GIS Based Optimal Design of Sewer Networks and Pump Stations

by

Newland Agbenowosi

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Civil Engineering

APPROVED:

Dr. R. G. Greene, Co-chairman

Dr. G. V. Loganathan, Co-chairman

Dr. D. F. Kibler

November, 1995

Blacksburg, Virginia.

GIS BASED OPTIMAL DESIGN OF SEWER NETWORKS AND PUMP STATIONS

by

Newland Agbenowosi

Dr. R. G. Greene, Co-Chairman and Dr. G. V. Loganathan, Co-chairman

Department of Civil Engineering

Virginia Tech.

(ABSTRACT)

In the planning and design of sewer networks, most of the decisions are spatially dependent because of the right of way considerations and the desire to have flow by gravity. This research addresses the application of combined optimization-geographic information system (GIS) technology in the design process. The program developed for the design uses selected manhole locations to generate the candidate potential sewer networks. The design area is delineated into subwatersheds for determining the locations for lift stations when gravity flow is not possible. Flows from upstream subwatersheds are transported to the downstream subwatersheds via a force main.

The path and destination of each force main in the system is determined by applying the Dijkstra's shortest path algorithm to select the least cost path from a set of potential paths. This method seeks to minimize the total dynamic head. A modified length is used to represent the length of each link or force main segment. The modified length is the physical length of the link (representing the friction loss) plus an equivalent length (representing the static head). The least cost path for the force main is the path with the least total modified length.

The design approach is applied to two areas in the town of Blacksburg, Virginia. The resulting network and the force main paths are discussed.

Acknowledgments

My first and foremost thanks to the Lord Jesus Christ. For it is by His mercy, grace and protection I have come this far.

I wish to express my gratitude to Dr. R. G. Greene and Dr. G. V. Loganathan for being more than advisors and mentors to me. Their interest in the success of every step of this project and their perceptive questions have been a major factor in directing the course and outcome of the project.

Dr. Greene's support through his provision of unlimited access to his workstation and software and his availability to guide me through all software and hardware related problems is greatly appreciated. His expertise in GIS and experience with ARC/INFO have also been an invaluable resource.

Dr. Loganathan's keen insights and patience in helping me with any problem (especially the hydraulics/hydrology related) I brought to him has been an precious resource for which I am truly thankful. His careful approach in editing the work and the brilliant suggestions he offered are all greatly appreciated.

My special thanks to Dr. D. F. Kibler for his interest in this project, his efforts in reviewing the paper, and his willingness to serve on my thesis committee.

Thanks to my father, Patrick, and my mother, Patience for their support and funding my education. My thanks also to all my brothers and sisters who have always been a source of great support for me.

Thanks to Mr. Dave Vogelsong, Mr. Gary Crouch, and Mr. Howard Lusk for their help in acquiring needed information and data for the work.

Thanks to the faculty, staff and graduate students of the Hydrosystems Division for their support both academically and socially.

Finally but not the least, I wish to thank all my friends for their encouragement when times were tough.

Table Of Contents

List Of Illustrations

List Of Tables

Chapter 1 - Introduction

1.1 Background

Sanitary sewer systems are tree networks which are designed to carry wastewater from different sources such as residential, industrial, or commercial areas, to an outfall. The outfall may be a wastewater treatment plant or a pump station from which wastewater is pumped to an appropriate destination. The sewer network is composed of manholes (nodes) linked to one another by pipes (links). The outfall manhole is the point to which all the other manholes are drained. Except the outfall manhole, each of the manholes is drained by a pipe. Sewers are generally designed to flow by gravity. When flow by gravity is impossible due to the topography or other physical obstruction, the sewage has to be pumped to a point of higher elevation and then allowed to flow by gravity. Pumping of the wastewater requires the use of a special pipe, referred to as a force main, to transport the flow under pressure from the location of the pump to the point where it is discharged.

The design of sanitary sewer systems involves the use of maps and other sources of detailed information regarding the pipe layout for a system (Design, 1970). These include topographic details on natural and artificial features in the area as well as information on the subsoil, groundwater levels, and other information necessary to determine the course, depths of excavation, and slopes of the pipelines in the system. Developmental information, which refers to both present and projected population densities of the area and the classification or type of zoning of the area must also be obtained. This information indicates whether the areas to be designated as residential, commercial, or industrial (Design, 1970). Information on the street layout, various property lots and positions of buildings

and other features prohibited for the passage of sewer lines are also obtained. The political districting concerns and the need for annexation and to identify the areas in nearby jurisdictions that will be contributing to the system to be designed should be taken into account.

Design of a sanitary sewer system in a relatively flat terrain without a need for pumping involves the selection of the location of manholes, an appropriate layout, pipe diameters and pipe slopes that meet hydraulic regulatory standards at a minimal cost. For sewer systems in hilly terrain, the design becomes more complex as the design of the downstream subwatershed network layouts depends on the contributions from upstream subwatersheds.

1.2 Goal

This research utilizes the spatial analysis capabilities of geographic information system (GIS) for the detailed design of sewer systems including, pump station locations and the determination of the force main path to minimize force main pipe costs and cost of pumping. GIS is a computer system capable of assembling, storing , manipulating, and displaying geographically referenced information. The research is also an extension of the gravity sewer design program (GSDPM3) developed by Gray et al. (1992). Using the surface elevations of manhole locations, pipe lengths, estimated flows in pipes, and the connectivity of pipes provided by the user, GSDPM3 designs the sewer network yielding the pipe sizes, slopes, depths and volume of excavation, and the total cost of the design of the system.

The various tasks involved in the sewer design process are described in the next several chapters. Chapter 2 presents a review of several approaches to sewer network layout generation and pipe design and the use of GIS in the modeling of

sewer systems. Chapter 3 describes the methodology for the layout generation, subwatershed delineation, and pump station location.

Chapter 4 contains the procedure for the determination of the pumping destination and the optimal path for the force main. In chapter 5, the criteria used for the design of the wet wells and pump selection for each subwatershed are given.

Chapter 6 integrates all the components including the layout generation and subwatershed delineation, force main design, path determination, wet well design, pump selection, and the design of pipes. The method used to estimate the system costs is also described in chapter 6.

In chapter 7, the design approach is applied to two areas in the town of Blacksburg, Virginia. A summary of the research is presented in chapter 8 together with the conclusions and suggestions for further research. The limitations of the overall approach are also discussed in chapter 8.

Chapter 2 - Literature Review

2.1. Network Layout Generating Algorithms

Several computer algorithms have been developed by previous researchers to assist sewer design engineers in the selection of optimal or near optimal network for an area. Two parameters of concern in the sewer network design are the location of manholes and the order in which they are linked (the network layout).

Walters (1985) developed an algorithm which takes possible locations of each manhole in the system and designs an optimal layout. The algorithm varies the location of each manhole according to a grid of alternative locations for that manhole. The grid of alternative locations for each manhole is provided by the user. By considering the sewer network as a serial system, Walters (1985) applied a dynamic programming approach to generate an optimal network.

Dajani et al (1977), applied various mathematical programming algorithms such as separable programming, dynamic programming, and geometric programming to generate optimal network layouts for sewer systems design.

Tekeli and Belkaya (1986) have developed a hydraulic design algorithm and a layout generation algorithm for a sewer system design. The layout generation algorithm generates the sewer network layout while the hydraulic design algorithm designs the pipes in the network. In the layout generation algorithm, Tekeli and Belkaya assumed a constant pipe diameter for the system. Three criteria were applied in the determination of the network layout. For each criterion, a shortest path spanning tree (SPST) was generated using Floyd's Algorithm (Tekeli and Belkaya, 1986). The first criteria involved assigning all possible links with their horizontal length (HL) measure. In the second criterion, the inverse slope (IS) measure was used. Whereas the HL measure refers to the

horizontal distance between two directly linked manholes, the IS measure is a measure of the inverse of the slope between two directly linked manholes. The third criterion involved an estimated excavation volume (EX) based on an assumed minimum pipe diameter, a minimum slope and a minimum depth of cover. The EX measure in most cases produced the least cost network compared to the other two measures and manually generated networks. Tekeli and Belkaya however, found out that the EX approach was not very good when applied to very large networks though it performed better when these large networks are zoned or subdivided.

The work of Tekeli and Belkaya (1986) does not address the location of pump or lift stations. Also, the user has to interactively identify links that cannot be used as paths before the program is run.

Pierre and Guisset (1984) optimize storm sewer system by the introduction of fictitious discharges into the system and used a non-linear programming procedure to solve the resulting equations. The MINOS program is utilized in solving the non linear equations. The program is based on a reduced gradient method with a quasi-Newton approximation of the Hessian matrix.

Lui et. al. (1990) have developed a sewer design algorithm that seeks to determine pipe sizes, slopes and locations of pump stations using a known network layout. The algorithm is comprised of two submodels: the first one optimizes the pipes and pump station location and the second submodel seeks the pipes in which flows can be adjusted to decrease an overall objective function. The final solution is arrived at by a continuous coordination of the two submodels. Apart from specifying the network layout, the user has to identify the locations which are possible sites for the placement of pump stations.

Orth et al. (1984) require a user to provide the layout and the locations which are potential sites for lift station. Their approach uses both a dynamic programming and a branch and bound procedure for the design of an urban drainage network. The program decides whether to use the lift stations or to use gravity sewers by laying the pipes deeper in the ground. The decision is based on a cost function used to assess the cost of each lift station used.

Charalambous et al. (1990) optimize the sewer system design by determining the least possible burial depth of each link in the network. The requirements include a user specified layout, manhole locations and their street elevations. The program provides locations where lift stations are necessary. It also gives a total cost of the system design. Though the program indicates points where lift stations are required, there is no information on the destination where flow form each of the lift stations will be pumped.

2.2 GIS Application in Sewer Systems Design and Modeling

The geographic information systems technology, has created an opportunity for significant improvements in the design of sewer systems. This improvement is possible both in the final design product and the minimization of labor in the design process. Geographic information systems, capable of storing, retrieving and managing spatial database are ideally suited for topological network design involving sewer systems design.

Przybyla and Kiesler (1991) review how GIS can be used in modeling sewer systems. Przybyla and Kiesler combined the graphics capabilities of GIS packages which lack sewer system modeling capabilities with existing sewer system modeling packages which lack graphics capabilities. The result is a sewer

system modeling program (Lexington-Fayette Sewer Modeling System (LFSMS)) which uses an FMS/AC running with AutoCAD together with HYDRA and NEWSE, both sewer modeling programs, for the analysis of sewer systems.

Bakken et al. (1992) have demonstrated how the Indianapolis Water Company is implementing GIS technology in updating land maps as well as modeling the water distribution system in the Indianapolis/Marion County Metropolitan area using KYPIPE. The GIS is used for the system modeling as well as to determine the adequacy or inadequacy of the various pipes and to redesign the pipes accordingly. The GIS also enables the development of meter reader routes once newly proposed systems are added to the existing system.

Anderson and Associates Inc. (Crouch, 1995), an engineering consulting firm utilizes GIS in the determination of profiles and excavation volumes required in their sewer systems design. Their approach involves inputting the already generated sewer network layout and pipe connectivities into the ARC/INFO GIS which in turn calculates the various required excavation volumes between any two manholes (Crouch, 1995).

2.3 GIS in Path Determination

When pumping is necessary in the design of sewer systems, it becomes desirable to lay force main in such a manner as to minimize costs of the over all system. The cost which is dependent on the length of force main and path of force main can be minimized by choosing force main path by an optimization process. The Argonne National Laboratory is developing a GIS package for the Gas Research Institute (GRI) which will aid in planning of pipe line routes by combining a wide variety of geographical data for the area through which the pipe

is to pass (GIS Leads, 1993). For example, in order to lay a pipe from point A to B, the decision depends on what obstacles lie between points A and B. These may be rocky subterrain, lakes, or man-made feature such as population centers, farmland and roads. Also, environmental restrictions may make certain undeveloped land, such as sanctuaries for endangered species, prohibited for the pipeline. The GRI package is supposed to provide route alternatives for the designer who will then make a decision based on different costs of right-of-ways. Depending on costs, some of these obstacles may either be traversed or avoided by choosing the path of pipe to negotiate around them. The GRI expects the incorporation of GIS in the decision making process to provide the means of analyzing various options for pipe paths before a field trip is made (GIS Leads, 1993).

