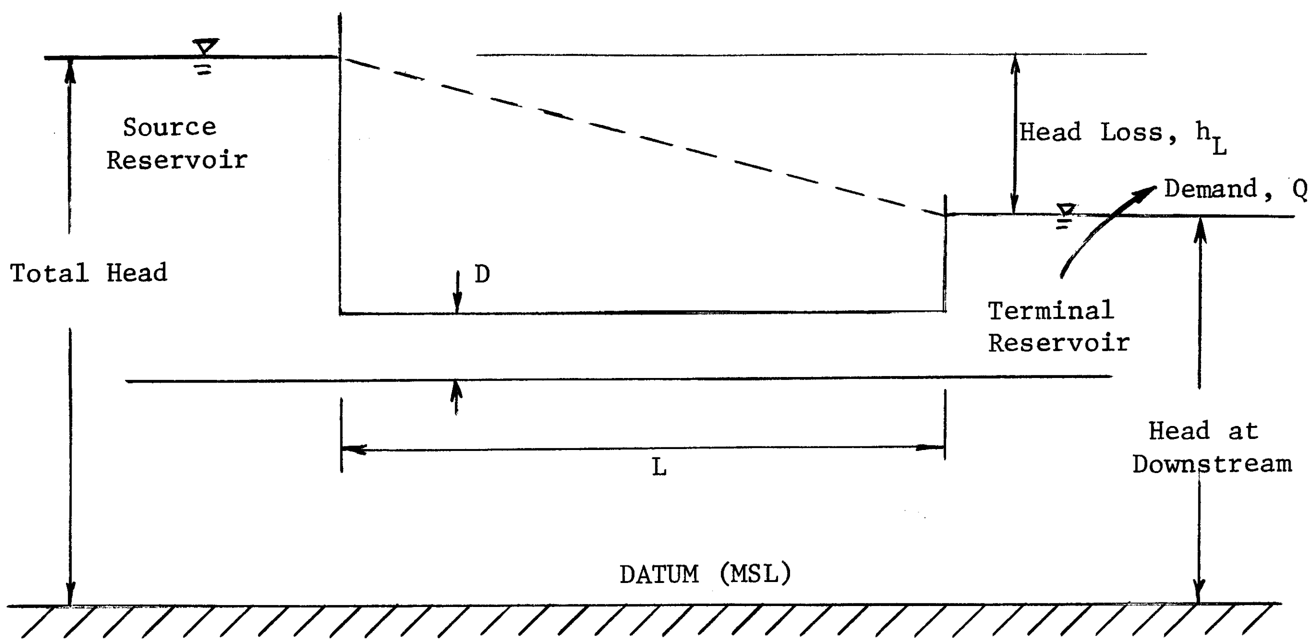


FIGURE 3-2: Simple System Arrangement

at which the expansion associated with t_j must be made. This decision process moves backward in time from the economic time horizon. It follows that

$$S_0 = T \quad , \tag{3-4}$$

that

$$S_1 = T - t_1 \tag{3-5}$$

and that, in general,

$$S_j = S_{j-1} - t_j \tag{3-6}$$

If the state of the system at the beginning of stage j is S_j, the maximum possible value of t_j is S_{j-1}.

The optional set of decisions $[t_1, t_2, \ldots ,t_n]$ is determined through an ordered search of the alternatives. This search procedure is based on Bellman's Optimality Principle which states that no matter what decisions have been made, in time, up to the present, the optional decision depends only on the immediate return from the present decision and on the present value of subsequent returns if subsequent decisions are made optionally thereafter.

To apply this principle, let the optimal value function be denoted by $f_{j-1}(S_{j-1})$. This gives the present value of returns subsequent to stage j if the process leaves stage j in state j-1 and if all decisions are made optimally in stages 1 to j-1.

The total return to be expected from decision t_j at stage j, with optimal activity thereafter is $r_j(t_j) + f_{j-1}(S_{j-1})$. This depends on the state upon entering stage j, since S_{j-1} is equal to $S_j + t_j$. The optional value function for stage j, according to the Optimality Principle is

$$f_j(S_j) = \min_{t_j} [r_j(t_j) + f_{j-1}(S_j + t_j)] \tag{3-7}$$

At each stage, a value of t_j is determined. After n stages, if n is the optimal number of design periods, the sum of all t_j will be equal to T. Accordingly, the value of $f_n(S_n)$ will be the minimum total present cost. Also, S_n must be zero since this denotes the present time if n is the optimal number of stages.

It is not until the optimization is complete that the optimum number of stages is known. Therefore, some procedure is needed to test if the current stage is the last stage. This is accomplished by comparing, at stage j + 1, the quantities $f_{j+1}(0)$ and $f_j(0)$. If $f_j(0)$ is not larger than $f_{j+1}(0)$, then it follows that n = j. If $f_j(0)$ is larger than $f_{j+1}(0)$, adding another stage decreases the total present cost so at least that additional stage is required. The quantity $f_0(0)$ may initially be set equal to some large number because S_0 must be equal to T, not zero.

Application to a Simple Water System

It appears possible to apply this approach to the problem of capacity expansion for the water distribution system. In developing such a dynamic programming model it appears rational, as a first step, to consider the simple system arrangement of a reservoir connected to a pipe discharging in response to the demand as shown in Fig. 3-2. The discussion of this simple system will be given in the next section. Considering this simple system arrangement will allow the following verification of the method before undertaking the entire network system:

(1) Dynamic programming does provide the optimum.

(2) The optimum occurs in a finite number of stages.

(3) The method is economical in terms of computer time.

These are not obviously satisfied as they would be in a linear programming problem since there is no packaged program available and a dynamic programming problem is solved by a tailor-made program. Development of such a program for the simple system case should easily be adapted to the total network case. Only the method for computing present costs at each stage should be different.

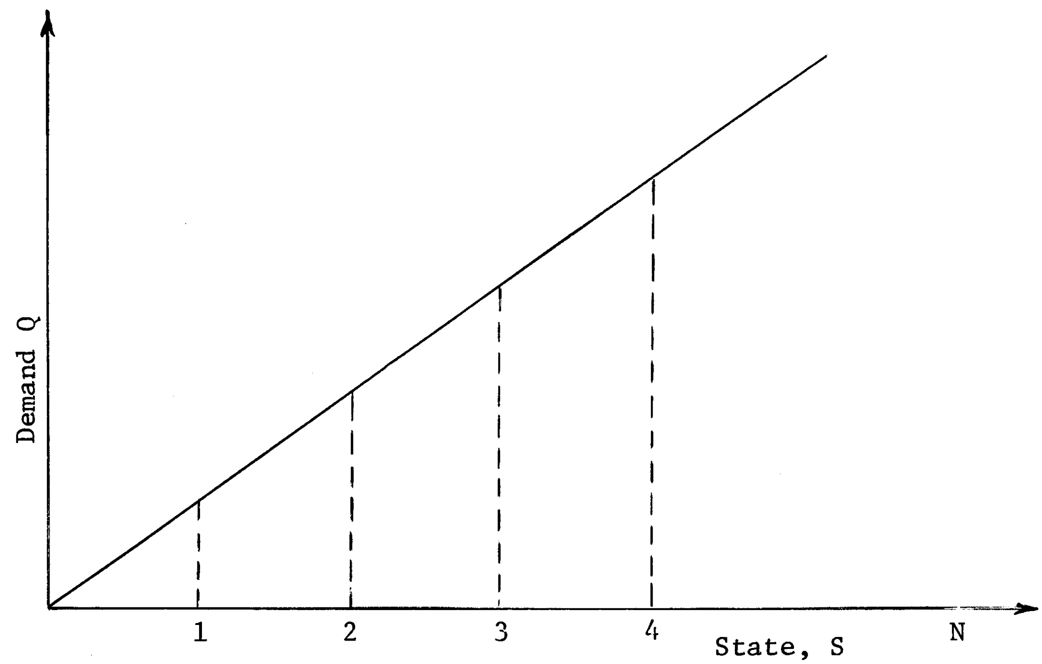
Common to both applications is the model representation of Fig. 3-3. The Tableau of Fig. 3-4 indicates how the method proceeds.

The simplified network considered is shown in Fig. 3-2. The simplified system consists of a source reservoir of infinity capacity, a pipe of length L and an outlet responding to an increasing demand. Assume that the maximum allowable pressure loss along the pipe is given. As the demand increases, the operating head loss along the pipe will increase. An additional pipe is required when the head loss reaches the allowable maximum value. Installation of larger pipe will have larger replacement intervals but will require larger investments.

The data and formula used for computation are summarized as follows:

The length of the pipe = 100,000 ft.

The economic time horizon = 35 years. (begins in year 1975 and ends in year 2010)



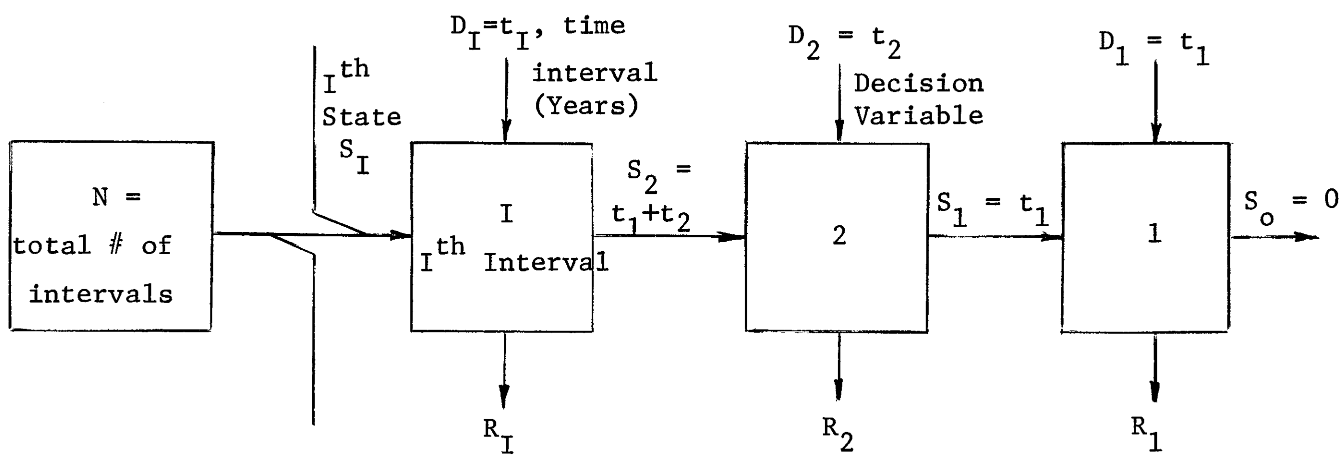


FIGURE 3-3: Dynamic Programming Model

Stage I	State Variable S_I	Decision Variable D_I	State Variable S_{I-1}	Return Function $R_I = R_I(S_I, D_I)$	$Q_I = f_{I-1} + R_I$	$f_I(S_I) =$ Min Q_I	$D_I = D_I$ (Min Q_I)
1	0	0	0	r_{10}	r_{10}	r_{10}	0
	1	1	0	r_{11}	r_{11}	r_{11}	1
	2	2	0	r_{12}	r_{12}	r_{12}	2
	⋮		0				
	N	N	0	r_{1N}	r_{1N}	r_{1N}	N
2	1	0	1	r_{20}	$r_{20} + r_{11}$	}	
		1	0	r_{21}	$r_{21} + r_{10}$		
	2	0	2	r_{20}	$r_{20} + r_{12}$	}	
		1	1	r_{21}	$r_{21} + r_{11}$		
		2	0	r_{22}	$r_{22} + r_{10}$		
	⋮						
	N						
⋮							
N	1						
	2						
	⋮						
	N						

FIGURE 3-4: Form of the Dynamic Programming Model Tableau

The demand grows geometrically according to

$Q(t) = Q_0(1+g)^t$ where Q_0 = Initial demand

g = growth rate

The required diameter, D, expressed in terms of demand and head loss, is

$$D = \frac{1.38\ Q^{0.38}}{C_{HW}^{\ 0.38}\ (HL/L)^{0.2}}$$

in which the units are D[ft.] and Q[cfs.].

The capital cost of pipe per linear feet has the form

$$C = \alpha D^{1.25}$$

in which the units are C[$/ft.] and where

$\alpha = 43.5$ from year 1975-1982

$\alpha = 30.5$ after year 1982 (This indicates there might be a breakthrough in construction technology in year 1982 to reduce the cost.)

After discounting, the present value of the unit capital cost would read

$$C = \frac{\alpha D^{1.25}}{(1+r)^t}$$

The results of the tests for various key variables are tabulated in Table 3-1. The design periods are restricted to be multiples of 5 years.

One of the most interesting results shows the effects of future cost changes on the optimal cycle time. The relationship illustrated in

Discount Rate %	Growth Rate %	Cost Change in 1985-%	Optimal First Years	Design Second Years	Periods Third Years	Total Present Cost $/lin.Ft.
10%	.5%	-30%	10.	15.	10.	720.
5.	1.0	-30.	10.	25.	0.	1247.
10	1.0	-30.	10.	15.	10.	1031.
2.5	1.0	0.	35.	0.	0	1423.
5.	1.0	0.	20.	15.	0.	1392.
10.	1.0	0.	15.	20.	0.	1157.
5.	1.0	+20.	35.	0.	0.	1423.
10.	1.0	+20.	15.	10.	10.	1198.
5.	3.0	-30.	10.	25.	0.	2420.

Table 3-1: Optimal Time Staging of Construction, for Simplified System Shown in Figure 3-2, (Dynamic Programming Solution)

Fig. 3-5 was derived according to a method given by Manne[14] which assumes that cost remains constant in time. It indicates that as the interest rate increases, it is optimal to defer construction and build a sequence of smaller projects. It is seen to be very sensitive to the discount rate and to be relatively insensitive to the growth rate. However, due to possible breakthroughs in construction technology, the assumption that cost remains constant with time is questionable.

Of primary importance is the length of the first period, t_1, since that is what must be presently designed for. The value of t_1 is plotted with respect to the discount rate in Fig. 3-6 (from data in Table 3-1). The relationship is similar to that in Fig. 3-5 which was derived for constant cost over time. These results are very sensitive to the time in which the cost change is expected to occur. For this analysis, the cost was expected to change in 15 years from the base year of 1970. Fig. 3-6 reflects the effects of such a change if it could be forecast. Nevertheless, the optimal design staging now appears equally dependent on any cost changes as well as the discount rate.

These considerations indicate that over an economic time horizon of 30 to 40 years, the optimal expansion will consist of 2 to 3 separate projects. Since the number of states per stage and hence the computational effort in the dynamic program is T/DT, where DT is the time interval considered, it seems adequate to look at a time interval of 10 years.