The algorithms currently used in optimal design of sewer networks lack an integrated approach to coordinate pump station location determination, wet well design, the optimal destination of wastewater from a pump station, and force main path determination. The method developed in this research provides a tool for a careful analysis of the topography and the effects of the presence of surface features to determine gravity sewer network layout. It also determines areas where pumping may be necessary. The inclusion of pump stations, wet wells, and force main allows not only for preliminary design but also an estimation of the overall system cost.

Chapter 3 - Watershed Delineation and Network Layout Selection

3.1 Introduction

This chapter addresses how GIS assists in the sewer network layout generation and the delineation of subwatersheds. The subwatersheds are generated when it is not feasible to use gravity sewer connections for the whole watershed. In such cases, pumps are used to transport the flow via a force main to a point were it can flow by gravity to the outfall. The need to use a minimum number of pumps is top priority.

3.2 Required Data

The information required for the design of sewer systems includes: the topography of the area, the desired locations of the manholes, estimated flows, and the locations of existing and proposed surface features. The topography may be obtained from a digital elevation model provided by the United States Geological Survey (USGS), or for better accuracy, topographic maps from the town engineer's office if available or topographical data from land surveys. Also needed is a map showing the location of various buildings and other surface and subsurface features that will restrict or prohibit the passage of a sewer line or force main. This layer is used by the program to help avoid designing sewer lines or force mains to pass through the locations of these features. For the purpose of this discussion, this layer is referred to as the prohibited areas layer. These prohibited areas may include building footprints, and other designated areas through which sewer lines may not pass. These are represented as polygons in the prohibited

areas layer. For a new development, a survey plat showing the boundaries of lots in the proposed subdivision would also indicate areas and right of ways reserved for location of sewers and other utilities. These are used for determining the paths to be taken by the sewer lines and the force main lines. Zoning maps and maps delineating property lots may be used as a guide for locating manholes and assigning the estimated flows to the various manholes.

3.3 Data Preprocessing

Apart from the topographic, prohibited areas, property lots and zoning maps, the user is required to create a layer which contains the various desired locations of manholes. To each manhole, two attributes are attached:

OUTLET - this specifies whether the manhole is the desired outlet or not. An attributed value of '1' indicates that manhole is the desired outlet and a '0' indicates otherwise. The user indicates the desired outlet by setting its *OUTLET* attribute equal to '1' and all other manholes will have their *OUTLET* parameter equal to '0'.

CON - indicates the number of direct connections to that manhole. A connection represents a household. Therefore a manhole with 5 connections indicates it has sewer lines from 5 houses draining into it. The estimated flow (in gal/day) to each manhole is calculated by:

Estimated flow = CON * PPC * FPP * PF (3.1)

where PPC is the number of persons per connection, FPP is the flow per person (in gal/day) and PF is the peak factor.

It is important to plan where to locate the manholes so that they can best serve the area. The manholes must be located in such a way that the spacing between them meets the standards set by the ASCE and the Water Pollution Control

Federation (Design, 1970). In general, the maximum spacing should range from 300 ft to 400 ft. When size of sewer pipes are large enough so that a person can enter them, spacing could be greater than 500 ft. Also in placing the manholes, the user should make sure all manholes can be linked without any of the links passing through prohibited areas.

Local regulations also determine certain factors, such as the maximum depth of manhole, minimum depth of cover for sewer lines, and the maximum allowable infiltration. Many other parameters that are relevant to the design of a sewer system for a particular area are determined by the standards set by the appropriate authorities, such as the local health board or the town or city planning department.

3.3.1 Initial Processing

The data required for the sewer system design are either acquired in digital form or are digitized manually into separate layers. Features in the layers are represented as polygons, lines, or points in the layers of a GIS database. For instance the boundaries of land parcels and other surface features in the area are represented as polygons.

The ARC/INFO software developed by Environmental Systems Research Institute, is the GIS software package used for this study. ARC/INFO is primarily a vector based GIS and is ideally suited for the design of sewer systems. A brief overview of ARC/INFO is provided in Appendix B.

A graphical user interface (GUI) has been created to assist the user in inputting the required information for running the program (see Appendix C). The information entered in the GUI menu includes the names of the layers, and the desired spacing of the grid. The grid spacing information will be used in force main path determination which is dependent on the scale of the topographic map,

the minimum depth of cover for sewer lines, and the maximum allowable drop beyond which a pump will be used instead of a gravity sewer pipe. The importance of using the grid will be described in detail in chapter 4.

Initially, all the lines that are the sides of each prohibited area polygon are extracted and stored in an ASCII file. Using the topographic data, a triangular irregular network (TIN) is generated. This TIN is used to determine the elevation of various points as the need arises. Using the prohibited areas layer, buffers are created around each of the polygons in the layer. The vertices of these created buffers are extracted for use in the determination of the force main paths. The elevations and x and y coordinates of these vertices are stored in an ASCII file.

3.4 Design Steps

Figure 3.1 shows the desired locations of manholes that will be used to explain the design procedure. Manhole #5 has been selected as the desired outfall of the system. Using the TIN generated from the topographic layer, the various surface elevations of the manhole locations are determined. The manholes are used to generate an initial network which will then be used to determine how the network layouts and the subwatersheds are generated. The initial network is created by using the extracted manhole elevations to generate a TIN network. It should be pointed out that this TIN generated from the manholes is different from the one initially generated from the topographic data, in that, the nodes represent the manholes and the links in the TIN are possible paths for sewer lines.

The layer containing the polygons that represent prohibited areas is used to perform an overlay on the generated network. This overlay, shown in Figure 3.2, is done in order to identify those links that are passing through any of the

prohibited area polygons. Any link passing through any of these polygons are deleted. The resulting network contains links which may be used as sewer lines (Figure 3.3). This network is then analyzed for flow by gravity from the various nodes to the chosen outlet or outfall node.

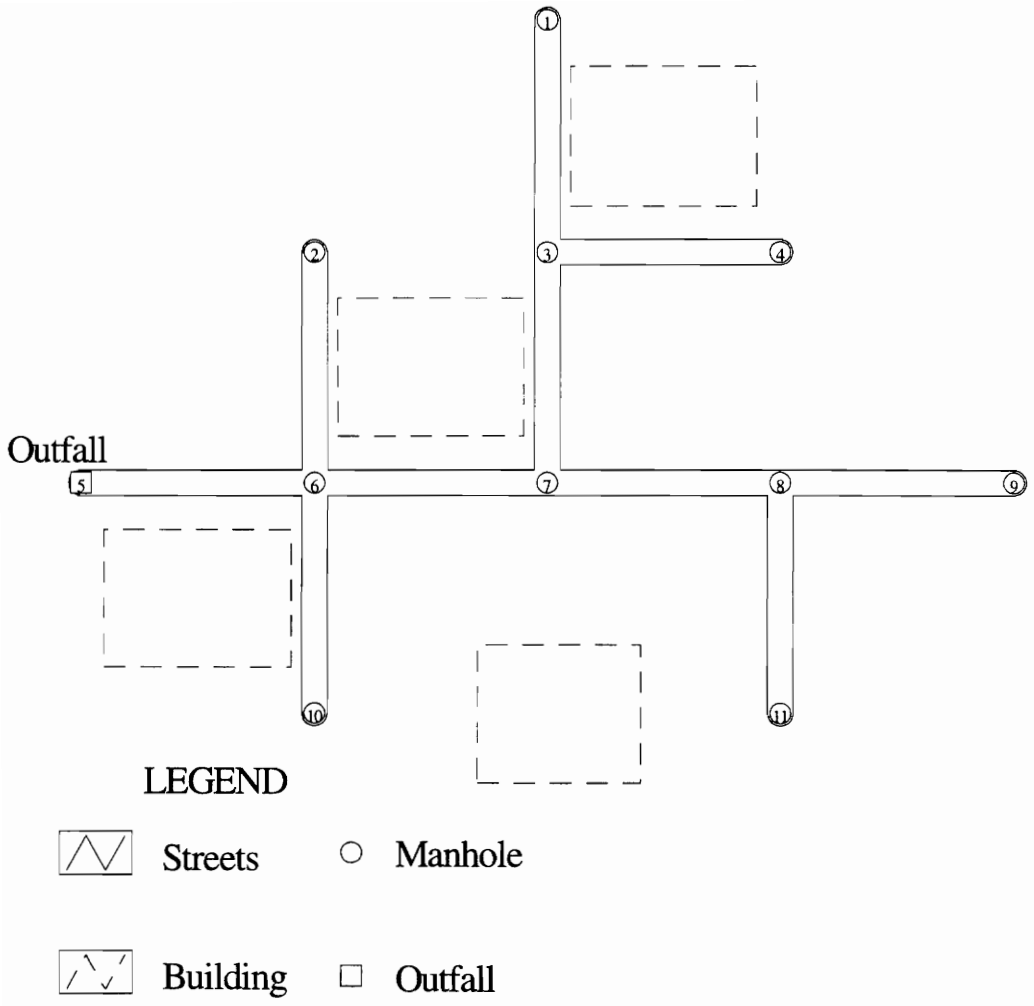


Figure 3.1 Desired Manhole Locations

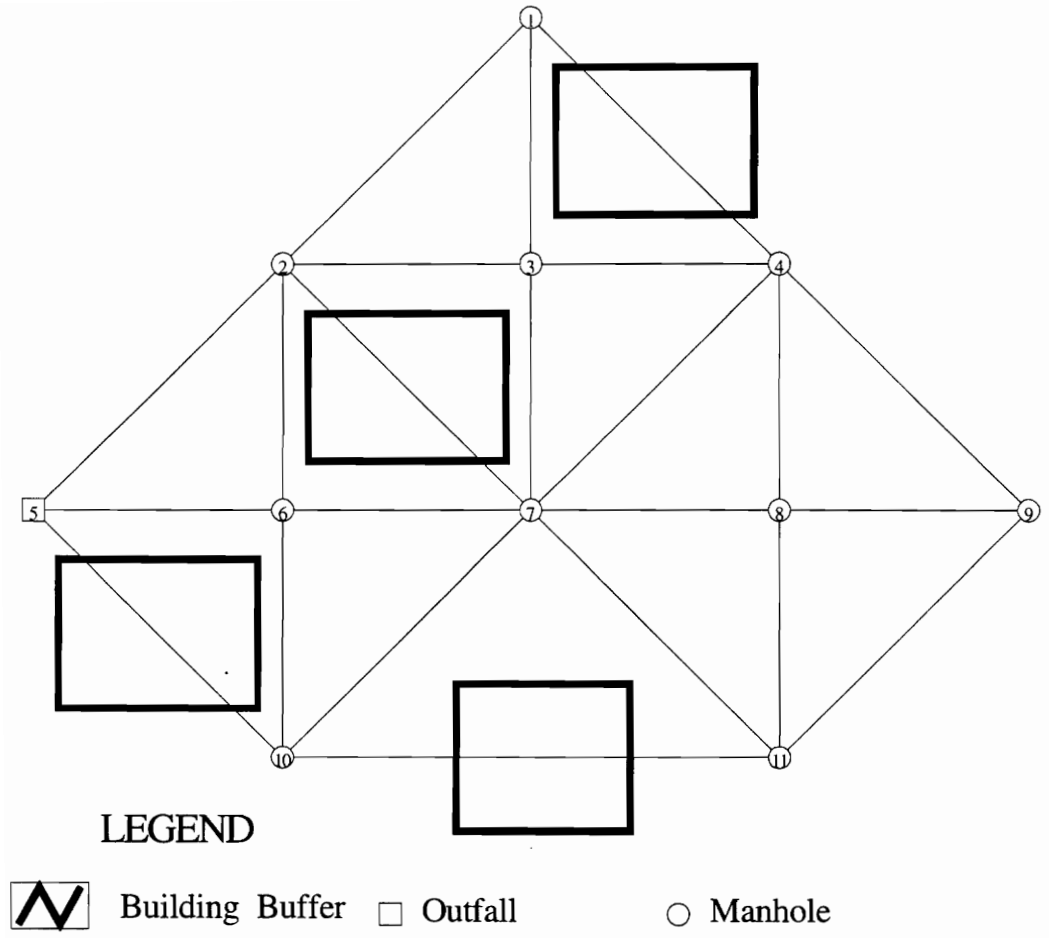


Figure 3.2 Overlay of Manhole TIN and Prohibited Area Polygons

3.4.1 Preliminary Subwatershed Delineation

The following criteria are utilized in delineating the preliminary subwatersheds. The initial network generated from the manhole elevations is examined for a maximum number of manholes that can drain by gravity to the outfall. Node *i* is drainable by gravity to node *k* if nodes *i* and *k* are directly linked and the elevation at node *i* equal to or higher than the elevation at node *k*.
A node may have more than one path to an outfall node. A node may also be drained to more than one outfall node.