A typical dynamic programming tableau is presented in Table 3-2. The optimality condition of negligible improvement from one stage to the

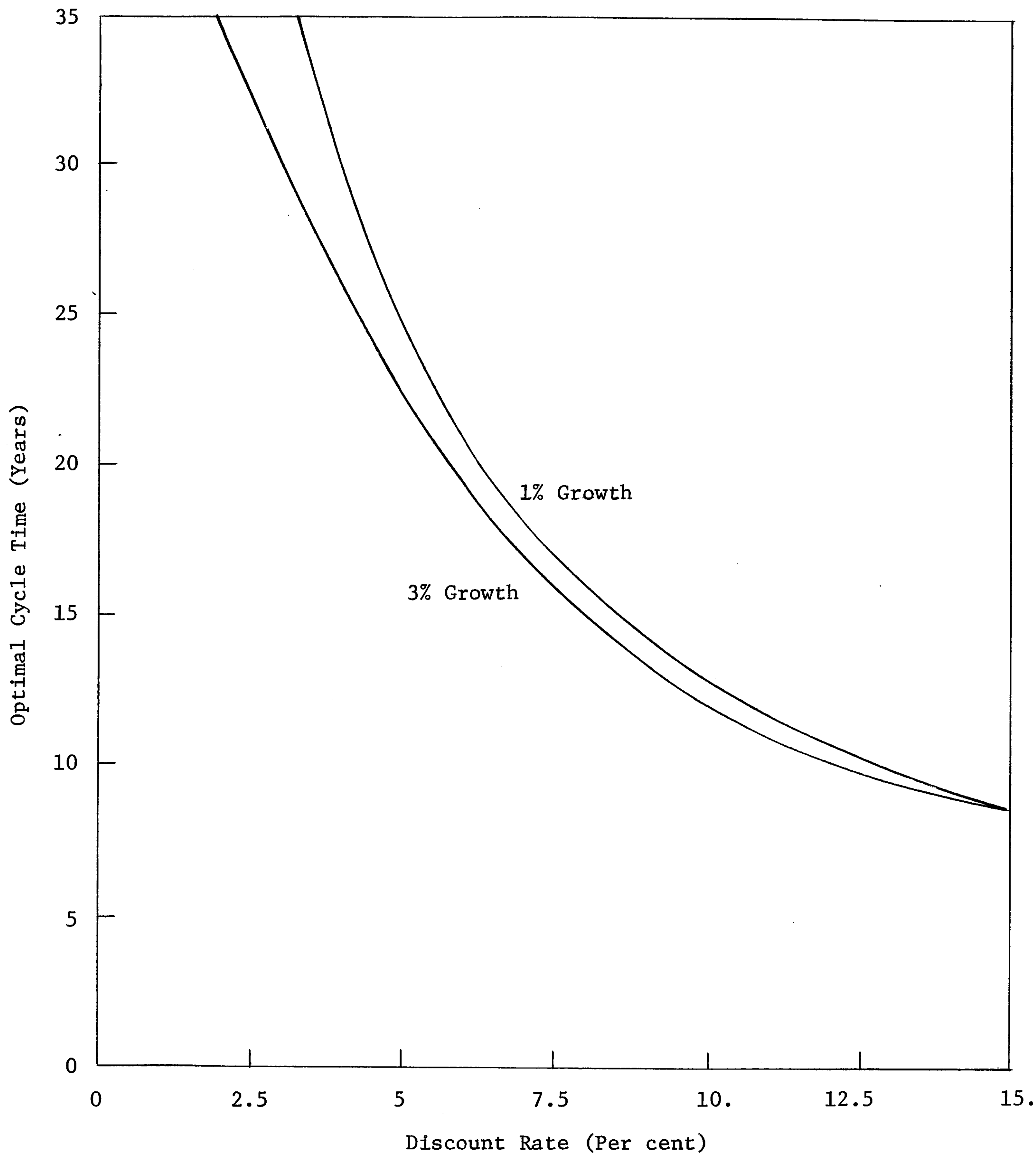


FIGURE 3-5: Manne Capacity Expansion Model

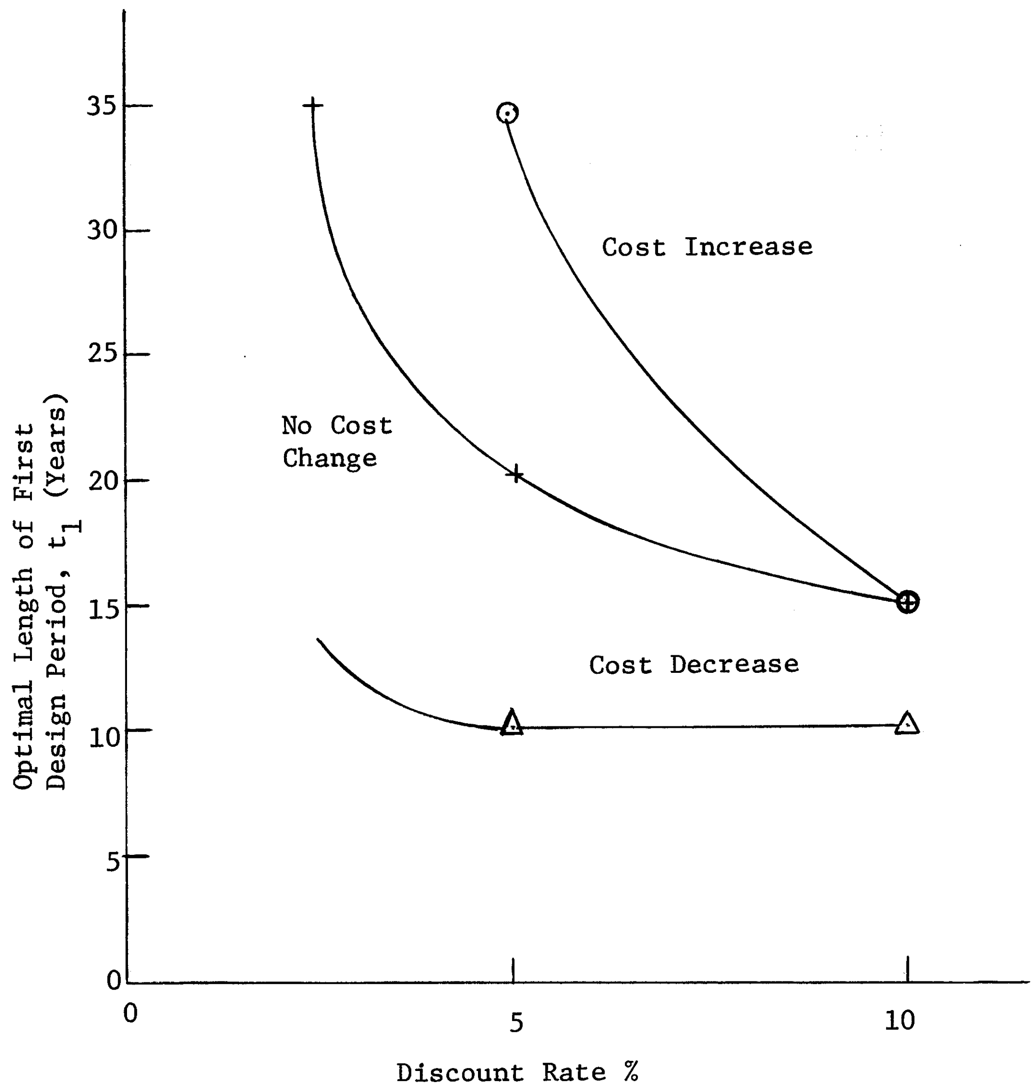


FIGURE 3-6: Optimal Length of First Design Period VS. Discount Rate (Dynamic Programming Solution)

TABLE 3-2

STAGE 1

STATE S_I years	DECISION D_I	STATE IN LAST STAGE S_{I-1}	RETURN NPC R_I	$Q_I = f_{I-1}+R_I$	$f_I(S_I) = \min Q_I$
0	0	0	0	0	0
5	5	0	525	525	525
10	10	0	738	738	738
15	15	0	906	906	906
20	20	0	1051	1051	1051
25	25	0	1183	1183	1183
30	30	0	1306	1306	1306
35	35	0	1423	1423	1423

Table 3-2: Dynamic Programming Tableau

- Simplified System -

Example: T = 35 years (1975-2010)

T = 5 years

R = 5% Discount Rate

G = 1% Growth Rate

H_i = 25' Allowable Head Loss

C = -30% Cost Reduction in 1985

Table 3-2 (continued)

STAGE 2

S_I	D_I	S_{I-1}	R_I	Q_I	$f_I(S_I)$	
5	5	0	525	525	525	
10	5	5	411	437		
	10	0	738	738	738	OPT'L
15	5	10	226	964		
	10	5	578	989		
	15	0	906	906	906	
20	5	15	177	1083		
	10	10	318	1056		
	15	5	710	1235		
	20	0	1051	1051	1051	
25	5	20	137	1188		
	10	15	249	1155		
	15	10	390	1128		
	20	5	823	1348		
	25	0	1183	1183	1128	
30	5	25	109	1237		
	10	20	195	1246		
	15	15	305	1211		
	20	10	452	1190		
	25	5	927	1452		
	30	0	1306	1306	1190	
35	5	30	85	1275		
	10	25	153	1281		
	15	20	239	2190		
	20	15	354	1260		
	25	10	509	1247		
	30	5	1023	1548		
	35	0	1423	1423	1247	

Cost After Stage 2 = 1247 .

Improvement = 176 → Continue

Table 3-2 (continued)

STAGE 3

S_I	D_I	S_{I-1}	R_I	Q_I	$f_I(S_I)$
5	5	0	525	525	525
10	5	5	411	937	
	10	0	738	738	738
15	5	10	226	964	
	10	5	578	989	
	15	0	906	906	906
20	5	15	177	1083	
	10	10	318	1056	
	15	5	710	1235	
	20	0	1051	1051	1051
25	5	20	139	1188	
	10	15	249	1155	
	15	10	390	1128	
	20	5	823	1348	
	25	0	1183	1183	1128
30	5	25	109	1237	
	10	20	195	1246	
	15	15	305	1211	
	20	10	452	1190	
	25	5	927	1452	
	30	0	1306	1306	1190
35	5	30	85	1275	
	10	25	153	1281	
	15	20	239	1290	
	20	15	354	1260	
	25	10	509	1247	= OPT'L
	30	5	1023	1548	
	35	0	1423	1423	1247

Cost After Stage 3 = 1247

Improvement = 0.0 (Terminate)

Optimal Solution: T_1 = 10 years
T_2 = 25 years

Net Present Cost: $1247.

next occurs after 3 stages. The optimal expansion scheme is obtained by reading the output back as shown. Note that for any state in stages beyond stage 1, the decision corresponding either to no construction in the current stage or to all construction in a previous stage (not shown in Table 3-2) is redundant since these situations have been evaluated in a previous stage. This would reduce substantially the number of enumerations that must be considered. The computation time (on an IBM System 1360 Model 67 time sharing computer) to do any row in Table 3-1 is in the order of 1 second. Thus, the dynamic program is computationally efficient.

In this chapter, the applicability of dynamic programming to the simplified system shown in Fig. 3-2 has been established. The effects of discount rate, demand growth rate and possible cost change to the optimal staging of water systems have also been studied. Application of dynamic programming to a network system will be discussed in the next chapter.

Chapter 4

A JOINT LINEAR PROGRAMMING-DYNAMIC PROGRAMMING MODEL FOR DISTRIBUTION SYSTEM DESIGN

Introduction

In the dynamic programming model, a cost function must be defined. For the water network design problem, this cost function is defined as the present value of the capital cost of satisfying demand for an allocated time interval t_j years in the future. For a given economic time horizon of T years, the dynamic program would indicate the optimal choice of design periods which gives the overall minimum present cost. If a distribution system is to be expanded to satisfy demands for the next t_j years, this should be done optimally so that, in fact, all costs are minimum.

At each stage of the Dynamic Programming model the cost function involves a Linear Programming network design to determine the minimum network cost for the additional capacity required until the end of the design period. Thus, there are two levels of optimization. The inner level gives the minimum cost design to satisfy the demand for the allocated t_j years in the future and the outer level gives the optimal staging over the economic time horizon.

Implicit in this procedure is an assumption that the existing configuration of the network does not depend on the expansion path up to that time. The validity of this assumption has not been tested. The flow chart given in Fig. 4-1 indicates the relationship between the LP model and the DP model. The example to be given should help make clear the application of Dynamic Programming to the total network system. The justification of this application is given in the next section.

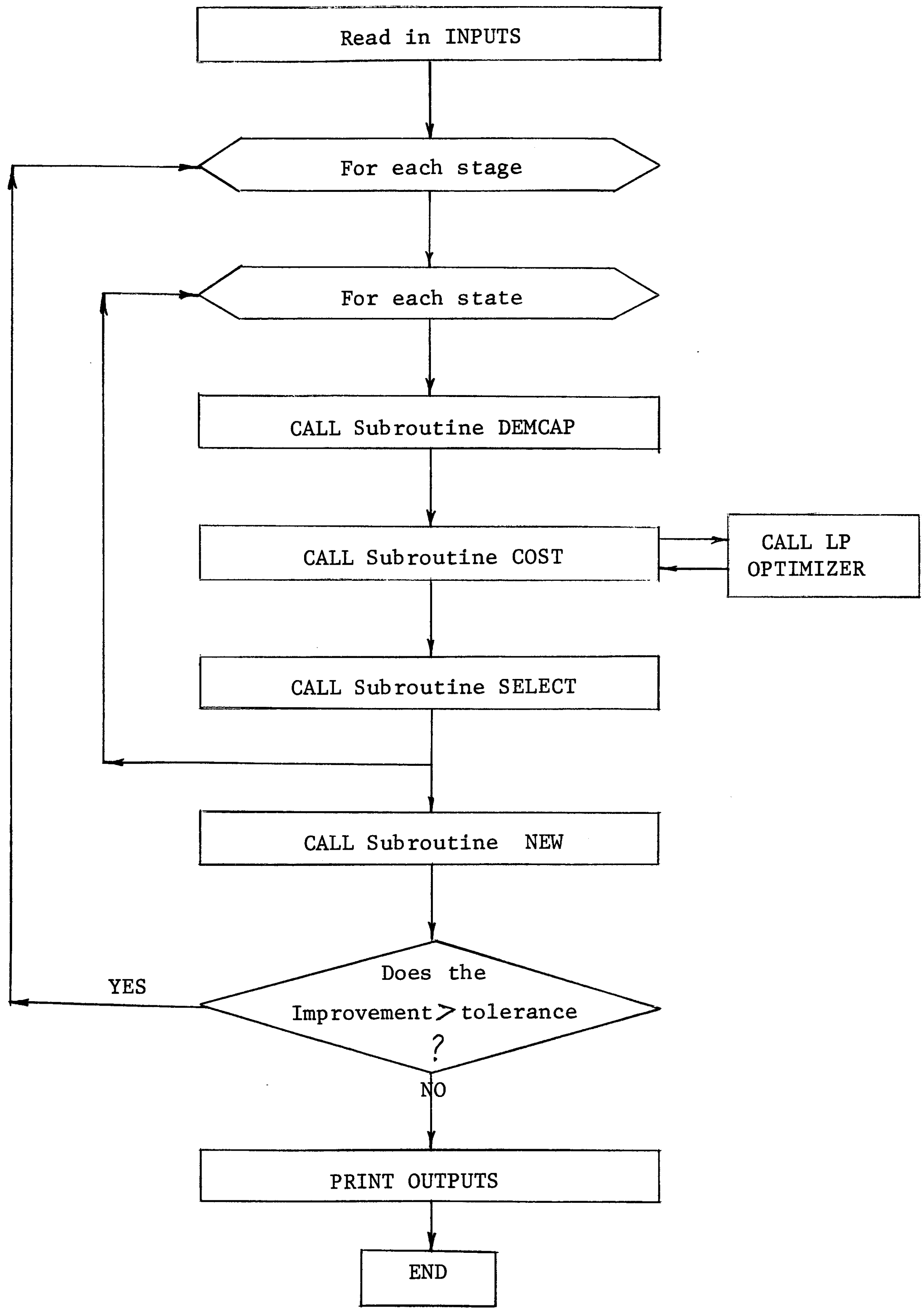


Figure 4-1 : Flow Chart of Dynamic Programming Program

Validity of Dynamic Programming to Total Network Problem

This particular Dynamic Programming model is applicable to network design only if decisions made in different states lead to consequences that are mutually independent. This is a critical factor in transportation networks where the links in any system are dependent. However, water supply networks may behave differently.

What is required is a definition of the appropriate existing network for consideration of additional capacity in any stage. Moreover, this updated existing network must reflect an optimal expansion policy up to that time. Since the linear programming formulation may specify branching (parallel) pipes to be built, equivalent pipe networks for the optimal expansion to each state must be defined after each stage. Consider the example Table 3-2 of Chapter 3. Suppose one is in the second stage, and S_I = 30 years and the range of decisions, D_I, are being considered. Each entry in the S_{I-1} column represents a different existing system for input into the linear program, and the decision is to add capacity D_I after S_{I-1} has been built. Clearly S_{I-1} must itself correspond to an optimal expansion involving one or more projects in previous stages. To consider the third stage, the optimal (minimum cost) expansions S_I in the second stage become, after the hydraulically equivalent network has been computed, the S_{I-1} variable for the third stage. This updating process prevents dependency and thus allows application of the one-dimensional approach.