The following terms used in explaining the procedure are defined as follows. The from-node (FNODE) of a link is the starting node of the link and the to-node (TNODE) of the link is its terminus or ending node. For gravity flow, elevation at the link's FNODE should be at a higher or equal level as the elevation at the TNODE. The FNODEs and TNODEs are preassigned to each link by ARC/INFO because it requires that each link be directed in terms of having an FNODE and TNODE. An *extant node* is the node that is being visited. It is also the node that is being used to perform a search for links connected to it. An algorithm called UP is used to determine all nodes that can be drained by gravity to an outfall.

Figure 3.3 is used to explain the gravity drainage determination procedure. Starting from node 5, the desired outfall, node 5 is set as the extant node. All links directly connected to node 5 are searched. The link with least length is selected. If the selected link is pointing away from the extant node, its direction is reversed.

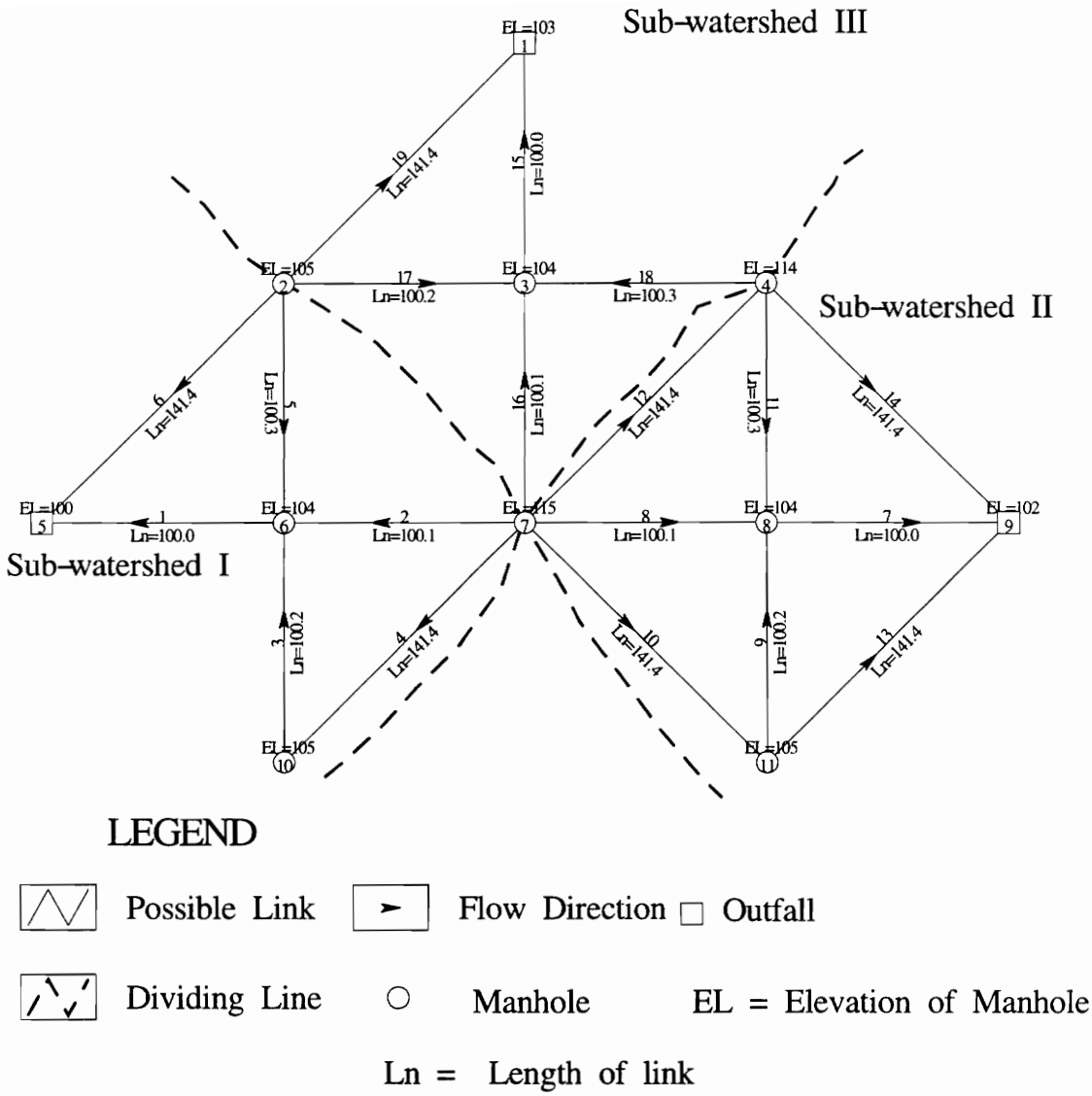


Figure 3.3 Initial/Preliminary Subwatershed Delineation

Next, a check is made to see if its FNODE elevation is higher than or equal to its TNODE elevation. If this is true, the program moves up the selected link. Else, the next shortest link connected to the extant node is selected. For example, in Figure 3.3 link 1 has the shortest length with its FNODE, node 6, with an elevation of 104 ft and TNODE, node 5, with an elevation of 100 ft and hence the program moves up link 1 which connects nodes 5 and 6. Node 6 is then set as the new extant node. All links connected directly to node 6 (links 2,3, and 5) are selected. The shortest, link 2, is then inspected to see if its FNODE elevation is at a higher or equal elevation as the extant node. Since that is true, the program travels up link 2 and sets node 7 as the new extant node.

Now, inspecting all the remaining links (4,10,8,12, and 16) directly connected to the current extant node, all these links are reversed to point to node 7. For each one of the links 4, 10, 8, 12, and 16, the FNODE (nodes 10, 11, 8, 4, and 3) elevation is at a lower elevation than the TNODE (node 7) elevation. The program therefore moves down link 2 back to node 6 and inspects links 3 and 5 which have not been traversed. Since link 3, the shortest of the two links and has its FNODE at a higher elevation than its TNODE, node 6, the program therefore goes up link 3, and then up link 4 and back down to node 6. At node 6, it goes up link 5 since link 5 is sloping toward node 6. At node 2 (the FNODE of link 5), for each of the connecting links 6, 17, and 19, when their directions are reversed to be pointing to node 2, the FNODE elevation (at nodes 1 and 3) is at a lower elevation than the TNODE (node 2) elevation. Therefore the program will move down link 5 back to node 6. Since all the links connected to node 6 have been traveled, the program will move down to node 5. At node 5, there still remains link 6 that has not been traveled. The program will therefore go up link 6, and then back to node 5. The

whole procedure will be repeated until no nodes can any longer drain by gravity to node 5 the selected outlet.

Because nodes 1,3,4,8,9, and 11 cannot be drained by gravity to node 5, the node with the least elevation (node 9) is selected from among these undrainable nodes and it becomes the new outlet for a new subwatershed (II). This method of assignment of subwatershed outlets progressively picks increasing outlet elevations with the increasing subwatershed numbers. The procedure for determining drainable nodes by gravity is repeated and this results in nodes 4, 7, 8, and 11 all draining to node 9, the new subwatershed outfall for a new subwatershed (III). Again, two nodes, namely, node 1 and 3, still remain which are not being drained to any outfall. Node 1 with the least elevation among the two is chosen as a new outfall. Again, it is determined that nodes 2, 3, 4, and 7 are all drainable by gravity to node 1. Note that node 7 drains into node 5 by links 2 and 4 and into node 1 via link 16. Once all nodes are being drained to an outlet or they are outlets themselves, the procedure ends. The next stage is to delete links that are redundant in the network.

3.4.2 Deletion of Redundant Links

Redundant links are defined as set of links that drain the same node in the same subwatershed. In Figure 3.3, three preliminary subwatersheds have been delineated: nodes 2, 6, 7, and 10 draining to node 5 (subwatershed I); nodes 4, 7, 8, and 11 draining to node 9 (subwatershed II); and nodes 2, 3, 4, and 7 draining to node 1 (subwatershed III). Node 7 for example has 2 redundant links (links 2 and 4) in subwatershed I and three redundant links (links 10, 8, and 12) in subwatershed II.

For each node that has redundant links in the same subwatershed, select the shortest redundant link and delete all other redundant links of that node in the same subwatershed. Therefore for node 7, in subwatershed I, link 2 is selected and link 4 deleted. For the same node 7 in subwatershed II, link 8 is selected and links 10 and 12 deleted. The shortest links are selected because, after the gravity drainage procedure, all links in each subwatershed are pointing in the downward direction of slope of the terrain. For a node being drained by two or more links, the total excavation costs for the subwatershed would be minimized if the shortest link draining that node is selected. The deletion procedure is applied to all the nodes with redundant links in the system. Figure 3.4 is the resulting network. Once the redundant links are deleted, the next stage is to attempt to merge the preliminary subwatersheds.

3.4.3 Merging of Subwatersheds

Because the program initially attempts to drain nodes strictly by following the downward slope of the terrain, there may be instances in which certain adjacent subwatersheds can be merged together by allowing pipes to be laid at deeper depths and in the opposite direction of ground slope to achieve gravity flow.

In order to determine whether any subwatershed k can be merged into another subwatershed j, the criterion used is whether it is possible to design a gravity sewer connection from the outfall of subwatershed k to the outfall of subwatershed j. In this design, the maximum manhole depth along the sewer lines leading from the outfall of subwatershed k to that of subwatershed j should not exceed a fixed maximum desirable manhole depth, MH_{max}. In addition, the minimum depth of cover, DC_{min} must also be satisfied together with a minimum pipe slope, S_{min}.

The design is carried out from the outfall of subwatershed k to the outfall of subwatershed j instead of just any node in subwatershed j because each time two subwatersheds are to be merged, the deciding factor would be whether it is possible to drain the outfall of subwatershed k to the out fall of subwatershed k while meeting the maximum manhole depth and minimum cover requirements. The depth at each manhole is decided by the invert elevation of the link discharging into it at the least level. Hence as long as it is possible to drain the outfall of subwatersheds k to the outfall of subwatershed j, it follows that every other node in subwatershed k can also be drained to the outfall of subwatershed j without violating the maximum manhole depth requirement.

The WCON parameter is used to identify those links that occur on the borders between two or more subwatersheds. The WCON of any link is the smallest subwatershed number that does not contain the link itself but has a manhole drained by that link. For example, in Figure 3.4, for link 2, WCON = II. These links at the borders of the subwatersheds are identified and isolated by the assigning of the WCON parameter because, in order to merge any two subwatersheds, the merging would have to occur via these links that connect the two subwatersheds. For example to merge subwatershed k to j, the connecting links m and n would be such that link m drains node p into subwatershed j and link n drains node p into subwatershed k. Only the links that drain a common node but belonging to different subwatersheds are assigned WCON values. If any two subwatersheds can be merged, the connection between them would be made up of two links that drain the same node that is common to both subwatersheds.

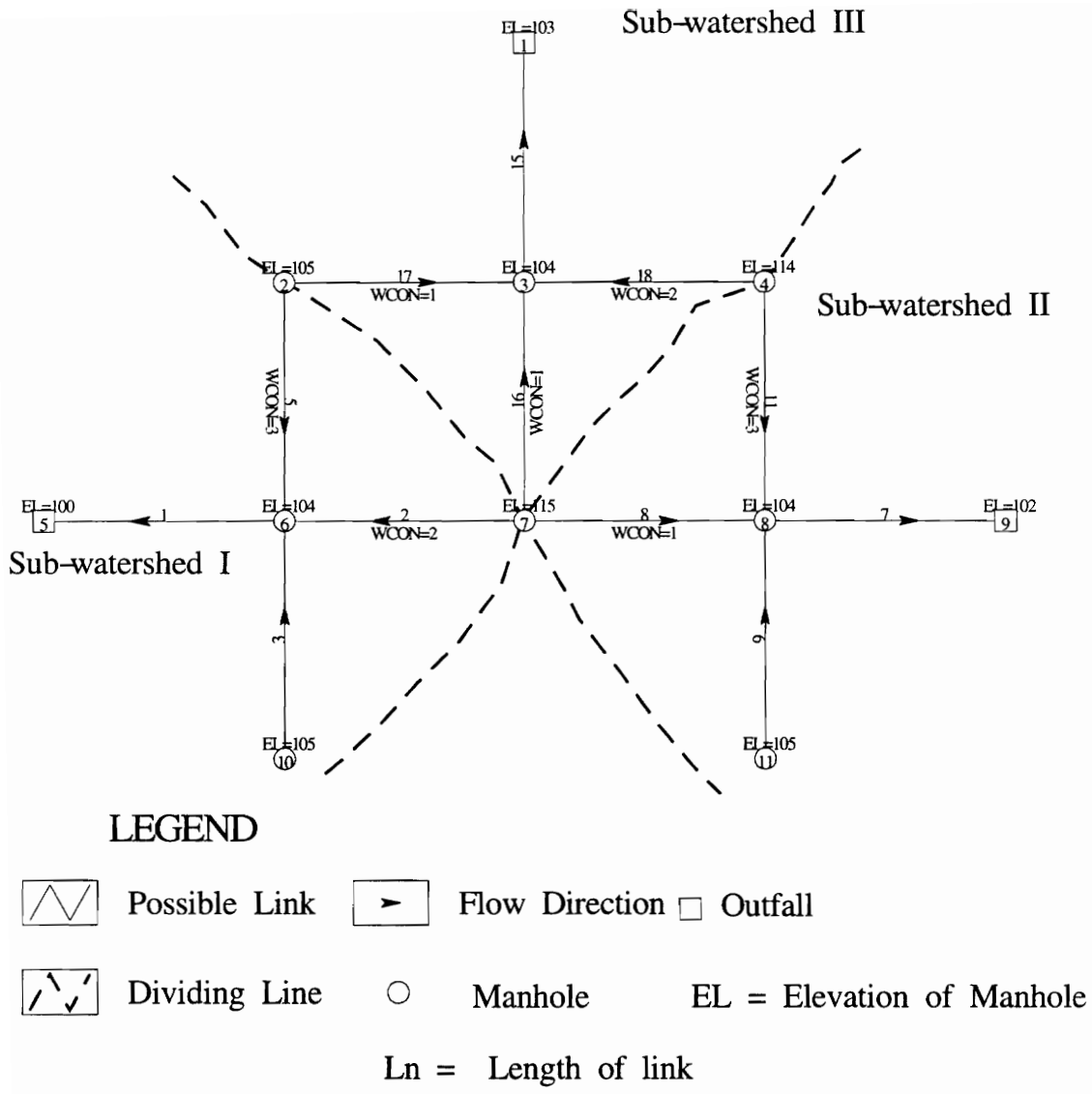


Figure 3.4 Deletion of Redundant Links.

The path followed to drain the outfall of subwatershed k to the outfall of subwatershed j is made up of two paths: the path from the outfall of subwatershed k to node p which is common to both subwatershed j and k, and the path from node p to the outfall of subwatershed j.

The merging procedure begins by assigning a WCON parameter to all links that drain nodes that belong to more than one subwatershed. The WCON for link i draining node p is defined as the least subwatershed number containing node p but not link i. For example, referring to Figure 3.4, node 7 is common to subwatersheds I, II, and III. For link 2, its WCON = II because II is the least subwatershed number containing node 7 but not link 2. For link 8 draining node 7, WCON = I because link 8 is not in subwatershed I but node 7 is common to I, II, and III. Link 16 also drains node 7 which is common to subwatersheds I, II, and III but link 16 is in subwatershed III. Therefore, for link 16, WCON = min (I, II) which is I. Link 18 is in subwatershed III and drains node 4 which is common to II and III, hence WCON equals II. Similarly, WCON equals I for link 17, WCON equals III for link 11.

Because the desired outfall is in subwatershed I, the initial approach would be to try to determine if any of the subwatersheds II and III can be merged to subwatershed I. A search is made for all links whose WCON = I resulting in the selection of links 17, 16, and 8. Of these three links, the link with the least FNODE elevation is selected. For example, the elevation of the FNODE of link 8 is 115 ft, that for link 16 is 115 ft, and for link 17, 105 ft.

The link with the least FNODE elevation is selected because in order to merge the subwatersheds, it will be required to lay the pipes against the direction of ground slope and the link with the least FNODE elevation is the most likely to

have the least difference between its outfall elevation and its FNODE elevation. This least difference between the link's FNODE elevation and its subwatershed outfall elevation is likely to result in the least volume of excavation. The idea is to make the sewer pipe close to the ground surface. Referring to Figure 3.5, all possible connections between nodes 1 and 5 are shown. Clearly, the path through node 2, the fnode of link 17, is better. Link 17 is therefore selected.

Since link 17 belongs to subwatershed III, an attempt will be made to merge subwatershed III with subwatershed I. An approximate design will then be performed from manhole 1, the outfall of subwatershed III to manhole 5, the outfall of subwatershed I. The approximate design is carried out from the outfall manhole of subwatershed III (manhole 1) because, if flow from manhole 1 can be drained by a gravity pipe to manhole 5 in subwatershed I, then flow from all other manholes in subwatershed III can also be drained by gravity to manhole 5.

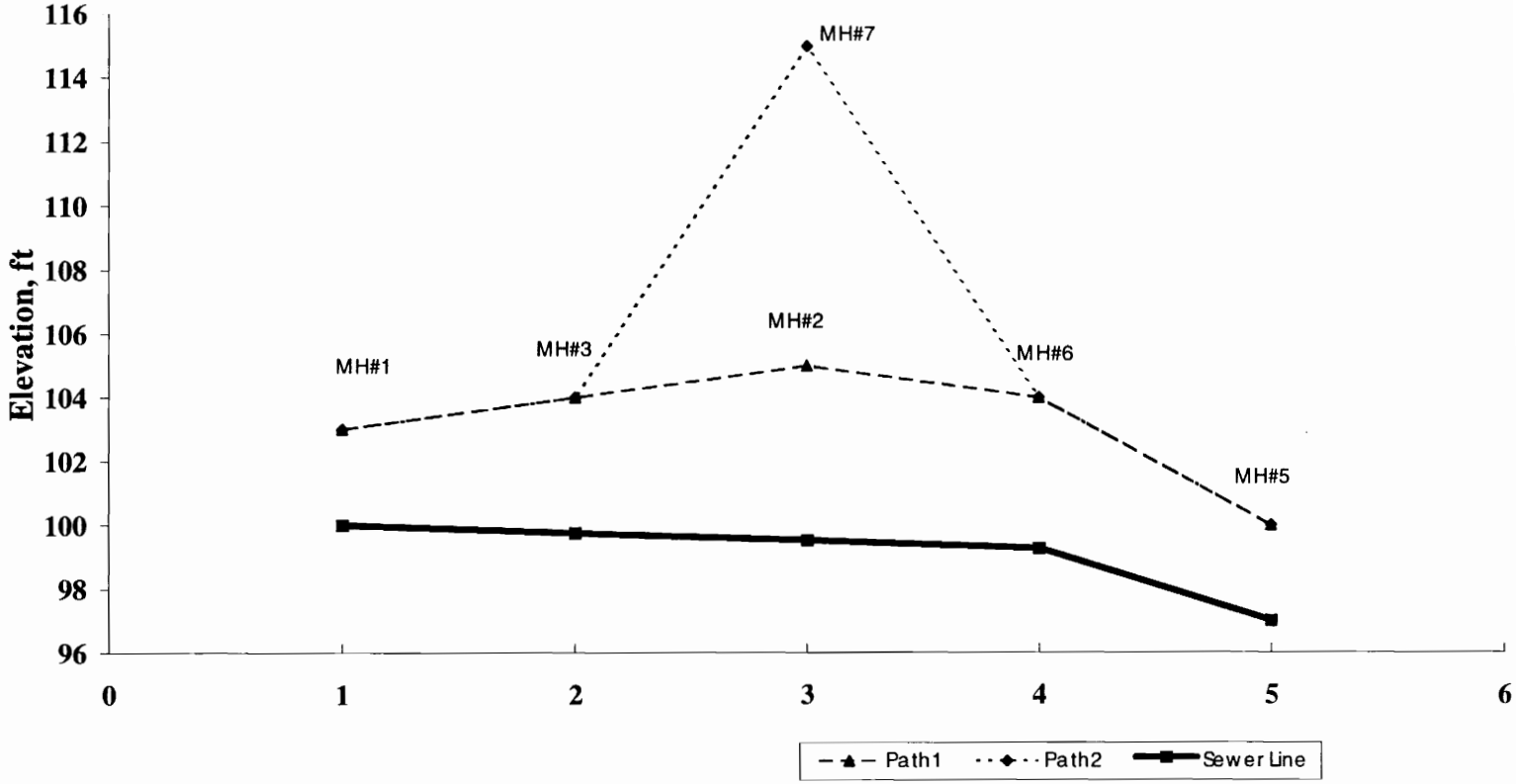


Figure 3.5 Profile of Alternative Paths Between Manhole #1 and #5

3.4.4 Approximate Design Procedure:

The approximate design is based on a 10 inch pipe with a desired minimum velocity of 2 ft/s. The required minimum slope for such a pipe is 0.0024 ft/ft. In the approximate design, the following constraints must be satisfied:

1. Minimum slope >= S_{min} = 0.0024 ft/ft.
2. Maximum depth of manhole <= MH_{max} = 10 ft.
3. Minimum depth of cover for each sewer is >= DC_{min} = 3 ft.

Links i = 1 to N represent consecutive the set of links that lead from the outfall of subwatershed k to the outfall of subwatershed j. The steps used are as follows:

Step 1: Set i = 1. Set $Du_i = DC_{min}$. Set FLAG = 0.

Step 2: $Dl_i = Du_i + (S_{min} * Ln_i) - (Eu_i - El_i)$

Step 3: If $Dl_i < DC_{min}$, then $Dl_i = DC_{min}$

Step 4: If $Dl_i <= MH_{max}$, then $Du_{i+1} = Dl_i$, go to Step 5. Else go to step 7.