Computer Programs

A special computer program which links the Linear Program and the Dynamic Program was prepared. The program consists of a main program and eight subprograms. The main program reads in input data, writes out output results and serves as a monitor program which controls the subsequent order of operations. The subroutine DEMCAP computes the demands at each node. The sum of demands is taken to be the total amount of supply. The subroutine COST calls the Linear Programming routine and converts the total system cost to net present value. The subroutines NCOST, ORGLP, LPROG, SIMPLX together comprise the Linear Programming model which has been explained in Chapter 2. The subroutine SELECT chooses the minimum cost decision given the state of the system. Subroutine NEW updates the existing system by using hydraulically equivalent pipes for two parallel pipes (existing and new pipes). Figure 4-1 is a flow chart of the over-all program. The listings in the program are given in Appendix B.

An example

The format of the input data is illustrated in detail in Fig. 4-2. The example chosen is concerned with the reliability of the existing New York City primary water supply tunnel system whose configuration is shown schematically in Fig. 4-3. Studies of the system have indicated that the failure of either pipe 1 or pipe 15 has the most severe impact upon the system. Assuming that pipe 15 fails to function it may be possible to meet the demands by constructing a pumping station at node 9 and by constructing a long tunnel directly from node 1 to node 9 so that the demands at nodes 11, 12, 13, and 14 are supplied through node 9. Since the minimum allowable head is 250 feet, the heads at nodes 8 and 15 are taken to be 250 feet. The head at node 9 before pumping is assumed to be 200 feet and after pumping 300 feet. The heads

Figure 4-2

INPUT FIELD FORMAT

Group 1 (4 F 10.1)

(i) Economic Time Horizon in Years

(ii) Interval in Years of Each State

(iii) Discount Rate

(iv) Cost Reduction or Increase in the Future

Group 2 (4 F 10.1)

(i) Number of Pipes

(ii) Number of Nodes

(iii) Hazen-Williams Coefficient

(iv) Scaling Factor for Coefficient Matrix A

Group 3 (6 F 10.1)

(i) Pipe Identification Number

(ii) Upstream Node Number

(iii) Downstream Node Number

(iv) Length of the Pipe

(v) Existing Pipe Diameter in Inches

(vi) Number Identifies the Estimate of Pipe Size

Group 4 (3 F 10.1)

(i) Node Identification Number

(ii) Demand or Supply Rate in cfs

(iii) Energy Head in ft.

Figure 4-2 (continued)

<u>Group 5</u> (3 F 10.1)

(i) Initial Total Population in Millions

(ii) Total Population in Millions at the End of Economic Time Horizon

(iii) Number of Boroughs

<u>Group 6</u> (2 F 10.3)

(i) Number of Nodes Allocated to a Borough

(ii) Initial Total Population in Millions in a Borough

<u>Group 7</u> (10 F 5.1)

Location of Each Node Given in Borough Number and demands at each node for year 2010 are shown in Fig. 4-4. The economic time horizon is here assumed to be 40 years. Also assumed is a linear demand growth rate 0.074 and discount rate of 3 per cent.

The summary of the results of the dynamic program for this particular example is tabulated in Fig. 4-5. The table indicates that it is optimal to construct the additional facilities at the present time to satisfy the demands of 40 years from now. The additional tunnels required are shown in dash lines in Fig. 4-4. The present investment for the tunnel alone is estimated to be 69 million dollars. This does not include pumping costs.

In this example, the dynamic program called the Linear Programming model 16 times. The CPU time taken for the whole computation was about 40 seconds so only 2 seconds were used for computations by the LP model. The computation is thus considered very efficient.

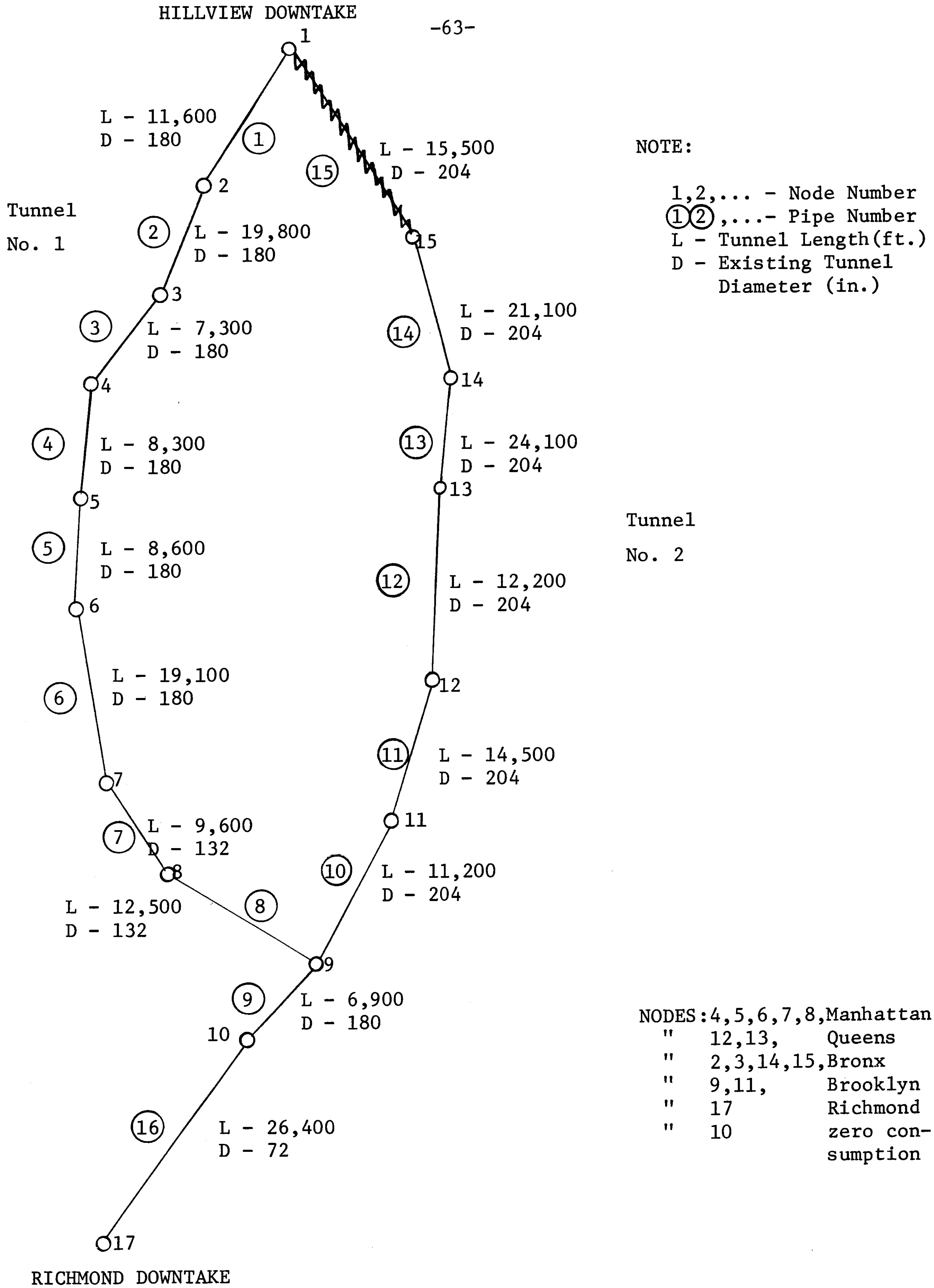


Figure 4-3: Existing System

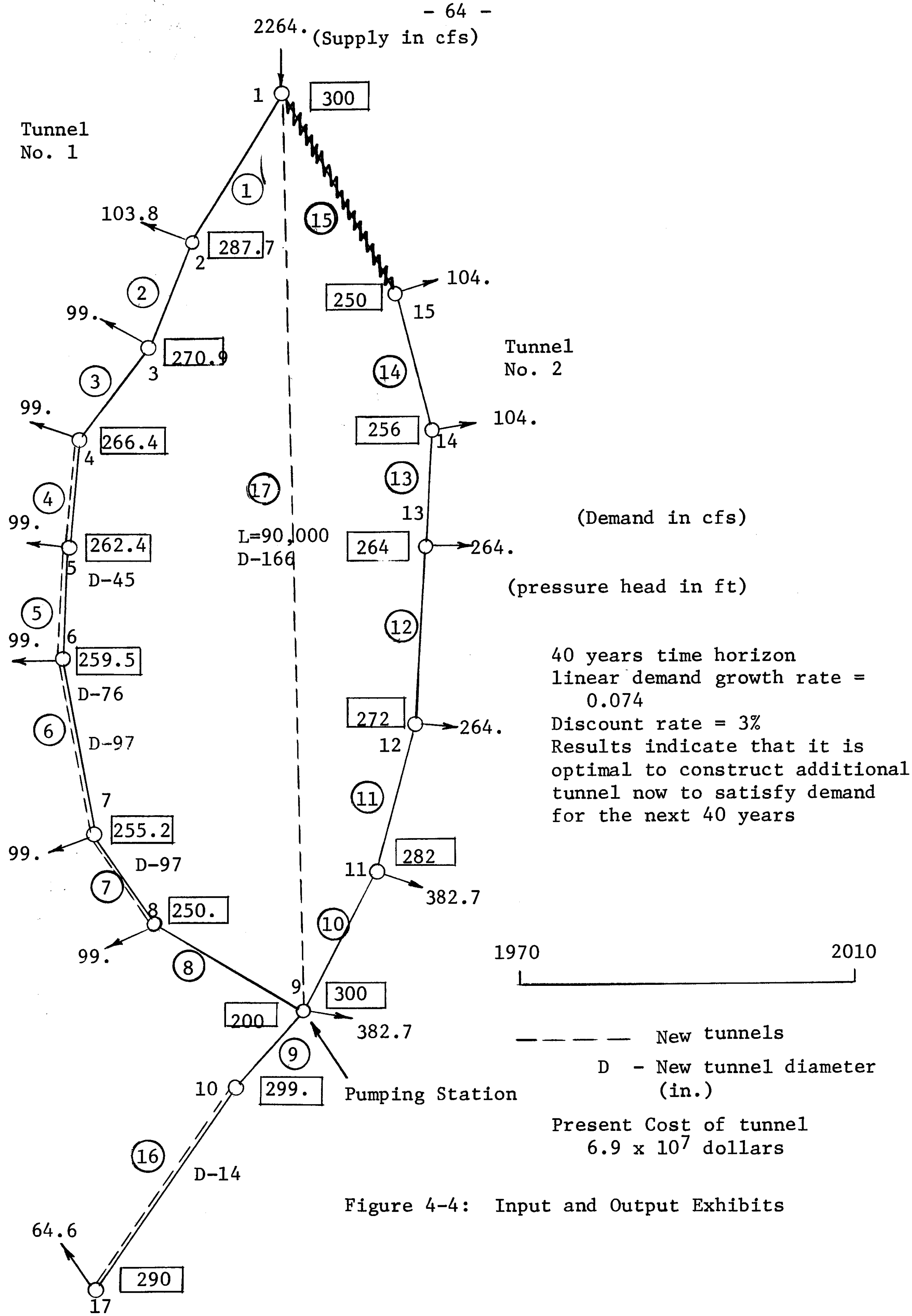


Figure 4-4: Input and Output Exhibits

CAPACITY EXPANSION BY DYNAMIC PROGRAMMING

NEW YORK CITY DATA

DESIGN PERIOD = 40.0 YEARS
TIME INTERVAL = 10.0 YEARS
DISCOUNT RATE = 0.030

	STATE	DECISION	STATE IN LAST STAGE	INCREMENTAL RETURN	TOTAL RETURN	MINIMUM RETURN FOR THIS STATE
STAGE NO. 1						
	0.0	0.0	0.0	0.0	0.0	
						0.0
	10.0	10.0	0.0	61147.4	0.0	
						61147.4
	20.0	20.0	0.0	63785.8	0.0	
						63785.8
	30.0	30.0	0.0	66136.4	0.0	
						66136.4
	40.0	40.0	0.0	69007.2	0.0	
						69007.2
STAGE NO. 2						
	0.0					
		0.0	0.0	0.0	0.0	
						0.0
	10.0					
		0.0	10.0	0.0	61147.4	
		10.0	0.0	61147.4	61147.4	
						61147.4
	20.0					
		0.0	20.0	0.0	63785.8	
		10.0	10.0	11389.2	72536.6	
		20.0	0.0	63785.8	63785.8	
						63785.8
	30.0					
		0.0	30.0	0.0	66136.4	
		10.0	20.0	8474.7	72260.4	
		20.0	10.0	15791.3	76938.6	
		30.0	0.0	66136.4	66136.4	
						66136.4
	40.0					
		0.0	40.0	0.0	69007.2	
		10.0	30.0	6529.9	72666.4	
		20.0	20.0	12051.2	75836.9	
		30.0	10.0	19522.3	80669.8	
		40.0	0.0	69007.2	69007.2	
						69007.2

Figure 4-5 SUMMARY OF THE OUTPUT RESULTS

REFERENCES

1. Urban Water Resources Research, A Study for OWRR, ASCE, New York, New York, September, 1968.

2. Pitchai, R., "A Model for Designing Water Distribution Pipe Networks", Ph.D. Thesis, Harvard University, 1966.

3. Shamir, U. and Howard, C. D.D., "Water Distribution System Analysis", J. Hydr. Div., Proc. ASCE, Vol. 94, No. HY1, Jan. 1968, P. 219.

4. de Neufville, R. and Hester J., Discussion of Reference 3, ASCE Hydraulic Div. J., Vol. 95, No. HY1, Jan. 1969.

5. Warga, J., "Determination of Steady State Flows and Currents in a Network", Proceedings, Instrument Society of America, Vol. 9, pt. 5, 1954, Paper 54-43-4.

6. Dillingham, J. H., "Computer Analysis of Water Distribution Systems, Part II", Water and Sewage Works, Feb. 1967, pp. 43-54.

7. Jacoby, S. L. S., "Design of Optimal Hydraulic Networks", ASCE Hydraulics Div. J., Vol. 94, No. HY3, May 1968, p. 641.

8. McPherson, M. B. and Heidari, M., "Power Consumption With Elevated Storage Compared to Direct and Booster Pumping", J. AWWA, Vol. 58, No. 12, 1966, p. 1585.

9. Karmeli, D., Gadish, Y. and Meyers S., "Design of Optimal Water Distribution Networks," ASCE J. of Pipeline Div., Vol. 94, No. PL1, Oct. 1968.

10. Smith, D. V., "Minimum Cost Design of Linearly Restrained Water Distribution Networks," Master Thesis, Dept. of Civil Engineering, M.I.T., 1966.

11. Hamming, R. W., "Numerical Methods for Scientists and Engineers", New York, McGraw-Hill, 1962.

12. Linaweaver, F. P., Jr. and Clark, C. S., "Cost of Water Transmission", J. AWWA, Vol. 56, No. 12, December 1964.

13. Hirshleifer, J., de Haven, J. C. and Milliman, J. W., "Water Supply: Economics, Technology and Policy", The University of Chicago Press, Second Impression, 1963.