Step 5: If i < N, then Set i = i + 1, go to Step 2. Else go to Step 6.

Step 6: End.

Step 7: FLAG = 1.

Where Eu_i = upper manhole elevation of link i; El_i = lower manhole elevation of link i; Du_i = depth at upper manhole of link i; Dl_i = depth at lower manhole of link i; Ln_i = length of link i.

When the above checking procedure returns a FLAG = 1, it implies that to let subwatershed k drain into subwatershed j by a gravity sewer pipe, there will exist at least one manhole whose depth will exceed the set maximum allowable depth. This implies that subwatershed k can not be merged to subwatershed j. Hence flow from the outfall of subwatershed k can not be drained by gravity connection.

Applying the approximate design procedure between node 1, the outfall of subwatershed III and node 5 (the desired outfall), the outfall of subwatershed I, we

have the following links in the path: link {1,3}:i=1, link {3,2}:i=2, link {2,6}:i=3, and link {6,5}:i=4 (see Figure 3.4). The following steps show the intermediate and final results when the approximate design procedure is applied between manhole 1 and manhole 5:

Starting from node 1,

$Du_1 = 3.$

$Dl_1 = Du_2 = 3 + (.0024 * 100.0) - (103 - 104) = 4.24;$

$Dl_2 = Du_3 = 4.24 + (.0024 * 100.2) - (104 - 105) = 5.48;$

$Dl_3 = Du_4 = 5.48 + (.0024 * 100.3) - (105 - 104) = 4.72;$

$Dl_4 = 4.72 + (.0024 * 100.0) - (104 - 100) = 0.96;$

Since $Dl_4 = 0.96 < DC_{min} = 3$, $Dl_4 = 3.$

Looking at the resulting Du and the Dl values which represent the manhole depths, it can be seen that none exceeds the set allowable maximum of 10 ft. Hence subwatershed III to which link 17 belongs can be merged with subwatershed I by the using gravity sewer lines. When subwatershed III is merged with subwatershed I, flow from nodes in subwatershed III will be directed to subwatershed I through link 17.

Link 16, as shown in Figure 3.4, which also has its WCON = I will be deleted because its presence would mean that node 7 is drained twice to the same outfall which would be redundant. When deleting the links after the subwatersheds are merged, the links are prioritized as follows: links in the subwatershed with the desired outfall take priority over every other link in other subwatersheds; links in subwatersheds with higher subwatershed numbers take priority over links in subwatersheds with lower subwatershed numbers. This is because, the

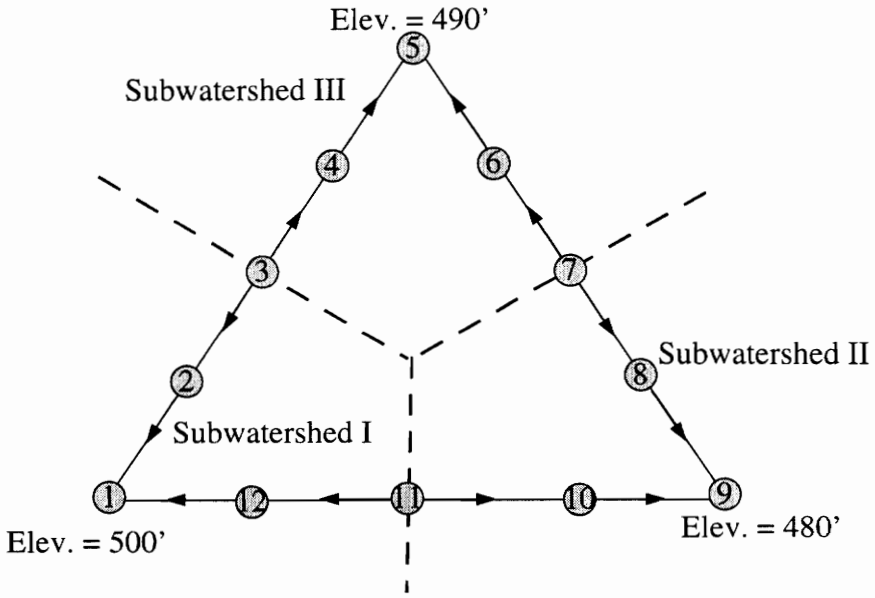
subwatershed number by assignment is a rank of the elevation of the subwatershed outfalls.

Figures 3.6a and 3.6b illustrate the criteria for the deletion of links redundant links after any two or more subwatersheds are merged. On Figures 3.6a and 3.6b, it is seen that manhole 3 drains into either subwatershed I or III; manhole 11 drains into either I or II; and node 7 drains into II and III. The link {2,3} is kept and link {3,4} is deleted because all links in the subwatershed with the main outfall have higher priority. Similarly, link {12,11} is kept and {11,10} is deleted. Because subwatershed III's outfall is at a higher elevation in comparison to subwatershed II's, link {6,7} is kept and {7,8} is deleted by the priority rules. To validate these rules, the following points are observed:

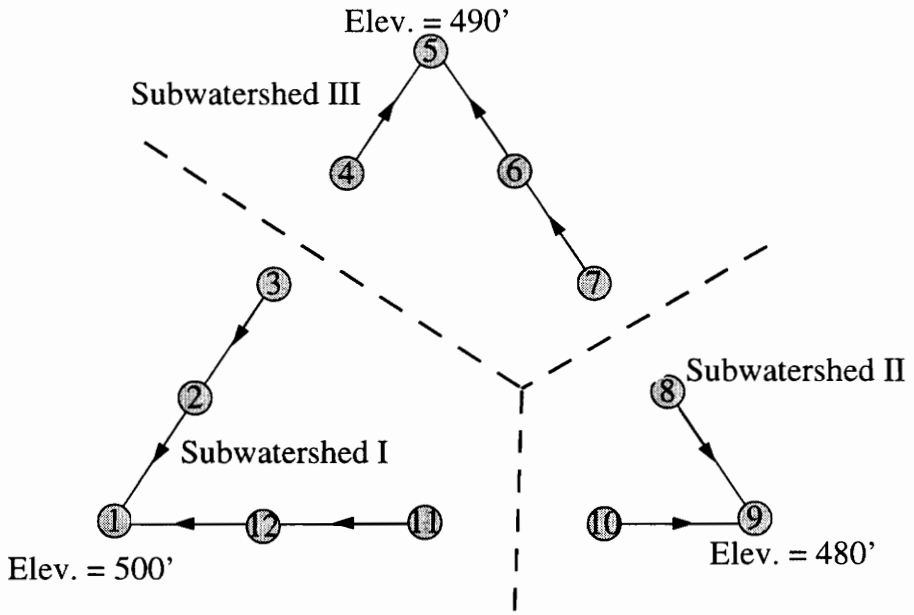
1. If a link (eg. link {2,3}) drains a node into the subwatershed with the main outfall (eg. node 3) by gravity, obviously, it is more economical to maintain such a link than to allow the node to be drained into an adjacent subwatershed via link {3,4} and eventually pumping waste water back to the subwatershed with the main outfall.
2. When a node, such as node 7 can be drained into two different subwatersheds (none of which contains the main outfall), the rule of diverting flow to the higher elevation outfall minimizes the static pump head. The static pump head between I and III is 10 ft (500-490), as opposed to 20 ft (500-480) for subwatershed II.

The outfall of subwatershed II will be at a lower elevation than the outfall of subwatershed III because of the fact that the subwatershed numbers indicate the ranking of the subwatershed outfall elevations. If a node is being drained by a link in subwatershed II and by another link in subwatershed III, the link in subwatershed II will be deleted and the one in subwatershed III maintained.

(a) Pre-merger



(b) Post-merger



Figures 3.6 a and b. Illustration of Redundant Link Deletion after Merger

It is preferred to maintain links in higher subwatershed numbers because, assuming it is not possible to merge either of subwatersheds II or III with subwatershed I, water from both subwatersheds II and III will have to be pumped to get to subwatershed I which contains the desired outfall. It is more desirable to have more of the flow in the subwatershed with a higher outfall elevation since that will make pumping cheaper.

The resulting network from the merging of subwatershed III to I and redeletion of redundant links is shown seen in Figure 3.7. Link 11, whose original WCON = 3, is now connected to subwatershed I, its WCON is set to I.

Again, a search for links with WCON = I produces links 11 and 8. The FNODE elevation for link 8 is 115 ft, and link 11 has FNODE elevation = 114 ft.

Since link 11 has the least FNODE elevation, the approximate design will be performed through link 11. Again, applying the approximate design procedure between node 9, the outfall of subwatershed II and node 5 (the desired outfall), the outfall of subwatershed I, we have the following: link {9,8}:i=1, link {8,4}:i=2, link {4,3}:i=3,link {3,2}:i=4, link {2,6}:i=5, and link {6,5}:i=6 (see Figure 3.6). The results of the approximate design is as follows:

Starting from node 9,

$Du_1 = 3.$

$Dl_1 = Du_2 = 3 + (.0024 * 100.0) - (102 - 104) = 5.24;$

$Dl_2 = Du_3 = 5.24 + (.0024 * 100.3) - (104 - 114) = 15.48 > 10$ ft;

$Dl_3 = Du_4 = 15.48 + (.0024 * 100.3) - (114 - 104) = 5.72;$

$Dl_4 = Du_5 = 5.72 + (.0024 * 100.2) - (104 - 105) = 6.96;$

$Dl_5 = Du_6 = 6.96 + (.0024 * 100.3) - (105 - 104) = 6.20;$

$Dl_6 = 6.20 + (.0024 * 100.0) - (104 - 100) = 2.44;$

Since $Dl_6 = 2.44 < DC_{min} = 3$, $Dl_6 = 3.$

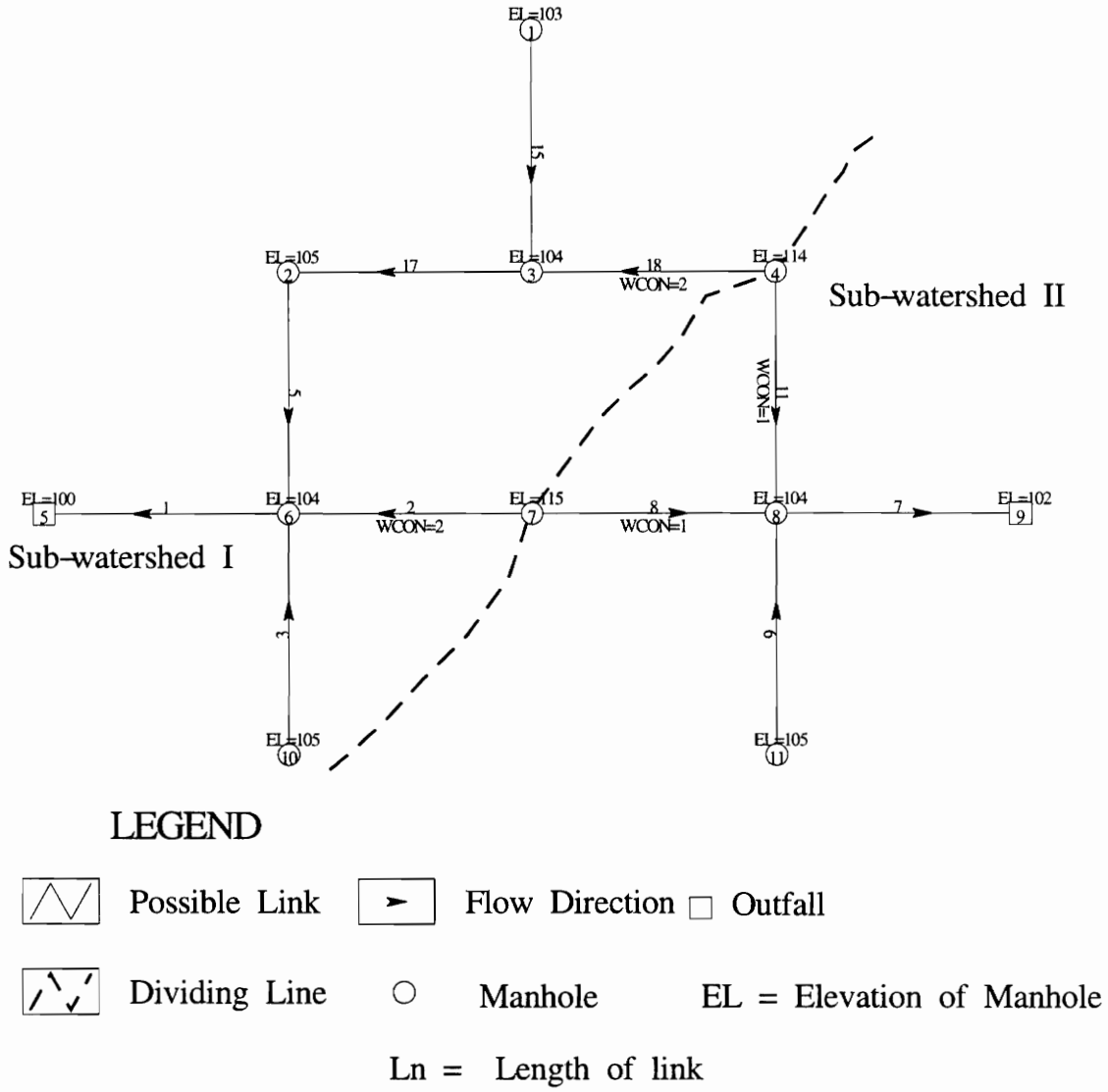


Figure 3.7. Subwatershed III Merged Into Subwatershed I

Looking at the resulting Du and the Dl values, it is seen that for i = 2, the lower manhole depth, Dl_2, exceeds the set allowable maximum of 10 ft. This implies that a gravity connection can not be used to link subwatersheds II to subwatershed I. Hence links 11 and 8 are deleted resulting in two distinct subwatersheds. The resulting subwatersheds are shown on Figure 3.8. A pump will therefore have to be used at node 9 to pump water from subwatershed II to a manhole in subwatershed I and then be allowed to flow by gravity to the outlet of subwatershed I.

The decision as to which manhole in subwatershed I will be the destination for waste pumped from subwatershed 2 and also the optimal path of the force main line is determined by a shortest path optimization procedure described in chapter 4.

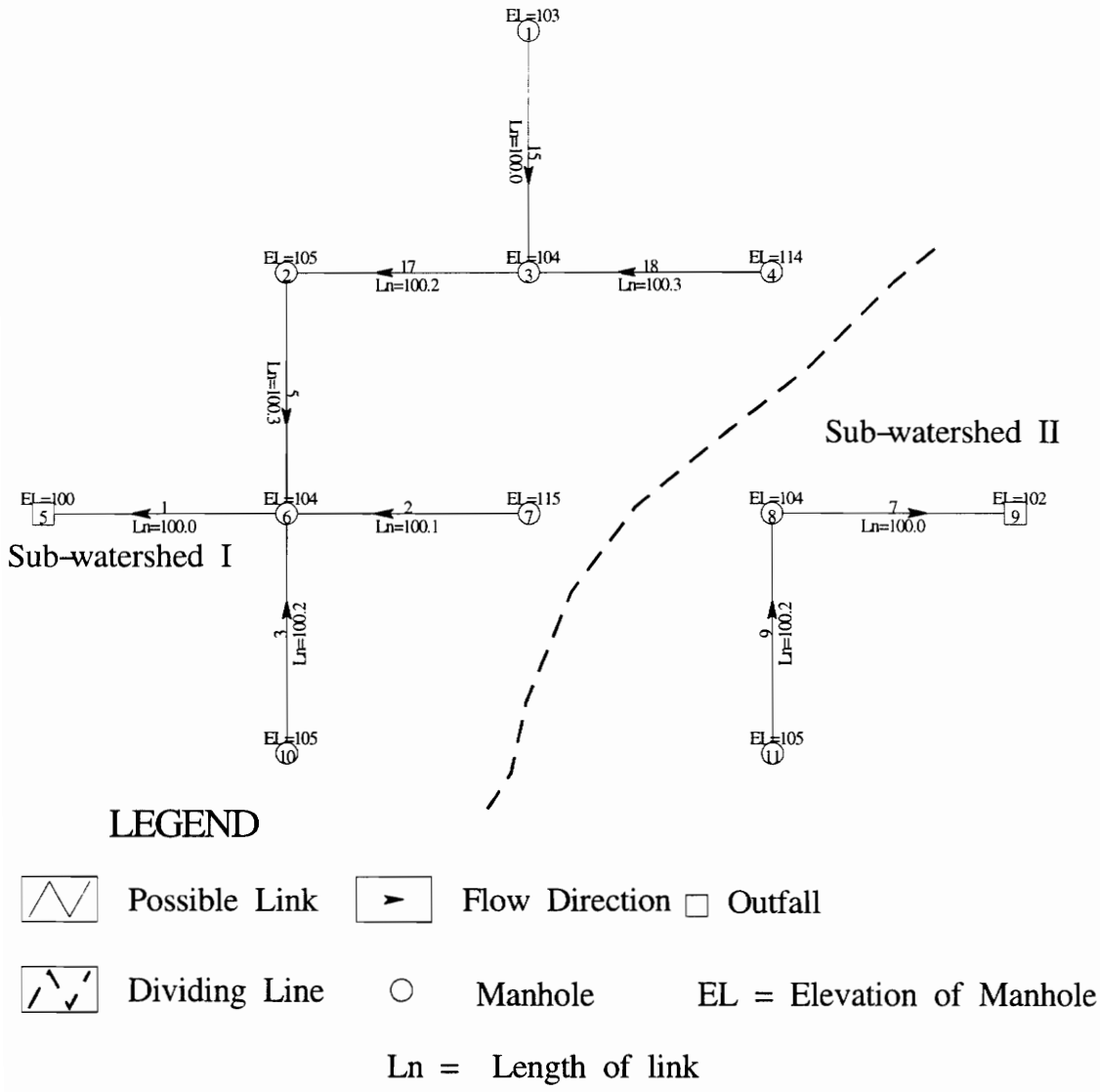


Figure 3.8 Two Unique Subwatersheds Delineated

Chapter 4 - Force Main Destination and Path Determination

4.1 Introduction

This chapter addresses the procedure used in determining the pumping destination of waste water pumped from the outlet of a subwatershed and the path taken by the force main. The procedure for determining the ordering of the subwatersheds is described in the proceeding steps.

Suppose N subwatersheds have been delineated and the desired outlet for the whole system is in subwatershed 1. Wastewater from subwatersheds 2 to N would then have to be pumped to subwatershed 1. The subwatersheds are considered as super nodes. For these super nodes, a minimal spanning tree is generated. This tree dictates how the subwatersheds are connected. For example if super node k is upstream of super node j, the waste from the outfall of subwatershed k will be pumped to a manhole in subwatershed j. Alternatively, the user can also dictate (interactively) the order in which the subwatersheds are to be connected by editing the subwatershed connectivity file and entering the desired connectivity of the subwatersheds based on their relative locations. The process of determining which manhole in subwatershed j the waste will be pumped to is determined using an optimization procedure. In the illustrated example, only two subwatersheds have been delineated; therefore wastewater from subwatershed 2 will be pumped to a manhole in subwatershed 1 because the desired outfall, manhole 5 is in subwatershed 1.

The summation of the local flows for each subwatershed begins after the connectivity information is entered. To determine the flow to any manhole, the number of connections to that manhole, number of persons per connection, flow per person and the peak factor must be known. A connection in this sense

represents a household. Suppose for a manhole, there are 20 connections, 3 persons per connection, 100 gallons per person per day, and peak factor is 2.5, then total flow to that manhole is 20 * 3 * 100 * 2.5 = 15,000 gallons per day.

If there are N delineated subwatersheds, then the total on-site flow, p, to the outfall of subwatershed *i* is given by:

$$p_i = \sum_{m=1}^{M} q_m \tag{4.1.1}$$

where M is the number manholes belonging to subwatershed *i* and q_m is the flow to manhole m.

After the total local flows for each subwatershed has been determined, the next stage is to determine the actual total flow at each subwatershed outfall considering flows from upstream subwatersheds. That is:

$$Q_i = \sum_{j \in U} p_j + p_i \tag{4.1.2}$$

where Q_i is the total flow to the outfall of subwatershed *i*, and U is the set of all subwatersheds *j* that are upstream of subwatershed *i*. Based on Q_i, the force main diameter is calculated. The pump power requirement, P, for each subwatershed is defined by

$$P = \gamma H_{ik} Q_i . \tag{4.1.3}$$

γ is the specific gravity of the wastewater.

H_{ik} is the total pump head to the waste from subwatershed i to subwatershed k, the subwatershed downstream of subwatershed I. H_{ik} is defined as:

$$H_{ik} = h_{ik(stat)} + hf_{ik} \tag{4.1.4}$$

in which:

$$hf_{ik} = \frac{8fl_{ik}Q_i^2}{\pi^2 g d^5} \tag{4.1.5}$$

By using $v = 4Q_i / \pi d^2$, hf_{ik} can be expressed as follows:

$$hf_{ik} = \frac{fl_{ik}}{4g}\sqrt{\frac{\pi v^5}{Q_i}} \qquad (4.1.6)$$

$h_{ik(stat)}$ is the static head between the outfall of subwatershed i and the manhole in subwatershed k into which the force main terminates;

hf_{ik} is the friction head in the pipe, in feet;

l_{ik} is the length of force main pipe linking subwatershed i and k, in feet;

d is the diameter of the force main, in feet;

g is the acceleration due to gravity, in ft/s^2;

f is the friction factor;

v is the velocity of flow in the force main, ft/s.

4.2 Modified Length And Equivalent Length Concept

It is important to know the path of the force main from each outfall and the manhole into which the force main terminates. With this information, the length of the force main can be determined, hence the associated friction head, trenching cost and pipe costs can be determined. The concept of modified lengths and equivalent lengths is used in comparing pumping costs and force main costs from one subwatershed to another.