14. Manne, A. S., "Capacity Expansion and Probablistic Growth," Econometrica, Vol. 29, No. 4, October 1961, pp. 632-649.

APPENDIX A

COMPUTER PROGRAM LISTING

FOR LINEAR PROGRAMMING NETWORK MODEL

```
      DIMENSION IP(50),JN(50),KN(50),FL(50),
     1EXD(50),HL(50),IN(50),Q(50),H(50),DIAUSE(50),FLOW(50),
     2XIP(50),XJN(50),XKN(50),XIN(50),KESDIA(50),ESDIA(50),DIANEW(50)
      IWRITE=6
      JWRITE=6
      WRITE(IWRITE,1)
1     FORMAT(1X,' INPUT FILE NUMBER')
      READ(5,2) IREAD
 2    FORMAT(I1)
      READ(IREAD,5001) XNP,XNN,HWC,FACTOR
5001  FORMAT(4F10.1)
      NP=XNP
      NN=XNN
      DO 10 I=1,NP
      READ(IREAD,5002) XIP(I),XJN(I),XKN(I),FL(I),EXD(I),ESDIA(I)
      IP(I)=XIP(I)
      JN(I)=XJN(I)
      KN(I)=XKN(I)
      KESDIA(I)=ESDIA(I)
10     CONTINUE
5002  FORMAT(6F10.1)
      DO 20 I=1,NN
      READ(IREAD,5002) XIN(I),Q(I),H(I)
      IN(I)=XIN(I)
 20     CONTINUE
      CALL NCOST(NN,NP,IN,IP,JN,KN,FL,EXD,Q,H,HWC,TCOST,DIANEW,DIAUSE,
     1KESDIA,OBJ,FLOW,FACTOR)
      WRITE(JWRITE,6009) TCOST
6009  FORMAT(//20X,'TOTAL COST =',F15.2,1X,'DOLLARS')
      OBJ=-OBJ
      WRITE(JWRITE,6005) (IP(I),DIANEW(I),EXD(I),DIAUSE(I),I=1,NP)
6005  FORMAT(///29X,' NEW ',5X,'EXISTING',8X,' USED ',/20X,'PIPE',4X,
     1'DIAMETER',          4X,'DIAMETER'7X,'DIAMETER',/(20X,I3,2X,F8.2,
     25X,F8.2,8X,F8.2))
C     WRITE(JWRITE,6010) OBJ
6010  FORMAT(1X,'OBJECTIVE FUNCTION=',E12.5)
C     WRITE(JWRITE,105)
105   FORMAT(1X,'FLOW IN EACH PIPE ARE')
C     DO 106 I=1,NP
C     WRITE(JWRITE,104) I,FLOW(I)
104   FORMAT(5X,3HQQ(,I2,3H )=,F10.3)
106   CONTINUE
      END
```

```
      SUBROUTINE NCOST(NODES,NTUNL,INODES,ITUNL,IUP,IDOWN,FLGTH,
     1EXIST,DEMND,HEAD,HWC,TCOST,DIANEW,DIAUSE,KESDIA,OBJ,FLOW,FACTOR)
      DIMENSION INODES(50),ITUNL(50),IUP(50),IDOWN(50),FLGTH(50),
     1EXIST(50),DEMND(50),HEAD(50),DIA(100),A(5000),KESDIA(50),
     2DIANEW(50),SP(100),FLOW(50),DIAUSE(50),ISS(10)
      DIMENSION HL(50),FK(50),ALPHP(50),KCODIA(50),B(50)
      NMAX=5000
      IWRITE=6
      JWRITE=8
C  COUNT THE NUMBER OF EXISTING PIPES
      NEP=0
      DO 15 I=1,NTUNL
      IF(EXIST(I)-.01) 15,15,16
 16   NEP=NEP+1
 15   CONTINUE
      NSUPY=0
      DO 500 I=1,NODES
      IF(DEMND(I))501,502,502
501   NSUPY=NSUPY+1
      ISS(I+1)=1
      GO TO 500
502   ISS(I+1)=0
500   CONTINUE
      WRITE(IWRITE,55)
55    FORMAT(1X,' INPUT INDICATOR, 1=PRINT OUT NODES ,NTUNL,NEP, 0=NOT')
      READ(5,2) IPRIN1
2     FORMAT(I1)
      IF(IPRIN1) 23,23,52
52    WRITE(JWRITE,6001) NTUNL,NEP,NODES
 6001 FORMAT (///10X,24HTOTAL NUMBER OF PIPES = ,I3/
     1           20X,27HNUMBER OF EXISTING PIPES = ,I3/
     2           30X,18HNUMBER OF NODES = ,I3)
23    DO 11 I=1,NTUNL
11    HL(I)=0.
      DO 5 I=1,NTUNL
      II=0
      JJ=0
      DO 6 J=1,NODES
      IF(IUP(I).EQ.INODES(J)) II=J
      IF(IDOWN(I).EQ.INODES(J)) JJ=J
 6    CONTINUE
      IF(II) 5,5,7
7     IF(JJ)5,5,9
9     HL(I)=HEAD(II)-HEAD(JJ)
5     CONTINUE
      WRITE(IWRITE,56)
56    FORMAT(1X,'INPUT INDICATOR, 1=PRINT NODES AND PIPES CONDITIONS,
     1 0=NOT')
      READ(5,2) IPRIN2
      IF(IPRIN2) 100,100,53
53    WRITE(JWRITE,6002)
6002  FORMAT(///12X,4HNODE,17X,8HEXISTING,5X,'HEAD LOSS',/1X,4HPIPE,
     14X,5HBEGIN,2X,3HEND,4X,6HLENGTH,4X,8HDIAMETER)
      DO 19 I=1,NTUNL
19    WRITE(JWRITE,6003) ITUNL(I),I P(I),IDOWN(I),FLGTH(I),EXIST(I),HL(I
```

```
     1)
 6003 FORMAT(I4,6X,I3,3X,I3,3X,F7.0,6X,F4.0,6X,  F7.1)
      WRITE(JWRITE,6008)
6008  FORMAT(//1X,'NODE',10X,'DEMAND(MPD)')
      DO 25 I=1,NODES
25    WRITE(JWRITE,6004) INODES(I),DEMND(I)
6004  FORMAT(I4,10X,F10.0)
100   DO 30 I=1,NTUNL
30    FK(I)=.00062*HWC*FLGTH(I)**(-0.54)
      M=NODES+NEP+1
      N=NMAX/M
      CALL ORGLP(M,N,A,FK,EXIST,DEMND,INODES,IUP,IDOWN,NTUNL,NODES,B,
     1HL,NEP,JMAX,NSUPY,ISS,FACTOR)
      ALPHA=1.1
      ITEND=10
      ITET=1
131   IF(ITET.GT.ITEND) GO TO 132
      DO 21 I=1,NTUNL
      KKK=KESDIA(I)
      GO TO (121,122,123,124,125,126), KKK
121   ALPHP(I)=3.72E-03
      GO TO 21
122   ALPHP(I)=9.75E-04
      GO TO 21
123   ALPHP(I)=4.84E-04
      GO TO 21
124   ALPHP(I)=2.41E-04
      GO TO 21
125   ALPHP(I)=2.16E-04
      GO TO 21
126   ALPHP(I)=1.58E-04
21    CONTINUE
      DO 129 I=1,NTUNL
      JJ=M*(I-1)+1
      A(JJ)=ALPHP(I)*FLGTH(I)
C     WRITE(JWRITE,601) JJ,A(JJ)
 601    FORMAT(1X,'A( ',I5,')=',F11.2)
129   CONTINUE
      IF(NSUPY) 503,503,504
504   DO 505 I=1,NSUPY
      JJ=JJ+M
      A(JJ)=0.
C      WRITE(JWRITE,601)JJ,A(JJ)
505   CONTINUE
503   IF(NEP) 250,250,251
251   JJJ=M*(NTUNL+NEP+NSUPY-1)+1
      DO 253 K=1,NEP
      JJ=JJ+M
      A(JJ)=0.
C     WRITE(JWRITE,601) JJ,A(JJ)
      JJJ=JJJ+M
      A(JJJ)=0.
C     WRITE(JWRITE,601)JJJ,A(JJJ)
253   CONTINUE
250   CONTINUE
```

```
      DO 254 I=1,JMAX
254   DIA(I)=0.
      CALL LPROG(M,M,JMAX,A,B,SP,DIA,OBJ)
      DO 133 I=1,NTUNL
      IF(DIA(I)-(60.**2.63)) 181,181,182
181   KCODIA(I)=1
      GO TO 133
182   IF(DIA(I)-(120.**2.63)) 183,183,184
183   KCODIA(I)=2
      GO TO 133
184   IF(DIA(I)-(180.**2.63)) 185,185,186
185     KCODIA(I)=3
      GO TO 133
186   IF(DIA(I)-(240.**2.63)) 187,187,188
187   KCODIA(I)=4
      GO TO 133
188   IF(DIA(I)-(300.**2.63)) 189,189,190
189   KCODIA(I)=5
      GO TO 133
190   KCODIA(I)=6
133   CONTINUE
      ICONT=0
      DO 127 I=1,NTUNL
      IF(KESDIA(I).NE.KCODIA(I)) ICONT=ICONT+1
127   CONTINUE
212   FORMAT(10F11.2)
      IF(ICONT.EQ.0) GO TO 132
      DO 128 I=1,NTUNL
      KESDIA(I)=KCODIA(I)
128   CONTINUE
      ITET=ITET+1
      GO TO 131
132   WRITE(JWRITE,134)ITET
134   FORMAT(1X,'ITERATIONS DONE ON LPROG=',I10)
      WRITE(JWRITE,212) (DIA(I),I=1,JMAX)
      KK=0
      DO 50 I=1,NTUNL
103   DIANEW(I)=(DIA(I)*FACTOR)**(1./2.63)
      IF(EXIST(I)-.01) 255,255,256
256   KK=KK+1
      DIAUSE(I)=(DIA(NTUNL+NSUPY+KK)*FACTOR)**(1./2.63)
      GO TO 50
255   DIAUSE(I)=0.
C     IF(HL(I)) 101,102,101
C     FLOW(I)=(FK(I)*(DIANEW(I)**2.63+DIAUSE(I)**2.63)*
C    1ABS(HL(I))**.54*HL(I)/(ABS(HL(I))*100.))*100.
C     GO TO 50
C     FLOW(I)=0.
50    CONTINUE
      TCOST=0.
      DO 35 I=1,NTUNL
      IF(DIANEW(I).LE.0.) GO TO 35
36    TCOST=TCOST+FLGTH(I)*1.1*DIANEW(I)**1.24
35    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORGLP (M,N,A,FK,EXD,Q,IN,JN,KN,NP,NN,B,HL,NEP,JMAX,
     1NSUPY,ISS,FACTOR)
      DIMENSION A(M,N),FK(50),EXD(50),FL(50),Q(50),IN(50),JN(50),
     1KN(50),B(50),HL(50),Z(50),ISS(10)
      B(1)=0.
      IWRITE=6
      DO 10 K=1,NN
10    B(K+1)=ABS(Q(K))
      J=1
      DO 30 K=1,NN
      J=J+1
      Z(J)=0.
      IF(Q(K)) 1,1,2
    1   SIG=1.
      GO TO 3
2     SIG=-1.
3     DO 25 I=1,NP
      IF(HL(I)) 101,22,101
101    IF(JN(I)-IN(K)) 20,21,20
21    A(J,I)=SIG*FACTOR*FK(I)*ABS(HL(I))**.54*HL(I)/ABS(HL(I))
      GO TO 25
20    IF(KN(I)-IN(K)) 22,23,22
23    A(J,I)=-SIG*FACTOR*FK(I)*ABS(HL(I))**.54*HL(I)/ABS(HL(I))
      GO TO 25
22    A(J,I)=0.
25    Z(J)=Z(J)+A(J,I)**2
      Z(J)=SQRT(Z(J))
      IF(ABS(Z(J))-.00001) 26,26,30
26    WRITE(8,27) K
27    FORMAT(1X,'NODE',I5,'SHOULD BE IGNORED')
30    CONTINUE
      IF(NSUPY)200,200,201
201   DO 202 I=1,NSUPY
      KK=NP+I
      DO 203 KKK=2,J
      IF(ISS(KKK)) 204,204,205
205   A(KKK,KK)=1.
      ISS(KKK)=0.
      KKKK=KKK+1
      DO 206 JJ=KKKK,J
206   A(JJ,KK)=0.
      GO TO 202
204   A(KKK,KK)=0.
203   CONTINUE
202   CONTINUE
C   CONSIDERATION OF EXISTING PIPES
200   LL=0
      DO 50 I=1,NP
      IF(EXD(I)-.01) 50,50,51
51    J=J+1
      B(J)=EXD(I)**2.63/FACTOR
      LL=LL+1
      NNP=NP+LL+NSUPY
      DO 52 K=1,NNP
52    A(J,K)=0.
```

```
      DO 53 JJ=2,J
53    A(JJ,NNP)=A(JJ,I)
      A(J,NNP)=1.
50    CONTINUE
      JMAX=NP+2*NEP+NSUPY
C   ADD SLACK VARIABLES TO CORRESPONDING RXISTING PIPES
      IF(NEP) 80,80,81
81     DO 60 K=1,NEP
      I=NP+NEP+K+NSUPY
      II=NN+1+K
      DO 70 KK=2,J
70    A(KK,I)=0.
      A(II,I)=1.
60    CONTINUE
80    WRITE(IWRITE,63)
63    FORMAT(1X,'INPUT INDICATOR,1=PRINT OUT A MATRIX,0=NOT')
      READ(5,64) INDICA
64    FORMAT(I1)
      IF(INDICA) 65,65,66
66    WRITE(8,6001)
6001  FORMAT('  COEFFICIENT MATRIX')
      DO 71 I=2,M
71    WRITE(8,6002) I,(A(I,K),K=1,JMAX)
6002  FORMAT(/1X,4HROW ,I2,1X,10F12.10/(8X,10F12.10))
      WRITE(8,6003)
6003  FORMAT(' CONSTRAINT MATRIX')
      WRITE(8,6004) (B(I),I=1,M)
6004  FORMAT(8X,10F11.2)
65    RETURN
      END
```

```
      SUBROUTINE LPROG(ME,M,N,A,B,Z,DIA,OBJ)
      DIMENSION A(1),B(1),Z(1),DIA(1),TITLE(1),INFIX(8),TOL(4),E(3000),
     1         KOUT(7),ERR(8),JH(100),X(100),P(100),Y(100),KB(100)
      INFIX(1) = 4
      INFIX(2) = N
      INFIX(3) = ME
      INFIX(4) = M
      INFIX(5) = 2
      INFIX(6) = 1
      INFIX(7) = 100
      INFIX(8) = 0
      TOL(1)=10.**(-4)
      TOL(2)=10.**(-4)
      TOL(3) = -10.**(-3)
      TOL(4) = 10.**(-10)
      WRITE(6,92)
92    FORMAT(' OUTPUT FROM LPROG')
      WRITE(6,91) N,ME,M
91    FORMAT(' N=',I10,'  ME=',I10,'  M=',I10)
      PRM = 0.
      B(1)=0.
      CALL SIMPLX (INFIX,A,B,TOL,PRM,KOUT,ERR,JH,X,P,Y,KB,E)
      DO 1 I=1,N
    1 Z(I) = 0.
      DO 2 I=1,N
      J = KB(I)
      IF (J) 2,2,3
    3 Z(I) = X(J)
    2 CONTINUE
      DO 5 I=1,N
    5 DIA(I)=Z(I)
      OBJ = Y(1)
11     WRITE(6,6500) (KOUT(I),I=1,7)
6500  FORMAT(7I10)
6502  FORMAT(4E12.5)
      RETURN
      END
```

```
      SUBROUTINE SIMPLX (INFIX,A,B,TOL,PRM,KOUT,ERR,JH,X,P,Y,KB,E)
C
      DIMENSION INFIX(8),A(1),B(1),TOL(4),KOUT(7),ERR(8),JH(1),X(1),
     1 P(1),Y(1),KB(1),E(1),ZZ(3), IOFIX(16) , TERR(8)
C
      EQUIVALENCE  (INFLAG,IOFIX(1) ), (N , IOFIX(2) ) ,
     1            (ME,IOFIX(3) ),  (M,IOFIX(4)), (MF,IOFIX(5)),
     2  (MC, IOFIX(6) ),  ( NCUT, IOFIX(7) ) , ( NVER, IOFIX(8) ),
     3 ( K, IOFIX(9) ), (ITER, IOFIX(10) ), (INVC , IOFIX(11) ) ,
     4 (NUMVR, IOFIX(12) ), ( NUMPV, IOFIX(13) ) ,
     5  (INFS, IOFIX(14)  ) , ( JT,  IOFIX(15) ) ,( LA  , IOFIX(16) ),
     6 (ZZ(1),TPIV), (ZZ(2),TZERO),(ZZ(3),TCOST)
C
C                           MOVE INPUTS  ...  ZERO OUTPUTS
      DO 1340  I= 1, 8
       TERR(I) = 0.0
      IOFIX(I+8)  =  0
 1340  IOFIX(I)  =  INFIX(I)
      LA   =   0
      DO 1308  I = 1 , 3
 1308 ZZ(I)  =  TOL(I)
      TCOST = - ABS   (TCOST)
      PMIX = PRM
      M2 = M**2
      INFS = 1
C                     CHECK FOR ILLEGAL INPUT
      IF (N)  1304, 1304, 1371
 1371  IF (M - MF ) 1304, 1304, 1372
 1372  IF (MF - MC)  1304, 1304, 1373
 1373  IF ( MC ) 1304 , 1304,  1374
 1374  IF (ME - M ) 1304, 1375, 1375
 1375 IF(  MOD(INFLAG, 4) -1)1400, 1320,100
C
C
C  NEW 1            STARTS PHASE ONE
C*****SUBROUTINE NEW (M,N, JH, KB, A, B, MF, ME )
C
C                              INITIATE
 1400   DO 1401 I = 1, M
 1401 JH(I) = 0
C                   INSTALL SINGLETONS
       KT  =  0
        DO 1402  J = 1, N
      KB(J) = 0
      MM  =  KT + MF
      LL  =  KT + M
C                             TALLY ENTRIES IN CONSTRAINTS
      KQ  = 0
```

```
      DO  1403 L = MM , LL
      IF (A(L)) 1404, 1403, 1404
 1404 KQ = KQ+1
      LQ = L
 1403   CONTINUE
C                                  CHECK WHETHER J IS CANDIDATE
            IF (KQ - 1) 1402, 1405, 1402
 1405 IA = LQ- KT
         IF ( JH(IA) ) 1402, 1406, 1402
 1406 IF (A(LQ)*B(IA)) 1402, 1407, 1407
C                                  J IS CANDIDATE. INSTALL
 1407 JH(IA) = J
      KB(J)  =  IA
 1402 KT = KT + ME
C
C**END OF NEW
C
C
 1320 CONTINUE
C
C  VER 1              FORMS INVERSE FROM  KB
C*****SUBROUTINE VER ( A, B, JH, X, E, KB, Y, IOFIX, TPIV, M2  )
C
C                      INITIATE
 1100 ASSIGN  1102  TO KPIV
      ASSIGN  1114  TO KJMY
      IF (LA)  1121, 1121, 1122
 1121 INVC = 0
 1122 NUMVR   =  NUMVR  +1
       DO 1101  I = 1, M2
 1101 E(I)=0.
      MM=1
       DO 1113  I = 1, M
      E(MM) =1.0
      X(I) = B(I)
 1113 MM = MM + M + 1
         DO 1110  I = MF, M
      IF (JH(I)) 1111, 1110, 1111
 1111 JH(I) = 12345
 1110 CONTINUE
      INFS = 1
C                     FORM INVERSE
         DO 1102  JT= 1, N
      IF ( KB(JT))  600 , 1102 , 600
C 600 CALL  JMY  (JT, A, E, M, Y )
C                               CHOOSE PIVOT
 1114 TY = 0.
         DO 1104  I = MF, M
```

```
      IF (JH(I)  - 12345 )  1104, 1105, 1104
 1105 IF (ABS (Y(I) ) - TY ) 1104, 1104, 1106
 1106 IR  = I
      TY =  ABS  (Y(I) )
 1104   CONTINUE
C                         TEST PIVOT
      IF  (TY - TPIV )    1107, 1108, 1108
C        BAD  PIVOT,  ROW IR,  COLUMN JT
 1107 KB(JT)= 0
      GO TO 1102
C                         PIVOT
 1108 JH(IR) = JT
      KB(JT) = IR
      GO TO 900
C 900 CALL PIV    (IR, Y, M, E, Z, X)
 1102      CONTINUE
C                    RESET ARTIFICIALS
        DO 1109  I = 1, M
      IF ( JH(I) - 12345  )  1109, 1112, 1109
 1112 JH(I) = 0
 1109   CONTINUE
C**END OF VER
C
C
  100 ASSIGN   705  TO  NDEL
      ASSIGN  1000  TO  KJMY
      ASSIGN   221  TO  KPIV
C
C                               PERFORM ONE ITERATION
C
C  XCK 1     X CHECKER
C*****SUBROUTINE XCK ( M, MF, JH, X, TZERO, JIN )
C
C              RESET X AND CHECK FOR INFEASIBILITIES
 1200   JIN  =  0
      NEG = 0
      DO 1201 I = MF, M
      IF   ( ABS (X(I) ) - TZERO)  1202, 1203, 1203
 1202 X(I) = 0.0
      GO TO 1201
 1203 IF ( X(I) )  1208, 1201, 1205
 1205 IF ( JH(I) )  1201, 1206, 1201
 1208 NEG = 1
 1206   JIN = 1
 1201   CONTINUE
C**END OF XCK
C
C
```

```
C             CHECK CHANGE OF PHASE.. GO BACK TO INVERT IF GONE INFEAS.
      IF (INFS - JIN ) 1320,  500,  200
C                      BECOME FEASIBLE
  200 INFS = 0
  201 PMIX  =  0.0
C
C  GET 1     GET PRICES
C*****SUBROUTINE GET ( M, MC, MF, JH, X, P, E, INFS, PMIX )
C
  500 MM  = MC
C                              PRIMAL PRICES
  502 DO 503 J = 1, M
      P(J) = E(MM )
  503 MM  =MM  + M
      IF ( INFS )  501, 599, 501
C                              COMPOSITE PRICES
  501 DO 504  J = 1, M
  504 P(J) = P(J)* PMIX
      DO 505  I = MF, M
      MM  =I
      IF  ( X(I) )  506, 507, 507
  506 DO 508  J = 1, M
      P(J) =  P(J) + E(MM )
  508 MM  = MM  + M
      GO TO 505
  507 IF (JH(I))  505, 509, 505
  509 DO 510  J = 1, M
      P(J) = P(J) - E(MM )
  510 MM  =  MM  +M
  505   CONTINUE
C
  599 CONTINUE
C**END OF GET
C
C
C
C    MIN          MIN D-J.  SELECTS COLUMN TO ENTER BASIS
C*****SUBROUTINE MIN ( JT, N, M, A, P, KB, MF, TCOST )
C
  700 JT = 0
      BB = TCOST
C
  701 DO  702  JM = 1, N
C                                   SKIP COLUMNS IN BASIS
  703 IF ( KB(JM) )  702, 300, 702
C 300 CALL DEL ( JM, DT, M, A, P)
  705 IF  ( DT - BB )  708, 702, 702
  708 BB = DT
```

```
      JT = JM
  702    CONTINUE
C
C**END OF MIN
C
         IF  (JT)    203, 203, 600
C             ALL COSTS NON-NEGATIVE...  K = 3 OR 4
  203 K = 3 + INFS
      GO TO 257
C                                   NORMAL CYCLE
C
C  JMY 1          J MULTIPLY.  BASIS INVERSE * COLUMN JT
C*****SUBROUTINE JMY (JT, A, E, M, Y, ME )
C
  600 DO 610   I= 1,M
  610 Y(I) =0.
      LP   =   JT*ME - ME
      LL = 0
      DO 605    I= 1,M
      LP = LP + 1
      IF  (A(LP))    601, 602, 601
  601 DO 606   J   =   1,M
      LL = LL + 1
  606 Y(J) = Y(J) + A(LP) * E(LL)
      GO TO 605
  602 LL = LL + M
  605 CONTINUE
C
  699 GO TO KJMY , ( 1000  , 1114 ,  1392  )
C**END OF JMY
C
C
C  ROW 1    ROW SELECTION--COMPOSITE
C*****SUBROUTINE  ROW  ( IR, M, MF, JH, X, Y, TPIV )
C
C AMONG EQS. WITH X=0, FIND MAX ABS(Y) AMONG ARTIFICIALS, OR, IF NONE,
C  GET MAX POSITIVE Y(I) AMONG REALS.
 1000 IR = 0
      AA = 0.0
      IA   =   0
      DO 1050   I = MF,  M
      IF ( X(I) )  1050, 1041, 1050
 1041 YI= ABS (Y(I) )
      IF  (  YI   -   TPIV    )   1050, 1050, 1042
 1042 IF  ( JH(I) )  1043, 1044, 1043
 1043 IF  (IA)    1050, 1048, 1050
 1048  IF  ( Y(I) )  1050, 1050, 1045
 1044 IF   (IA)    1045, 1046, 1045
```

```
 1045 IF   (  YI    -  AA    )   1050, 1050, 1047
 1046     IA = 1
 1047 AA  =   YI
      IR  =  I
 1050 CONTINUE
      IF(IR)1099,1001,1099
 1001 AA=1.0E+20
C                   FIND MIN. PIVOT AMONG POSITIVE EQUATIONS
      DO 1010 IT = MF , M
      IF  ( Y(IT)   -   TPIV )  1010, 1010, 1002
 1002 IF  ( X(IT) ) 1010, 1010, 1003
 1003 XY = X(IT) / Y(IT)
      IF ( XY - AA )  1004, 1005, 1010
 1005 IF ( JH(IT))  1010, 1004, 1010
 1004 AA = XY
      IR = IT
 1010 CONTINUE
      IF (NEG)   1016, 1099, 1016
C  FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS THAN THE
C MINIMUM X/Y IN THE POSITIVE EQUATIONS, THAT HAS THE LARGEST ABSF(Y)
 1016 BB = - TPIV
      DO 1030 I  = MF , M
      IF (X(I))   1012, 1030, 1030
 1012  IF   ( Y(I)     -   BB   )   1022, 1030, 1030
 1022 IF   (  Y(I) * AA  -   X(I)    )   1024, 1024, 1030
 1024 BB   =   Y(I)
      IR   =   I
 1030 CONTINUE
 1099 CONTINUE
C**END OF ROW
C
C                                   TEST PIVOT
  206   IF( IR )   207, 207, 210
C                                   NO PIVOT
  207 K = 5
  257 IF (PMIX)   201, 400, 201
C                             ITERATION LIMIT FOR CUT OFF
  210 IF (ITER -NCUT )    900,  160,  160
C                                   PIVOT FOUND
C
C  PIV 1           PIVOT.  PIVOTS ON GIVEN ROW
C*****SUBROUTINE PIV  ( IR, Y, M, E, X, NUMPV, TECOL )
C                             LEAVE TRANSFORMED COLUMN IN Y(I)
C
  900 NUMPV  =   NUMPV  +   1
      YI  = -Y(IR)
      Y(IR) = -1.
C                                   TRANSFORM INVERSE
```

```
      LL = 0
  903 DO 904  L =  IR, M2, M
      IF  ( E(L) )  905, 914, 905
  914 LL = LL + M
      GO TO 904
  905 XY =  E(L) / YI
      E(L) =0.
      DO 906  I  = 1, M
      LL= LL +1
  906 E(LL) = E(LL) +XY* Y(I)
  904 CONTINUE
C                                     TRANSFORM X
       XY = X(IR) / YI
      X(IR) = 0.
      DO 908  I  = 1, M
  908 X(I) = X(I) +XY* Y(I)
C                               RESTORE Y(IR)
      Y(IR) = -YI
C
  999  GO TO KPIV , ( 221, 1102 )
C**END OF PIV
C
  221  IA   = JH(IR)
      IF ( IA ) 213, 213, 214
  214 KB( IA ) = 0
  213 KB(JT)    = IR
      JH(IR)    = JT
      LA   =   0
      ITER    = ITER    +1
      INVC    = INVC    +1
C                               INVERSION FREQUENCY
      IF (INVC - NVER  ) 1200, 1320,1200
C                                          CUT OFF ... TOO MANY ITERATIONS
  160 K = 6
C
C
C  ERR 1              ERROR CHECK.  COMPARES AX WITH B, PA WITH ZERO
C*****SUBROUTINE ERR ( M, A, B, TERR, JH, X, P, Y, ME, LA )
C
C                                     STORE AX-B AT Y
  400  ASSIGN 410 TO NDEL
      DO 401 I = 1, M
  401 Y(I) =-B(I)
      DO 402 I = 1, M
      JA = JH(I)
      IF (JA) 403, 402, 403
  403 IA =ME* (JA-1)
      DO 405 IT = 1, M
```

```
      IA = IA + 1
      IF(A(IA) )  415, 405, 415
  415 Y(IT) =Y(IT) +X(I) * A(IA)
  405 CONTINUE
  402 CONTINUE
C                                  FIND SUM AND MAXIMUM OF ERRORS
      DO 481  I = 1, M
      YI  = Y(I)
      IF ( JH(I) )  472, 471, 472
  471  YI = YI + X(I)
 472  TERR (LA+1) = TERR(LA+1) +ABS (YI)
      IF (ABS (TERR (LA+2) )-ABS (YI) ) 482,481,481
  482 TERR(LA+2) = YI
  481   CONTINUE
C                                  STORE P TIMES BASIS AT DT
      DO 411  I = 1, M
      JM = JH(I)
      WRITE (6,1036)  JM
 1036 FORMAT(//'JM='F10.3)
      IF ( JM     )  300 , 411 , 300
C 300 CALL DEL ( JM, DT, M, A, P)
 410  TERR(LA+3) =TERR (LA+3) +ABS(DT)
      IF (ABS (TERR(LA+4))  -  ABS (DT) ) 413, 411,411
  413  TERR(LA+4)  =  DT
  411 CONTINUE
C**END OF ERR
C
C
      IF (LA)   193, 191,  193
  191 LA   =  4
      IF (INFLAG  -  4  )  1320,  193,  193
  193 IF  (K-5)     1392,  194,  1392
  194 ASSIGN 1392 TO KJMY
      GO TO 600
C 600 CALL JMY ( . . . . . . . )
C     GO TO 1392
 1304 K  =  7
C                      SET EXIT VALUES
 1392   DO  1309  I=  1, 8
 1309 ERR(I)           =    TERR(I)
       DO 1329  I = 1, 7
 1329 KOUT(I)  =  IOFIX(I+8)
      RETURN
C
C    DEL          DELTA-JAY.  PRICES OUT ONE MATRIX COLUMN
C*****SUBROUTINE DEL (  JM, DT, M, A, P, ME )
C
C
```

```
  300 DT = 0.
       LL  = (JM - 1) * ME
  301 DO  303   MM  = 1, M
      LL   =  LL  + 1
      IF ( A( LL ))304, 303, 304
  304 DT = DT + P( MM ) * A ( LL )
  303 CONTINUE
C
  399   GO TO NDEL , ( 410 , 705 )
C**END OF DEL
C
        END
```