In selecting pumps, the two parameters that principally dictate the size of the pump are the flow rate of the water and the pump dynamic head (H_T). The pump head is defined as the sum of all the losses and vertical height (static head) over which water has to be lifted. The total dynamic head (H_T), in feet, can be expressed as:

$$H_T = H_{stat} + h_{fs} + h_{fd} + \sum h_{ms} + \sum h_{md} + \frac{V_d^2}{2g} \quad \text{(see Figure 4.1)} \qquad (4.2.1)$$

(Tchobanoglous, 1981)

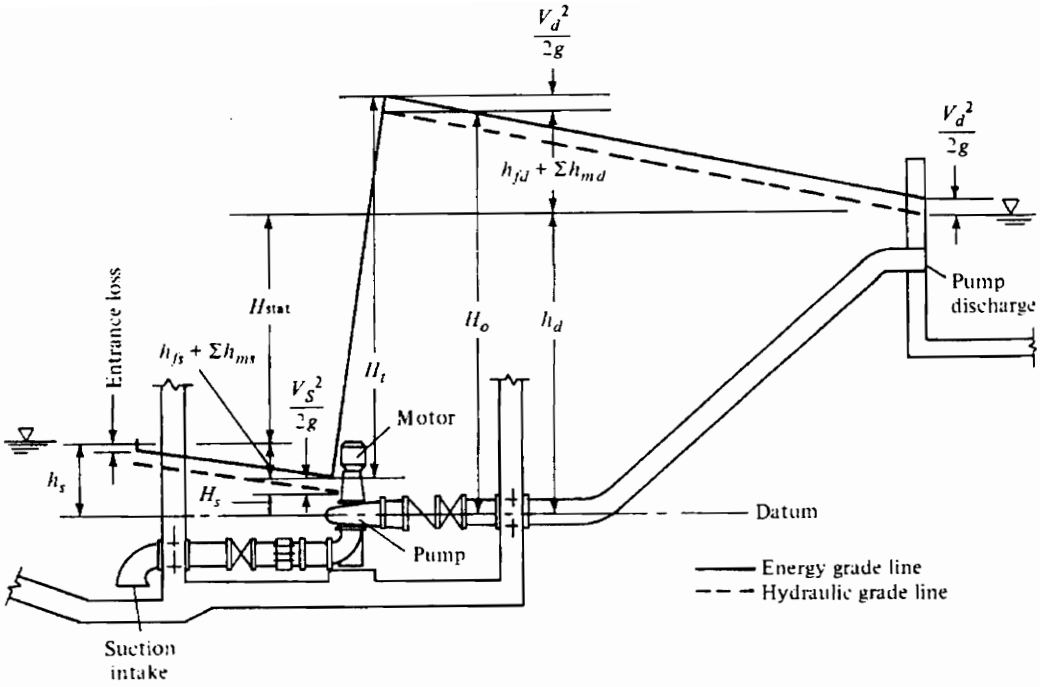


Figure 4.1 Definition Sketch For The Head On A Pump

(Source: Tchobanoglous, 1981)

H_{stat} = static head;

h_{fs} = friction head loss in suction piping, ft;

h_{fd} = friction head loss in discharge piping, ft;

h_{ms} = minor fitting and valve losses in suction piping system, ft

h_{md} = minor fitting and valve losses in discharge piping system, ft

V_d = velocity in discharge nozzle, ft/s;

g = acceleration due to gravity, 32.2ft/s^2

The static discharge head, h_d is the difference between the elevation of the discharge liquid level and the centerline of the pump impeller (see Figure 4.1). Static suction head, h_s is the difference between the elevation of the suction liquid level and the centerline of the pump impeller. The friction head is the head of water that must be supplied to overcome frictional loss caused by fluid flow through the pipes in the system. The velocity head, $\frac{V^2}{2g}$, is the energy contained in the liquid being pumped at any point in the system. The minor fitting and valve losses is the head of water required to overcome losses through fittings and valves in the system.

The power required to pump the liquid from a pump station to a destination, is defined as:

$$P = \gamma Q H_T . \qquad (4.2.2)$$

where P is the power required, γ is the specific gravity of the wastewater, Q is the flow of the wastewater, and H_T is the pump total dynamic head. To pump wastewater from a pump station to one of several destinations, no matter what that single destination is, the flow rate for the pump has to be the total wastewater flow at that subwatershed outfall. Hence the decision made is mainly based on choosing the destination that has the least total dynamic head.

The following modifications can be made to the definition of H_T as follows. For all purposes, h_s, h_{fs}, and h_{ms} will remain constant. For V_d, a recommended non-settling velocity of 4 ft/s is used (Design, 1970). Since all these parameters are constant, their quantity will not affect the choice of the H_T . The minor fitting and valve losses would also remain negligible compared to the total length of pipe. Hence the head corresponding to the discharge side of the force main, denoted by H_{dis}, could be redefined as:

$$H_{dis} = H_{stat} + h_{fd} + h_{md} \tag{4.2.3}$$

where H_{dis} is the discharge head in feet;

For any comparison between 2 or more destinations, the choice will be based on the friction losses in discharge pipes and the static head.

The friction head loss, h_{fd} is directly proportional to the length of pipe. Using the Darcy-Weisbach equation, the static head is also converted to an equivalent length of pipe.

From the Darcy-Weisbach equation, head loss is given by

$$h_{loss} = \left(\frac{fv^2}{2dg}\right)L, \tag{4.2.4}$$

where h_{loss} is the head loss in the pipe in feet; L is the length of pipe in feet, v is the velocity in the pipe in ft/s; d is the diameter of the pipe in feet; g is the acceleration due to gravity in ft/s^2; and f is the friction factor. For fully rough, turbulent flow,

$$f = \left[2\log\left(\frac{d}{e}\right) + 1.14\right]^{-2} \tag{4.2.5}$$

in which e is the pipe roughness in feet.

By replacing the h_{loss} by H_{stat}, the sum of static head loss, an equivalent length is obtained as

$$L_{eq} = \left(\frac{fv^2}{2dg} \right)^{-1} H_{stat} . \quad (4.2.6)$$

The modified length (distance) between a pump station and any destination is therefore defined as:

$$L_{mod} = L_s + L_{eq} \quad (4.2.7)$$

where L_{mod} is the total modified length of the pipe; L_{eq} is the equivalent length resulting from the static head and L_s is the actual length of the pipe which takes into account the frictional head loss. L_s is defined as:

$$L_s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} . \quad (4.2.8)$$

4.3 Force Main Path Determination

By converting the static head to an equivalent length, the problem of determining least pumping cost can therefore be solved as a shortest path problem using the Dijkstra's shortest path algorithm (Phillips et al, 1976). The minimum energy cost is attained when $\sum_{i \in M} L_{mod}$ is a minimum. Where M is the set of all pipe segments that make the force main.

The force main path and the destination determination procedure begins by the creation of a grid layer for the area of interest. The grid layer is needed for the representation of the terrain. By the use of grid points, the elevations of several points on the land surface can be determined. These grid points, together with the vertices of the buffer polygons created around prohibited areas, and manhole locations and their elevations are then used in determining the force main path. The grid points are used to analyze the change in topography in the area between a source and destination. An example grid point layer is illustrated in Figure 4.2. The points from the grid layer, the vertices of the prohibited area buffers, the

manhole locations from subwatershed I and the manhole representing the outlet of subwatershed II are all combined as shown in Figure 4.2. By overlaying these points with the TIN created from the topographic data, the elevations of all these points are determined.

Next, a network is generated in which every point is connected to every other point. By overlaying the generated network with the prohibited areas layer, links passing through locations of prohibited areas are deleted. In addition , if the grid spacing is w, all links with length, v, such that $v > \sqrt{2w^2}$ is also deleted. The value $\sqrt{2w^2}$ represents the length of the diagonal of a box formed by four adjacent grid points from the grid layer. Links greater in length than $\sqrt{2w^2}$ are deleted because their presence will cause the shortest path finding algorithm to skip certain grid points and hence overlook any changes in topography along that link. For the illustrated example, the resulting network is shown in Figure 4.3 and is used for the shortest path analysis. Each of the links is undirected, that is, it can be traversed in both directions. Using the modified head loss equation, an equivalent length of each link is calculated. The equivalent length, which accounts for the static head in that link is added to the slope length which accounts for the friction loss head in the link to obtain the modified length of that link. Each link segment in the network is assigned a modified length.

A decision is to be made concerning to which manhole in subwatershed I the flow from the outfall of subwatershed II will be pumped. The shortest and least cost path is determined based on the modified length of the links from manhole 9 (the outfall of subwatershed II) to each of the manholes 1, 2, 3, 4, 5, 6,

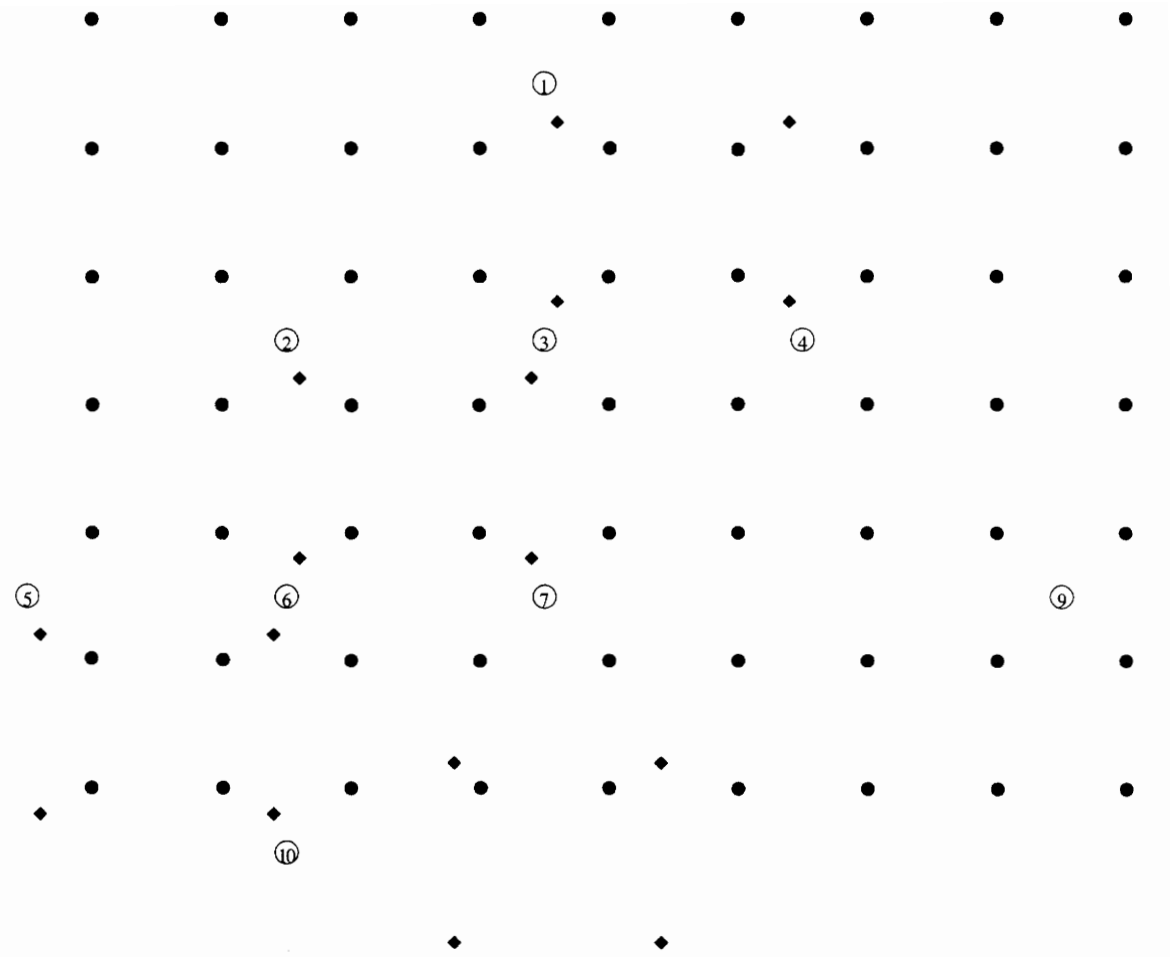
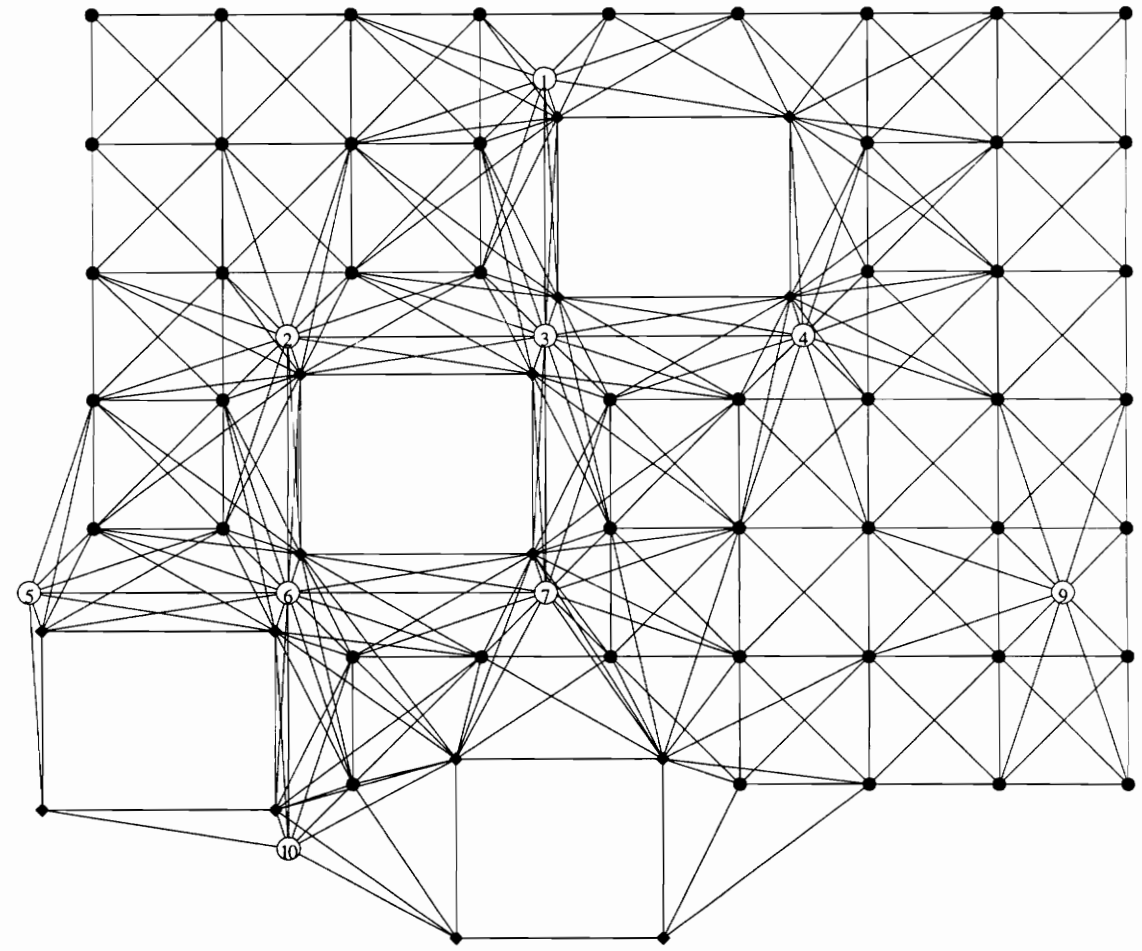