APPENDIX B

JOINT LINEAR PROGRAMMING-DYNAMIC PROGRAMMING

MODEL COMPUTER LISTINGS

```
C     CAPACITY EXPANSION FOR NEW YORK CITY BY DYNAMIC PROGRAMMING
C     WILLIAM MADDAUS     ROOM 1-371  MIT      SPRING,1969
C     DYNAMIC PROGRAMMING MONITOR
      DIMENSION IUP(50),IDOWN(50),ITUNL(50),FLGTH(50),H(50),
     1EXIST(50),KESDIA(50),INODES(50),XIP(50),XJN(50),XKN(50),
     1XIN(50),LOC(50),P(10),NUM(10),ESDIA(50),XNUM(10),XLOC(50)
      COMMON/DYNAMC/I,J,K,N,DELT,DR,G,HL,ALF85,NODES,NTUNL,S(10,10),
     1D(10,10,10),R(10,10,10),Q(10,10,10),F(10,10),ISTATE(10,10),
     2DESIGN(30,10,10),EXCON(30,10),DIAS(30,10,10),QF(50),IWRITE
C     READ PARAMETERS
      IWRITE=8
      WRITE(6,101)
101   FORMAT(1X,'INPUT FILE NUMBER')
      READ(5,102) IREAD
102   FORMAT(I1)
      READ(IREAD,100) T,DELT,DR,G,ALF85
100   FORMAT(5F10.1)
      WRITE(IWRITE,106) T,DELT,DR,G,ALF85
106   FORMAT(1X,'INPUT DATA'5F8.3)
       READ(IREAD,5001) XNP,XNN,HWC,FACTOR
5001   FORMAT(4F10.1)
       NTUNL=XNP
       NODES=XNN
       DO 2 IF=1,NTUNL
       READ(IREAD,5002) XIP(IF),XJN(IF),XKN(IF),FLGTH(IF),EXIST(IF),
     1ESDIA(IF)
       ITUNL(IF)=XIP(IF)
       IUP(IF)=XJN(IF)
       IDOWN(IF)=XKN(IF)
       KESDIA(IF)=ESDIA(IF)
2      CONTINUE
5002   FORMAT(6F10.1)
       DO 3 IJ=1,NODES
       READ(IREAD,5002) XIN(IJ),QF(IJ),H(IJ)
       INODES(IJ)=XIN(IJ)
3      CONTINUE
       READ(IREAD,5001) POI,POF,XND
       ND=XND
       WRITE(IWRITE,103) ND
103     FORMAT(5I5)
        WRITE(IWRITE,5002) POI,POF
       DO 7 I=1,ND
      READ(IREAD,5003) XN,POP
      NUM(I)=XN
      P(I)=POP
C      WRITE(IWRITE,5005) NUM(I),P(I)
7      CONTINUE
      READ(IREAD,5004) (XLOC(I),I=1,NODES)
      DO 8 I=1,NODES
      LOC(I)=XLOC(I)
C      WRITE(IWRITE,5006) LOC(I)
8      CONTINUE
5003   FORMAT(2F10.3)
5004   FORMAT(10F5.1)
5005   FORMAT(I10,F10.3)
```

```
5006   FORMAT(10I5)
       N=T/DELT+1
       I=1
       K=1
C      THE COST FOR STATE ZERO IS ZERO
       WRITE(IWRITE,110) I
       DO 5 I=1,N
       S(I,1)=0.
110    FORMAT(5X,'STAGE NUMBER'I2)
       D(I,1,1)=0.
       R(I,1,1)=0.
       F(I,1)=0.
5      CONTINUE
       I=1
C      FOR STAGE 1
       DO 6 L=1,NTUNL
6      EXCON(L,1)=EXIST(L)
       TP=0.
       DO 10 J=2,N
       TP=TP+DELT
       S(I,J)=TP
       D(I,J,K)=TP
       YEAR=1970.+(J-1)*DELT
       CALL DEMCAP(NODES,LOC,QF,POI,POF,P,NUM,YEAR,T)
       CALL COST(INODES,ITUNL,IUP,IDOWN,FLGTH,H,HWC,EXIST,KESDIA,FACTOR)
       F(I,J)=R(I,J,K)
10     CONTINUE
       CALL NEW
C      FOR ALL SUCCEEDING STAGES
       DO 50 I=2,N
       WRITE(6,110) I
       WRITE(IWRITE,110)I
C      FOR EACH STATE
       DO 45 L=1,NTUNL
       DO 45 J=1,N
45     DIAS(L,J,J)=0.0
       TP=0.
       DO 40 J=2,N
       WRITE(IWRITE,115) J
       WRITE(6,115) J
115    FORMAT(2X,'STATE NO.'I2)
       TP=TP+DELT
       S(I,J)=TP
C      FOR EACH DECISON
       TD=0.
       D(I,J,1)=0.
       R(I,J,1)=0.
       YEAR=1970.+(J-1)*DELT
       CALL DEMCAP(NODES,LOC,QF,POI,POF,P,NUM,YEAR,T)
       DO 30 K=2,J
       TD=TD+DELT
       D(I,J,K)=TD
       CALL COST(INODES,ITUNL,IUP,IDOWN,FLGTH,H,HWC,EXIST,KESDIA,FACTOR)
30     CONTINUE
       CALL SELECT
```

```
40     CONTINUE
        NSTAGE=I
       CALL NEW
       XPROV=F(I-1,N)-F(I,N)
       WRITE(IWRITE,125) I,F(I,N),XPROV
125     FORMAT(///,5X,'COST AFTER STAGE'I2,2X'IS  $'
      1F12.1,5X,'IMPROVEMENT = $'F10.1//)
       IF(XPROV .LT. 100.0) GO TO 60
50     CONTINUE
60     WRITE(IWRITE,120) F(I,N)
120    FORMAT(15X,'MINIMUM STAGED CONSTRUCTION COST =  $'F12.1)
        WRITE(IWRITE,200)
200     FORMAT(////20X'CAPACITY EXPANSION BY DYNAMIC PROGRAMMING'//)
        WRITE(IWRITE,205)
205     FORMAT(/25X'NEW YORK CITY DATA'/)
        WRITE(IWRITE,210) T,DELT,DR
210     FORMAT(30X,'DESIGN PERIOD ='F5.1,2X'YEARS',/30X'TIME INTERVAL ='F
      15.1,2X'YEARS'/30X'DISCOUNT RATE ='F5.3)
        WRITE(IWRITE,215)
215     FORMAT(//25X'STATE'3X'DECISION'2X'STATE IN'4X'INCREMENTAL'2X'TOTA
      1L'3X'MINIMUN RETURN'/43X'LAST STAGE'5X'RETURN'5X'RETURN'2X'FOR THI
      2S STATE'//)
        DO 95 I=1,NSTAGE
        WRITE(IWRITE,220) I
220     FORMAT(/,5X,'STAGE NO.'I3)
        DO 95 J=1,N
        IF(I .GT. 1) GO TO 85
        DEC=D(I,J,1)
        PREVD=0.0
        WRITE(IWRITE,225) S(I,J),DEC,PREVD,R(I,J,1),Q(I,J,1)
225     FORMAT(20X,3F10.1,3X,2F10.1)
        F(I,J)=R(I,J,1)
        GO TO 92
85      WRITE(IWRITE,230) S(I,J)
230     FORMAT(20X,F10.1)
        DO 90 K=1,J
        DEC=D(I,J,K)
        PREVD=S(I,J)-DEC
        WRITE(IWRITE,235) DEC,PREVD,R(I,J,K),Q(I,J,K)
235     FORMAT(30X,2F10.1,3X,2F10.1)
90      CONTINUE
92      WRITE(IWRITE,240) F(I,J)
240     FORMAT(74X,F10.1)
95      CONTINUE
       END
```

```
      SUBROUTINE DEMCAP(NODES,LOC,QF,POI,POF,P,NUM,YEAR,T)
      DIMENSION P(10),NUM(10),NY(10),CA(10),CM(10),LOC(50),QF(50)
      MY=2
      IWRITE=8
      ND=6
      NY(1)=1970.
      NY(MY)=YEAR
     D=1./T
      R=(POF-POI)/(POI*T)
      DT=NY(MY)-NY(1)
      DO 18 I=1,ND
     CA(I)=P(I)*(1.+R*DT)
      DN=NUM(I)
      CA(I)=150.0*CA(I)/DN
      CM(I)=1.5*CA(I)
18    CONTINUE
      QSUM=0.0
      DO 25 I=2,NODES
      LOCNOD=LOC(I)
      QF(I)=CM(LOCNOD)
      QSUM=QSUM+QF(I)
25    CONTINUE
     QF(1)=-(QSUM+100.)
     WRITE(IWRITE,100) YEAR,R
100   FORMAT(/10X'YEAR ='F10.1,5X'GROWTH RATE ='F10.4/)
      RETURN
      END
```

```
      SUBROUTINE COST(INODES,ITUNL,IUP,IDOWN,FLGTH,H,HWC,EXIST,KESDIA,
     1FACTOR)
C     CAPACITY EXPANSION FOR NEW YORK CITY BY DYNAMIC PROGRAMMING
C     WILLIAM MADDAUS    ROOM 1-371   MIT     SPRING,1969
C     THIS SUBROUTINE CALLS THE OPTIMIZATION ROUTINE AND CONVERTS
C     THE TUNNEL DIAMETER COST TO A NET PRESENT COST
      DIMENSION IUP(50),IDOWN(50),ITUNL(50),FLGTH(50),H(50),
     1 TCOST(30),DIANEW(50),EXIST(50),INODES(50),DIAUSE(50),KESDIA(50)
      COMMON/DYNAMC/I,J,K,N,DELT,DR,G,HL,ALF85,NODES,NTUNL,S(10,10),
     1D(10,10,10),R(10,10,10),Q(10,10,10),F(10,10),ISTATE(10,10),
     2DESIGN(30,10,10),EXCON(30,10),DIAS(30,10,10),QF(50),IWRITE
      IREAD=5
      IF(I .EQ. 1) GO TO 3
      DO 2 L=1,NTUNL
2     EXIST(L)=EXCON(L,J-K+1)
      WRITE(IWRITE,301)I,J,K
301   FORMAT(3I10)
      WRITE(IWRITE,200) (EXIST(L),L=1,NTUNL)
200   FORMAT( 10F11.2)
3     CALL NCOST(NODES,NTUNL,INODES,ITUNL,IUP,IDOWN,FLGTH,EXIST,
     1QF,H,HWC,TC,DIANEW,DIAUSE,KESDIA,FACTOR)
      WRITE(IWRITE,302)TC
302   FORMAT(' TCOST=',F11.1)
      WRITE(IWRITE,201)
201   FORMAT(' *****EXISTING DIAMETERS*****')
      WRITE(IWRITE,200) (EXIST(L),L=1,NTUNL)
      WRITE(IWRITE,204)
204   FORMAT(' *****USED DIAMETER*****')
      WRITE(IWRITE,200)  (DIAUSE(L),L=1,NTUNL)
      WRITE(IWRITE,202)
202   FORMAT(' *****NEW DIAMETER*****')
      WRITE(IWRITE,200)  (DIANEW(L),L=1,NTUNL)
      WRITE(IWRITE,203)
203   FORMAT('*****DEMAND*****')
      WRITE(IWRITE,200) (QF(L),L=1,NODES)
      TCOST(K)=TC
      DO 5 L=1,NTUNL
5     DIAS(L,J,K)=DIANEW(L)
      ALF75=43.5
      IF(I .GT. 1) GO TO 6
      XINT=(J-1)*DELT
      XPREV=0.0
      COSCHG=1.0
      R(I,J,K)=TCOST(K)
      DO 8 L=1,NTUNL
8     DESIGN(L,J,I)=DIANEW(L)
      DO 9 L=1,NTUNL
9     EXCON(L,J)=EXIST(L)
      ISTATE(J,1)=J
      GO TO 19
6     XINT=(K-1)*DELT
7     XPREV=(J-1)*DELT-XINT
C     COST INCREASE OR DECREASE IN 1982
      IF(XPREV .GT. 7.0) GO TO 10
      COSCHG=1.0
```

```
      GO TO 15
10    COSCHG=ALF85/ALF75
15    TCOST(K)=TCOST(K)*COSCHG
C     NET PRESENT COST
18    R(I,J,K)=TCOST(K)/((1+DR)**XPREV)
19    WRITE(IWRITE,100) K,XINT,XPREV,TCOST(K),R(I,J,K)
100   FORMAT(1X,'COST',I10,4(5X,F10.1))
      RETURN
      END
```

```
      SUBROUTINE NCOST(NODES,NTUNL,INODES,ITUNL,IUP,IDOWN,FLGTH,
     1EXIST,DEMND,HEAD,HWC,TCOST,DIANEW,DIAUSE,KESDIA,FACTOR)
      DIMENSION INODES(50),ITUNL(50),IUP(50),IDOWN(50),FLGTH(50),
     1EXIST(50),DEMND(50),HEAD(50),DIA(100),A(15000),KESDIA(50),
     2DIANEW(50),SP(100),FLOW(50),DIAUSE(50),ISS(10)
      DIMENSION HL(50),FK(50),ALPHP(50),KCODIA(50),B(50)
      NMAX=15000
      IWRITE=8
C  COUNT THE NUMBER OF EXISTING PIPES
      NEP=0
      DO 15 I=1,NTUNL
      IF(EXIST(I)-.01)  15,15,16
 16   NEP=NEP+1
 15   CONTINUE
      NSUPY=0
      DO 500 I=1,NODES
      IF(DEMND(I))501,502,502
501   NSUPY=NSUPY+1
      ISS(I+1)=1
      GO TO 500
502   ISS(I+1)=0
500   CONTINUE
C     WRITE(6,55)
55    FORMAT(1X,' INPUT INDICATOR, 1=PRINT OUT NODES ,NTUNL,NEP, 0=NOT')
C     READ(5,2) IPRIN1
2     FORMAT(I1)
C     IF(IPRIN1) 23,23,52
      WRITE(IWRITE,6001) NTUNL,NEP,NODES,FACTOR
 6001 FORMAT (///10X,24HTOTAL NUMBER OF PIPES = ,I3/
     1           20X,27HNUMBER OF EXISTING PIPES = ,I3/
     2           30X,18HNUMBER OF NODES = ,I3,
     3           40X,'FACTOR=',F10.5)
23    DO 11 I=1,NTUNL
11    HL(I)=0.
      DO 5 I=1,NTUNL
      II=0
      JJ=0
      DO 6 J=1,NODES
      IF(IUP(I).EQ.INODES(J)) II=J
      IF(IDOWN(I).EQ.INODES(J)) JJ=J
 6    CONTINUE
      IF(II) 5,5,7
7     IF(JJ)5,5,9
9     HL(I)=HEAD(II)-HEAD(JJ)
5     CONTINUE
C      WRITE(6,56)
56    FORMAT(1X,'INPUT INDICATOR, 1=PRINT NODES AND PIPES CONDITIONS,
     1 0=NOT')
C     READ(5,2) IPRIN2
C     IF(IPRIN2) 100,100,53
      WRITE(IWRITE,6002)
6002  FORMAT(///12X,4HNODE,17X,8HEXISTING,5X,'HEAD LOSS',/1X,4HPIPE,
     14X,5HBEGIN,2X,3HEND,4X,6HLENGTH,4X,8HDIAMETER)
      DO 19 I=1,NTUNL
19    WRITE(IWRITE,6003) ITUNL(I),IUP(I),IDOWN(I),FLGTH(I),EXIST(I),HL(I
```

```
     1)
 6003 FORMAT(I4,6X,I3,3X,I3,3X,F7.0,6X,F4.0,6X,   F7.1)
C     WRITE(IWRITE,6008)
6008  FORMAT(//1X,'NODE',10X,'DEMAND(MPD)')
C     DO 25 I=1,NODES
C     WRITE(IWRITE,6004) INODES(I),DEMND(I)
6004  FORMAT(I4,10X,F10.0)
100   DO 30 I=1,NTUNL
30    FK(I)=.00062*HWC*FLGTH(I)**(-0.54)
      M=NODES+NEP+1
      N=NMAX/M
      CALL ORGLP(M,N,A,FK,EXIST,DEMND,INODES,IUP,IDOWN,NTUNL,NODES,B,
     1HL,NEP,JMAX,NSUPY,ISS,FACTOR)
      ALPHA=1.1
      ITEND=10
      ITET=1
131   IF(ITET.GT.ITEND) GO TO 132
      DO 21 I=1,NTUNL
      KKK=KESDIA(I)
      GO TO (121,122,123,124,125,126), KKK
121   ALPHP(I)=3.72E-03
      GO TO 21
122   ALPHP(I)=9.75E-04
      GO TO 21
123   ALPHP(I)=4.84E-04
      GO TO 21
124   ALPHP(I)=2.41E-04
      GO TO 21
125   ALPHP(I)=2.16E-04
      GO TO 21
126   ALPHP(I)=1.58E-04
21    CONTINUE
      DO 129 I=1,NTUNL
      JJ=M*(I-1)+1
      A(JJ)=ALPHP(I)*FLGTH(I)
C     WRITE(IWRITE,601) JJ,A(JJ)
 601    FORMAT(1X,'A( ',I5,')=',F11.2)
129   CONTINUE
      IF(NSUPY) 503,503,504
504   DO 505 I=1,NSUPY
      JJ=JJ+M
      A(JJ)=0.
C      WRITE(IWRITE,601)JJ,A(JJ)
505   CONTINUE
503   IF(NEP) 250,250,251
251   JJJ=M*(NTUNL+NEP+NSUPY-1)+1
      DO 253 K=1,NEP
      JJ=JJ+M
      A(JJ)=0.
C     WRITE(IWRITE,601) JJ,A(JJ)
      JJJ=JJJ+M
      A(JJJ)=0.
C     WRITE(IWRITE,601)JJJ,A(JJJ)
253   CONTINUE
250   CONTINUE
```

```
      DO 254 I=1,JMAX
254   DIA(I)=0.
      CALL LPROG(M,M,JMAX,A,B,SP,DIA,OBJ)
      DO 133 I=1,NTUNL
      IF(DIA(I)-(60.**2.63)) 181,181,182
181   KCODIA(I)=1
      GO TO 133
182   IF(DIA(I)-(120.**2.63)) 183,183,184
183   KCODIA(I)=2
      GO TO 133
184   IF(DIA(I)-(180.**2.63)) 185,185,186
185     KCODIA(I)=3
      GO TO 133
186   IF(DIA(I)-(240.**2.63)) 187,187,188
187   KCODIA(I)=4
      GO TO 133
188   IF(DIA(I)-(300.**2.63)) 189,189,190
189   KCODIA(I)=5
      GO TO 133
190   KCODIA(I)=6
133   CONTINUE
      ICONT=0
      DO 127 I=1,NTUNL
      IF(KESDIA(I).NE.KCODIA(I)) ICONT=ICONT+1
127   CONTINUE
212   FORMAT(10F11.2)
      IF(ICONT.EQ.0) GO TO 132
      DO 128 I=1,NTUNL
      KESDIA(I)=KCODIA(I)
128   CONTINUE
      ITET=ITET+1
      GO TO 131
C     WRITE(IWRITE,134)ITET
134   FORMAT(1X,'ITERATIONS DONE ON LPROG=',I10)
C     WRITE(IWRITE,212) (DIA(I),I=1,JMAX)
132   KK=0
      DO 50 I=1,NTUNL
103   DIANEW(I)=(DIA(I)*FACTOR)**(1./2.63)
      IF(EXIST(I)-.01) 255,255,256
256   KK=KK+1
      DIAUSE(I)=(DIA(NTUNL+NSUPY+KK)*FACTOR)**(1./2.63)
      GO TO 50
255   DIAUSE(I)=0.
C     IF(HL(I)) 101,102,101
C     FLOW(I)=(FK(I)*(DIANEW(I)**2.63+DIAUSE(I)**2.63)*
C    1ABS(HL(I))**.54*HL(I)/(ABS(HL(I))*100.))*100.
C     GO TO 50
C     FLOW(I)=0.
50    CONTINUE
      TCOST=0.
      DO 35 I=1,NTUNL
      IF(DIANEW(I).LE.0.) GO TO 35
36    TCOST=TCOST+FLGTH(I)*1.1*DIANEW(I)**1.24
35    CONTINUE
      TCOST=TCOST/1000.
      RETURN
      END
```

```
      SUBROUTINE ORGLP (M,N,A,FK,EXD,Q,IN,JN,KN,NP,NN,B,HL,NEP,JMAX,
     1NSUPY,ISS,FACTOR)
      DIMENSION A(M,N),FK(50),EXD(50),FL(50),Q(50),IN(50),JN(50),
     1KN(50),B(50),HL(50),Z(50),ISS(10)
      B(1)=0.
      IWRITE=8
      DO 10 K=1,NN
10    B(K+1)=ABS(Q(K))
      J=1
      DO 30 K=1,NN
      J=J+1
      Z(J)=0.
      IF(Q(K)) 1,1,2
1       SIG=1.
      GO TO 3
2     SIG=-1.
3     DO 25 I=1,NP
      IF(HL(I)) 101,22,101
101    IF(JN(I)-IN(K)) 20,21,20
21    A(J,I)=SIG*FACTOR*FK(I)*ABS(HL(I))**.54*HL(I)/ABS(HL(I))
      GO TO 25
20    IF(KN(I)-IN(K)) 22,23,22
23    A(J,I)=-SIG*FACTOR*FK(I)*ABS(HL(I))**.54*HL(I)/ABS(HL(I))
      GO TO 25
22    A(J,I)=0.
25    Z(J)=Z(J)+A(J,I)**2
      Z(J)=SQRT(Z(J))
      IF(ABS(Z(J))-.00001) 26,26,30
26    WRITE(IWRITE,27) K
27    FORMAT(1X,'NODE',I5,'SHOULD BE IGNORED')
30    CONTINUE
      IF(NSUPY)200,200,201
201   DO 202 I=1,NSUPY
      KK=NP+I
      DO 203 KKK=2,J
      IF(ISS(KKK)) 204,204,205
205   A(KKK,KK)=1.
      ISS(KKK)=0.
      KKKK=KKK+1
      DO 206 JJ=KKKK,J
206   A(JJ,KK)=0.
      GO TO 202
204   A(KKK,KK)=0.
203   CONTINUE
202   CONTINUE
C   CONSIDERATION OF EXISTING PIPES
200   LL=0
      DO 50 I=1,NP
      IF(EXD(I)-.01) 50,50,51
51    J=J+1
      B(J)=EXD(I)**2.63/FACTOR
      LL=LL+1
      NNP=NP+LL+NSUPY
      DO 52 K=1,NNP
52    A(J,K)=0.
```

```
      DO 53 JJ=2,J
53    A(JJ,NNP)=A(JJ,I)
      A(J,NNP)=1.
50    CONTINUE
      JMAX=NP+2*NEP+NSUPY
C   ADD SLACK VARIABLES TO CORRESPONDING RXISTING PIPES
      IF(NEP) 65,65,81
81     DO 60 K=1,NEP
      I=NP+NEP+K+NSUPY
      II=NN+1+K
      DO 70 KK=2,J
70    A(KK,I)=0.
      A(II,I)=1.
60    CONTINUE
C     WRITE(6,63)
63    FORMAT(1X,'INPUT INDICATOR,1=PRINT OUT A MATRIX,0=NOT')
C     READ(5,64) INDICA
64    FORMAT(I1)
C     IF(INDICA) 65,65,66
C     WRITE(IWRITE,6001)
6001  FORMAT('  COEFFICIENT MATRIX')
C     DO 71 I=2,M
C     WRITE(IWRITE,6002) I,(A(I,K),K=1,JMAX)
6002  FORMAT(/1X,4HROW ,I2,1X,10F12.10/(8X,10F12.10))
      WRITE(IWRITE,6003)
6003  FORMAT(' CONSTRAINT MATRIX')
      WRITE(IWRITE,6004) (B(I),I=1,M)
6004  FORMAT(8X,10F11.2)
65    RETURN
      END
```

```
      SUBROUTINE LPROG(ME,M,N,A,B,Z,DIA,OBJ)
      DIMENSION A(1),B(1),Z(1),DIA(1),INFIX(8),TOL(4),E(3000),
     1          KOUT(7),ERR(8),JH(100),X(100),P(100),Y(100),KB(100)
      IWRITE=8
      INFIX(1) = 4
      INFIX(2) = N
      INFIX(3) = ME
      INFIX(4) = M
      INFIX(5) = 2
      INFIX(6) = 1
      INFIX(7) = 100
      INFIX(8) = 0
      TOL(1)=10.**(-4)
      TOL(2)=10.**(-4)
      TOL(3) = -10.**(-3)
      TOL(4) = 10.**(-10)
C     WRITE(IWRITE,92)
92    FORMAT(' OUTPUT FROM LPROG')
C     WRITE(IWRITE,91) N,ME,M
91    FORMAT(' N=',I10,'  ME=',I10,'  M=',I10)
      PRM = 0.
      B(1)=0.
      CALL SIMPLX (INFIX,A,B,TOL,PRM,KOUT,ERR,JH,X,P,Y,KB,E)
      DO 1 I=1,N
    1 Z(I) = 0.
      DO 2 I=1,N
      J = KB(I)
      IF (J) 2,2,3
    3 Z(I) = X(J)
    2 CONTINUE
      DO 5 I=1,N
    5 DIA(I)=Z(I)
      OBJ = Y(1)
11     WRITE(IWRITE,6500) (KOUT(I),I=1,7)
6500  FORMAT(7I10)
6502  FORMAT(4E12.5)
      RETURN
      END
```

```
      SUBROUTINE SELECT
C     CAPACITY EXPANSION FOR NEW YORK CITY BY DYNAMIC PROGRAMMING
C     WILLIAM MADDAUS    ROOM 1-371   MIT     SPRING,1969
C     THIS SUBROUTINE CHOOSES THE MINIMUM COST DECISION GIVEN THE STATE
      COMMON/DYNAMC/I,J,K,N,DELT,DR,G,HL,ALF85,NODES,NTUNL,S(10,10),
     1D(10,10,10),R(10,10,10),Q(10,10,10),F(10,10),ISTATE(10,10),
     2DESIGN(30,10,10),EXCON(30,10),DIAS(30,10,10),QF(50),IWRITE
      IREAD=5
      QMIN=1.E 10
      JJ=J
      DO 10 K=1,J
      Q(I,J,K)=R(I,J,K)+F(I-1,JJ)
C     SEARCH FOR MINIMUM COST
      IF(Q(I,J,K) .GT. QMIN) GO TO 9
      QMIN=Q(I,J,K)
      KOPT=K
      JJOPT=JJ
9     JJ=JJ-1
10    CONTINUE
      F(I,J)=QMIN
      DO 15 L=1,NTUNL
15    DESIGN(L,J,I)=DIAS(L,J,KOPT)
      WRITE(IWRITE,100) (DESIGN(L,J,I),L=1,NTUNL)
100   FORMAT(10F10.1)
       XINT=(KOPT-1)*DELT
       XPREV=(J-1)*DELT-XINT
      ISTATE(J,I)=JJOPT
      WRITE(IWRITE,105)KOPT,XINT,JJOPT,XPREV,F(I,J),DESIGN(L,J,I)
105   FORMAT(1X,'SELECT'I8,F8.1,I8,3E8.1)
      RETURN
      END
```

```
      SUBROUTINE NEW
C     WILLIAM MADDAUS    ROOM 1-371  MIT     SPRING,1969
C     CAPACITY EXPANSION FOR NEW YORK CITY BY DYNAMIC PROGRAMMING
C     THIS SUBROUTINE UPDATES THE EXISTING SYSTEM BY USING
C     HYDRAULICALLY EQUIVALENT PIPES FOR TWO PARALLEL PIPES
      COMMON/DYNAMC/I,J,K,N,DELT,DR,G,HL,ALF85,NODES,NTUNL,S(10,10),
     1D(10,10,10),R(10,10,10),Q(10,10,10),F(10,10),ISTATE(10,10),
     2DESIGN(30,10,10),EXCON(30,10),DIAS(30,10,10),QF(50),IWRITE
      IREAD=5
      P=2.63
      N1=N-1
      WRITE(IWRITE,90)
90    FORMAT(10X,'NEW EQUIVALENT PIPES')
      DO 10 L=1,NTUNL
      DO 10 JK=1,N1
      J=N-JK+1
      JJ=ISTATE(J,I)
      DEXIST=EXCON(L,JJ)
      DNEW=DESIGN(L,J,I)
C     DIAMETER OF THE HYDRAULICALLY EQUIVALENT PIPE
      DEQUIV=(DEXIST**P+DNEW**P)**(1.0/P)
      EXCON(L,J)=DEQUIV
100   FORMAT(10F10.1)
10    CONTINUE
      WRITE(IWRITE,100)(EXCON(L,J),L=1,NTUNL)
      RETURN
      END
```

APPENDIX C

AN ALGORITHM FOR THE OPTIMAL ALLOCATION

OF PRESSURE LOSS ALONG A PROPOSED PIPELINE

An Algorithm for the Optimal Allocation Of Pressure Loss Along a Proposed Pipeline

A pipeline is to be constructed from A to B. The hydraulic grade line is fixed at A and at B as well.

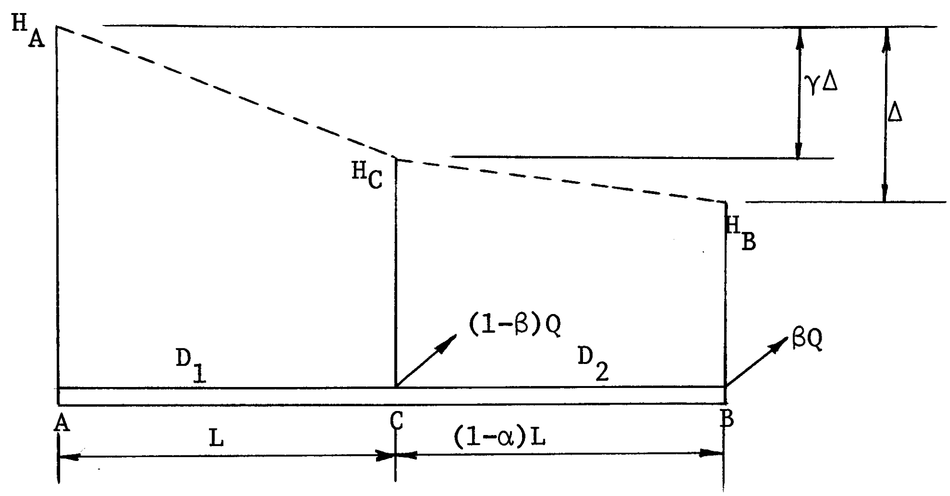


Figure C-1

The purpose of the pipeline is to deliver the total quantity of water, Q. Part of the total flow is required at B and part at C, an intermediate location between A and B. The amount delivered to B is βQ, where $0 \leq \beta \leq 1$. The remainder is delivered to C. Location C is a distance αL from A, where $0 \leq \alpha \leq 1$.

The total head loss between A and B is

$$\Delta = H_A - H_B$$

and this is fixed at some given value. Ultimately, we wish to determine the diameters D_1 and D_2 which are optimal in the sense that the total cost of the pipeline from A to B is a minimum. This simple problem is referred to below as the "basic problem".

Of greater practical interest is the more general problem where there are many intermediate points between the extremities of the pipeline. This is referred to below as the "serial problem". In the general serial problem, there are many more decision variables than there are in the basic problem.

It appears that the serial problem is equivalent to a cascade of basic problems. Both problems involve the allocation of the total head loss across the system to the branches within the system.

If we knew how to do this for the basic problem, where there are only two branches, we could solve the serial problem as a simultaneous set of solutions to the basic problem.

To find the optimal allocation of head loss in the basic problem, we proceed as follows. For any one branch, the pipe diameter needed to transport a flow rate q at a total pressure loss h over a distance ℓ is:

$$D = K\, q^{r}\, h^{p}\, \ell^{s} \qquad \text{(C-1)}$$

where, for D and ℓ in feet and q in mgd,

$$K = 1.264 C_{HW}^{-.381}$$

$$s = .205$$

$$r = .381$$

$$p = -.205$$

(C_{HW} is the Hazen Willismas pipe coefficient).

The total cost of a pipe is

$$C_D = cD^{m}\ell \qquad \text{(C-2)}$$

where

$$c \approx 1.89 \text{ for pipes and } 5.8 \text{ for tunnels}$$

$$m \approx 1.24$$

so that the pipe cost, as a function of head loss is

$$C_D = ck^m\, q^{rm}\, h^{pm}\, \ell^{1+sm} = (c)(1.825)\, C_{HW}^{-.473}\, q^{.473}\, h^{-.255}\, \ell^{1.255} \tag{C-3}$$

The total cost of the system from A to B is

$$C_T = (c)(1.825)(C_{HW}^{-.473})\, [(\alpha L)^{1.255}\, Q^{.473}(H_A - H_C)^{-.255} + ((1-\alpha)L)^{1.255}\, (\beta Q)^{.473}\, (H_C - H_B)^{-.255}] \tag{C-4}$$

Let γ be the proportion of Δ to be allocated between locations A and C

$$H_A - H_C = \gamma\Delta$$

so that

$$H_C - H_B = (1-\gamma)\Delta$$

Then

$$C_T = e[\xi_1\, \gamma^{-.255} + \xi_2 (1-\gamma)^{-.255}] \tag{C-5}$$

in which

$$e = (c)(1.825)(C_{HW}^{-.473})\, L^{1.255}\, Q^{.473}\, \Delta^{-.255} \tag{C-6}$$

$$\xi_1 = \alpha^{1.255} \tag{C-7}$$

$$\xi_2 = (1-\alpha)^{1.255}\, \beta^{.473} \tag{C-8}$$

The minimum cost obtains from

$$\frac{dC_T}{d\gamma} = 0$$

where

$$\frac{dC_T}{d\gamma} = e[-.255\xi_1 \gamma^{-1.255} + .255\xi_2(1-\gamma)^{-1.255}]$$

thus

$$\left(\frac{1-\gamma}{\gamma}\right)^{1.255} = \frac{\xi_2}{\xi_1} \quad \text{(C-9)}$$

so that

$$\gamma = \frac{1}{1+\left(\frac{\xi_2}{\xi_1}\right)^{0.797}} \quad \text{(C-10)}$$

Note that ξ_1 and ξ_2 are functions only of α and β. Since γ is a surrogate for the allocation of the total pressure loss from A to B, it follows that the optimal allocation of this head loss is a function only of the relative location of the intermediate point and the relative amount of the initial flow transported all the way from A to B. In terms of α and β:

$$\gamma = \frac{1}{1+\left(\frac{1-\alpha}{\alpha}\right)\beta^{0.377}} \quad \text{(C-11)}$$

Example:

Let $\alpha = .5$, $\beta = .5$

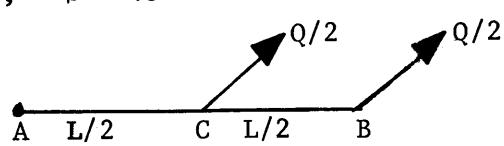


Figure C-2

Then

$$\xi_1 = \alpha^{1.255} = .418$$

$$\xi_2 = (1-\alpha)^{1.255}\beta^{.473} = .301$$

$$C_T = e[.418\gamma^{-.255} + .301(1-\gamma)^{-.255}]$$

γ	$\gamma^{-.255}$	$(1-\alpha)$	$(1-\gamma)^{-.255}$	$(.418)(\gamma^{-.255})$	$(.301)(1-\gamma)^{0.255}$	$C_{T/e}$
.2	1.507	.8	1.059	.629	.319	.948
.4	1.263	.6	1.139	.528	.343	.871
.6	1.139	.4	1.263	.477	.380	.857
.8	1.059	.2	1.507	.443	.453	.896

$$\gamma_{optimal} = \frac{1}{1+(0.5)^{0.377}} = 0.565$$

A Second Example

The next step is to apply these results to a more general case where there are two intermediate withdrawal locations.

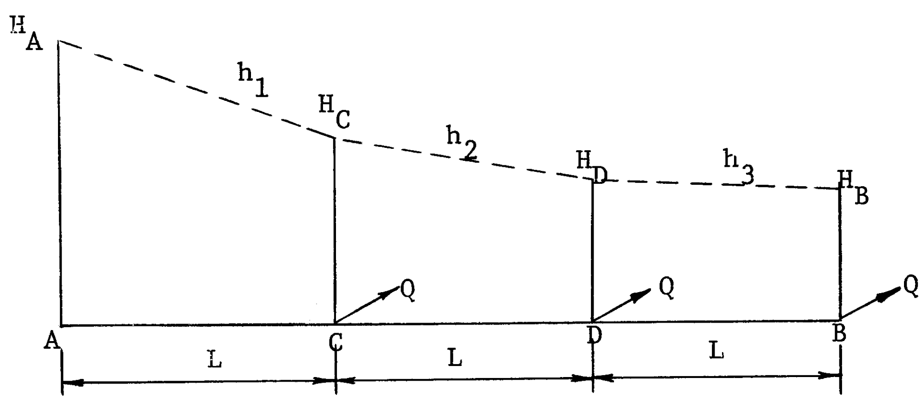


Figure C-3

As before, the total head loss between A and B is Δ. Assume, for example, that C and D are equally spaced between A and B and that the withdrawals are as shown in Figure C-3.

Let the unknown head losses between these locations be h_1, h_2 and h_3 as shown in the figure above. Applying the basic algorithm to the interval from A to D gives

$$\gamma_1 = \frac{h_1}{h_1 + h_2} = 0.565$$

Similarly, for the interval from C to B,

$$\gamma_2 = \frac{h_2}{h_2 + h_3} = 0.565$$

With the requirement that

$$h_1 + h_2 + h_3 = \Delta$$

we have three simultaneous equations.

The solution is:

$$h_1 = 0.422\Delta$$

$$h_2 = 0.326\Delta$$

$$h_3 = 0.252\Delta$$

The Serial Problem

To apply the basic algorithm, to the serial problem, proceed as follows. Let the total number of demands be N so that there are N concatenated branches, and N decision variables; and we need N simultaneous equations. The first equation, as above, derives from the fact that the sum of the decision variables (i.e. the h_i) must equal the total pressure loss. The other N-1 equations result from applying the basic algorithm to every contiguous pair of branches. For each pair, a value of γ must be computed from Equation (C-11). The simultaneous equations are particularly easy to solve by substitution because most of them involve only two unknowns. The first equation links the other equations together.

Sensitivity Analysis of Total Cost

Equations C-5 to C-8 are useful, not only for computing the total cost of the solution to the basic problem, but also for estimating the effect on total cost of small changes in the various quantities that have an impact on total cost.

It was initially assumed that L, Q, C_{HW} and Δ are given as fixed quantities. As a matter of post-optimal interest, we may want to know how changes in these quantities affect the total cost. In particular, what change in each of these will produce a one per cent change in the total cost. This is a classic problem because the total cost equation is of the simple form

$$y = ax^{b} \tag{C-12}$$

where x is the particular variable of interest to the sensitivity analysis, and y is the total cost. We need to differentiate Equation C-12

$$\frac{dy}{dx} = abx^{b-1} \tag{C-13}$$

so the change in y for a small change in x is approximately

$$\Delta y = abx^{b-1}\,\Delta x \tag{C-14}$$

The relative change in y is given by $\Delta y/y$, which, from Equations C-14 and C-12, give

$$\frac{\Delta y}{y} = \frac{abx^{b-1}}{ax^{b}}\,\Delta x = b\,\frac{\Delta x}{x} \tag{C-15}$$

It follows that the percent change in x to produce a one percent change in y is approximately equal to 1/b. (C-16)

Economists have defined a specific measure of sensitivity that they apply to analysis of price changes caused by factors which affect prices. That measure is called "elasticity" and is defined, in our notation, as the ratio

$$\frac{\frac{\Delta y}{y}}{\frac{\Delta x}{x}} = E \qquad \text{(C-17)}$$

The value of E is a measure of the relative change in the factor, x needed to produce a unit relative change in price, y. In this case, y denotes total cost. To follow the economist's lead, adopt the convention that the total cost is inelastic, or relatively insensitive, to change if $|E|$ is less than unity; conversely, if greater.

In our case, we have the simple relation

$$E = \frac{\frac{\Delta y}{y}}{\frac{\Delta x}{x}} = b \qquad \text{(C-18)}$$

so the exponent of a term in the total cost equation gives us a direct measure of the elasticity of the total cost to changes in that term.

We can summarize the total cost sensitivity analysis in the following table.

Factor	Elasticity (E)	Sensitivity	Per Cent Change to Produce One Percent Change in Total Cost
$\Delta = H_A - H_B$	-.255	Inelastic	-3.9
Q	.473	Inelastic	2.1
L	1.255	Elastic	0.8
C_{HW}	-.473	Inelastic	-2.1
D*	1.24	Elastic	0.8

Table C-1

(*Note, the diameter, D, is not a factor which directly is controllable. The first 4 factors jointly constrain the feasible diameter so the value of D results from the decision making process. The fact that the cost is sensitive to D is why we examine the decision making process closely.)

Sensitivity Analysis of Head Loss Allocation

The parameter γ denotes the proportion of the total head loss from A to B which is allocated to the branch from A to C. Therefore, this parameter is a good surrogate for the term "head loss allocation". We are interested in how this term is affected by other factors in the problem, and we are also interested in how small changes in γ affect total cost.

Consider, first, the factors which may influence γ. Inspection of Equation C-11 shows that only α and β are used to compute γ. Hence, we can conclude that the four terms Δ, Q, L and C_{HW} have no affect at all on γ! Next, consider the two factors α and β.

First, as before, compute the elasticity

$$E_\alpha = \frac{\frac{\Delta y}{y}}{\frac{\Delta\alpha}{\alpha}} \quad \text{and} \quad E_\beta = \frac{\frac{\Delta y}{y}}{\frac{\Delta\beta}{\beta}}$$

The algebra is more involved than before so the details are not presented here. After some manipulation we get

$$E_\alpha = \frac{\frac{\Delta\gamma}{\gamma}}{\frac{\Delta\alpha}{\alpha}} = \frac{\frac{\beta^{0.377}}{\alpha}}{1+\left(\frac{1-\alpha}{\alpha}\right)\beta^{0.377}} \tag{C-19}$$

and

$$E_\beta = \frac{\frac{\Delta\gamma}{\gamma}}{\frac{\Delta\beta}{\beta}} = \frac{-0.377\left(\frac{1-\alpha}{\alpha}\right)\beta^{0.377}}{1+\left(\frac{1-\alpha}{\alpha}\right)\beta^{0.377}} \tag{C-20}$$

In this case, the elasticities depend on the values α and β. Suppose we have $\alpha = 0.5$ and $\beta = 0.5$ as in the first example. Then the elasticities we obtain are summarized below in Table C-2.

Factor	Elasticity	Sensitivity	Percent change in factor needed to produce one percent change in γ
α	0.87	Inelastic	1.15
β	-0.164	Inelastic	-6.1

Table C-2

These results show that the head loss allocated to the first branch is relatively much more sensitive to the location of the intermediate node than to the relative distribution of withdrawal between the two nodes.

Increasing α, which corresponds to moving point C closer to point B, results in a small increase in γ, which would increase the head loss from A to C with a corresponding decrease from C to B. Increasing β, which corresponds to moving some of the water use from C to B, results in a small decrease in γ, which would decrease the head loss from A to C with a corresponding increase from C to B.

Also of interest is the sensitivity of the total cost to changes in γ. Because the value of γ is derived from setting $\frac{dC_T}{d\gamma}$ equal to zero, it follows that the total cost is insensitive to small variations in γ in the vicinity of the optimum. The marginal elasticity of C_T with respect to γ is zero!

Implications for Distribution Networks

The method described here for the serial problem can be adapted to non-looping networks with a single source of supply. A non-looping network is shaped like a tree. In this kind of network, the number of branches is one less than the number of nodes. The supply occurs at one of the nodes so there are as many branches as demand nodes. As before, it is possible to set up a set of simultaneous equations to compute the allocation of pressure losses by recursive application of an expanded form of the basic algorithm. The basic algorithm is not sufficient to determine the pressure allocations where more than two branches meet at the same node. The new system of simultaneous equations contains non-linear as well as linear equations so a special solution technique is also needed.

Consider the case where there are three branches.

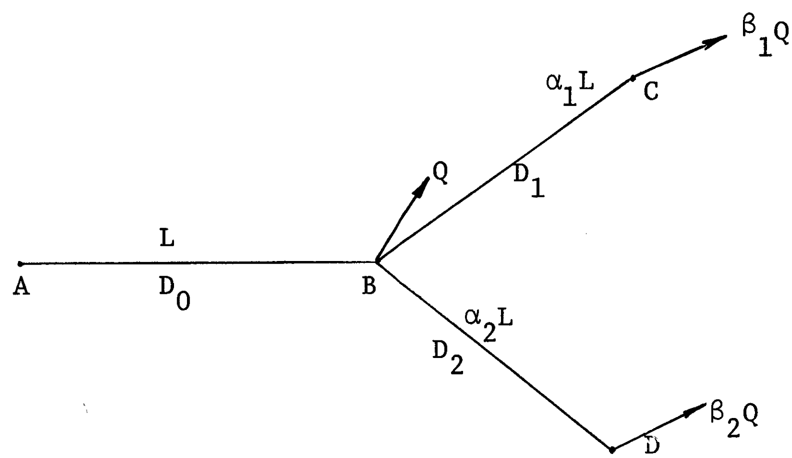


Figure C-4

The supply is at A. There are demands at B, C and D. (The notation used here is slightly different from the previous notation.) The head loss from A to C and from A to D is fixed. Let

$$\Delta_1 = H_A - H_C$$

$$\Delta_2 = H_A - H_D$$

We want to find the rule for computing

$$\Delta_0 = H_A - H_B$$

so that the total cost is a minimum. The total cost, according to Equation C-3 is

$$C_T = 1.825cC_{HW}^{-.473}\, Q^{.473}\, L^{1.255}\, \{(1+\beta_1+\beta_2)^{.473}\, \Delta_0^{-.255} + \beta_1^{.473}\, \alpha_1^{1.255}\, (\Delta_1-\Delta_0)^{-.255} + \beta_2^{.473}\, \alpha_2^{1.255} (\Delta_2-\Delta_0)^{-.255}\} \tag{C-21}$$

The optimum solution follows from

$$\frac{dC_T}{d\Delta_0} = 0 \tag{C-22}$$

This leads, after some algebraic manipulation, to the implicit equation for Δ_0

$$(1+\beta_1+\beta_2)^{.473}\, \Delta_0^{-1.255} - \beta_1^{.473}\, \alpha_1^{1.255}\, (\Delta_1-\Delta_0)^{-1.255} - \beta_2^{.473}\, \alpha_2^{1.255}\, (\Delta_2-\Delta_0)^{-1.255} = 0 \tag{C-23}$$

The algorithm for allocating the pressure losses throughout a non-looping network must account for the non-linear equations. There is one non-linear equation for each junction node. There is one linear equation for each node that is not a junction. These linear equations involve only the pressure losses in two adjacent conduits. The set of linear equations for the nodes between two junction nodes can be used to solve for the head loss in the conduits adjacent to each junction as a function of the total head loss in the link between the junction nodes. In this way, most of the linear equations can be solved separately from the non-linear equations.

Upon substituting the previous results from the linear equations, the non-linear equations are transformed into a set of relations between total head losses in adjacent links. Originally, the non-linear equations related only the head losses in conduits adjacent to the junction node so the new equations are much more useful than the original equations.

Remaining to be solved are a system of linear and non-linear equations. We have one non-linear equation for each junction node. We have one linear equation for each extremity which specifies the total head loss through the system from the supply to the extremity of the system. The unknowns in these remaining equations are the link head losses. When these are determined, the problem reduces to a set of original serial problems.

An approach to solving this system of non-linear equations is as follows. Let X denote the unknown head losses. Let B denote the vector of right-hand-sides of the equations. Then the equations are equivalent to

$$AX = B$$

in which A is a matrix of coefficients, and some of these coefficients are functions of X. Assume an initial estimate of X exists. Call this X_0. Use the values of X_0 to compute the coefficients of A. Call this A_0. Since we must have

$$X = A^{-1} B$$

The algorithm is then

$$X_{i+1} = A_i^{-1} B$$

and we continue until X_{i+1} is close enough to X_i. The link head losses are then contained in X_{i+1}. These are the total losses between junction nodes and between junction nodes and boundary nodes. The problem of allocating the head loss between these nodes is the same as the original serial problem.