Figure 4.2 Grid Points, Buffer Vertices, and Manholes Overlay



LEGEND

Grid Point

Possible Link

Building Buffer

Manhole

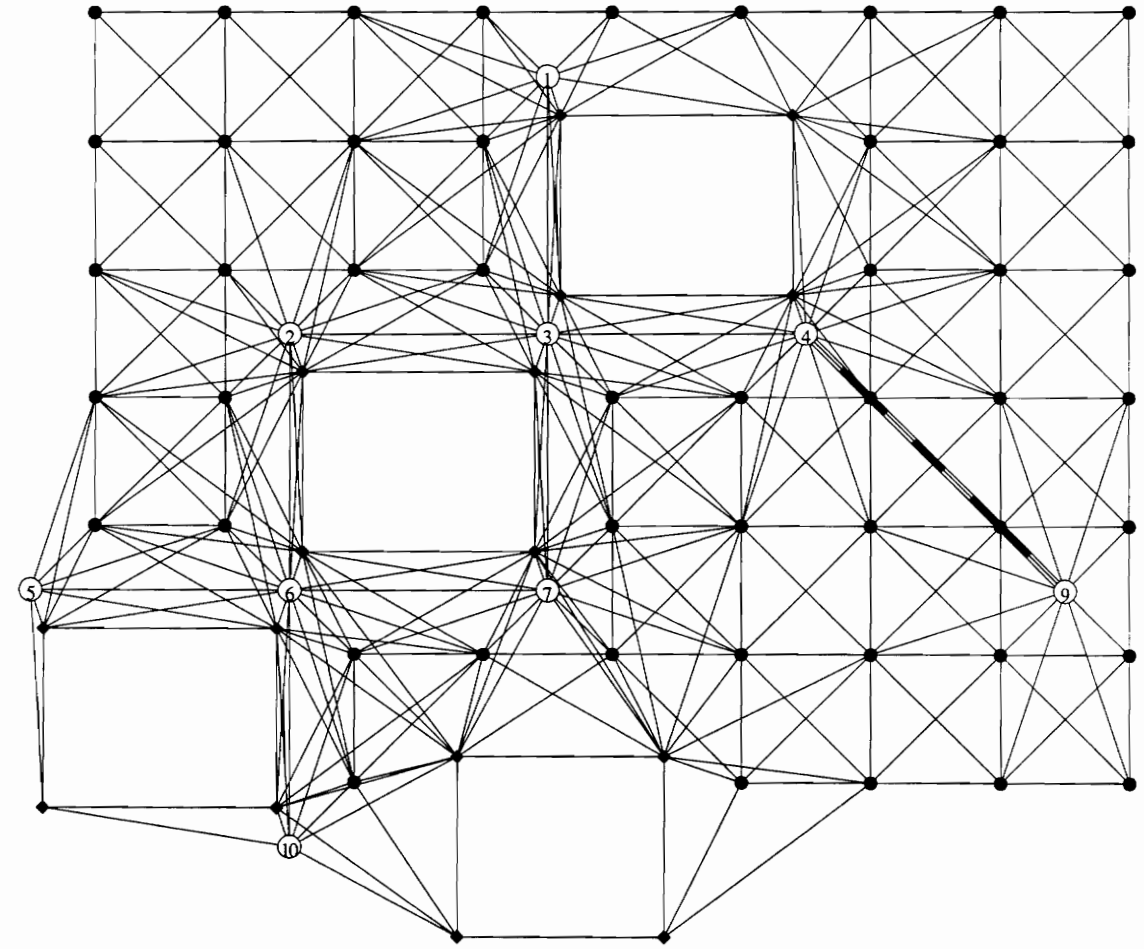
Figure 4.3 Network Resulting From Deletions

7, and 10 in subwatershed I. The destination that yields the least $\sum_{i \in M} L_{mod}$ is then chosen as the manhole in which the force main from manhole 9 would terminate. The path associated with the least $\sum_{i \in M} L_{mod}$ will chosen as the path for the force main. For example, in the present illustration, the least cost path and destination is one that goes to manhole 4 in subwatershed I (see Figure 4.4). The path for the force main has also been indicated. The sewer network for each subwatershed and the connecting force main is shown in Figure 4.5.

In this case there are only two subwatersheds. In cases where several subwatersheds exist, the order of linking the subwatersheds (also referred to as supernodes) is accomplished according to the following two steps:

1. For k = 2 to N and j = 1 to N - 1, $j \neq k$, the link {k, j} is formed. Any link {k, j} refers to a possibility of pumping flow from the outfall of subwatershed k to a manhole in subwatershed/supernode j. For each link {k, j}, the length is determined by the distance between the outfall manhole of subwatershed k and the manhole in subwatershed j that is closest to the outfall of subwatershed k.
2. Using the links {k, j}, a minimum spanning tree is generated which connects all the supernodes to form a tree network with supernode 1 at the root of the tree.

With this initial tree, the force main path and destination determination procedure is then applied between each of the two sets of the subwatersheds {k, j} that are linked directly in the minimum spanning tree. The minimum spanning tree is only used by the program as an initial rudimentary way of determining the order in which the subwatersheds are linked. The user has the option of specifying the order of linking the subwatersheds and then letting the program apply the force main path and destination determination procedure.



LEGEND

Grid Point

Force main Path

Building Buffer

Manhole

Figure 4.4 Least Cost Path and Destination Determination

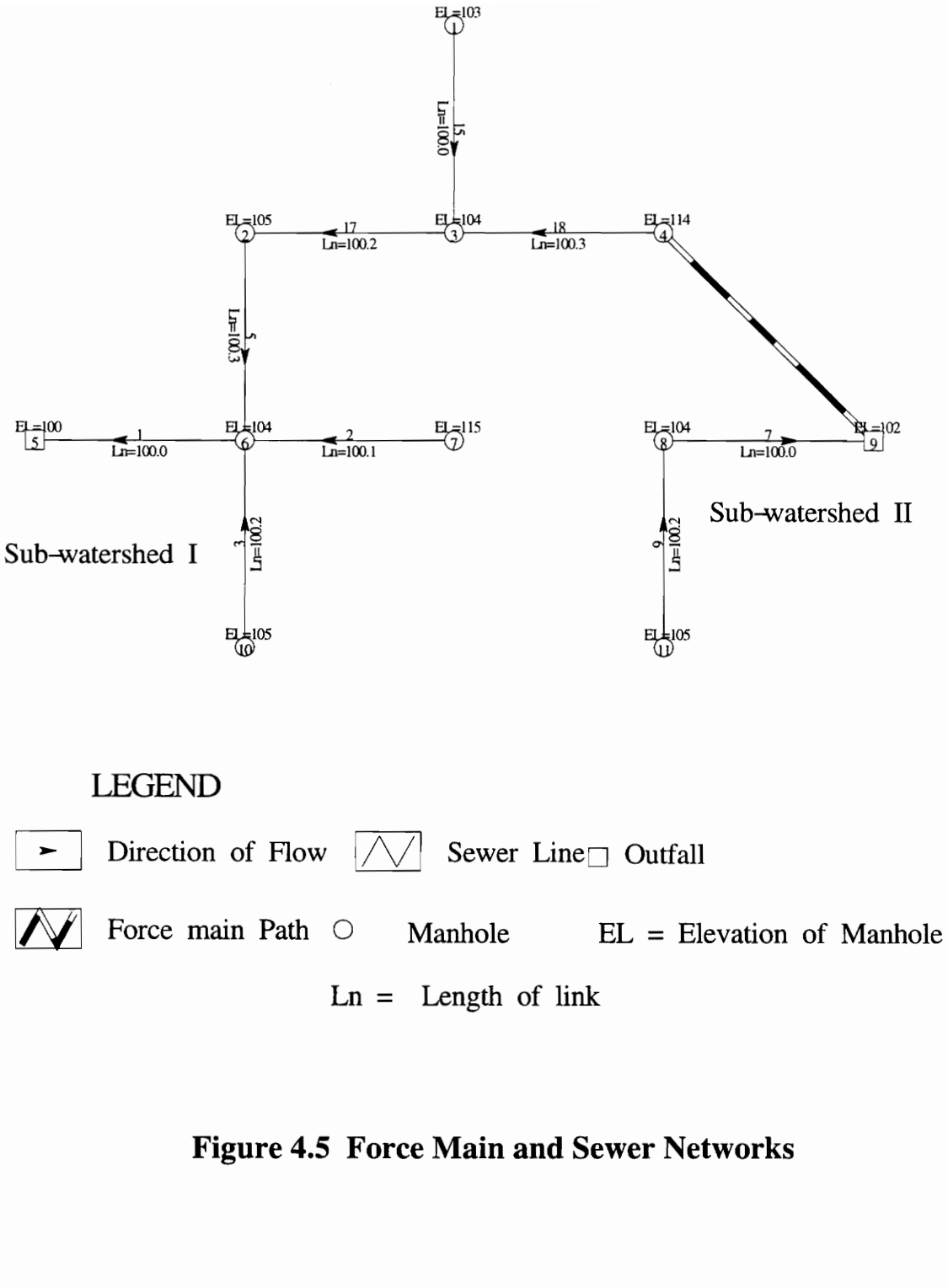


Figure 4.5 Force Main and Sewer Networks

The force main path determination procedure is followed to determine the force main path and destination for each subwatershed that requires a force main. In each case, the force main length is determined. Also because the destination of the force main has been determined, the static head can also be determined by determining the difference in elevation between the outfall of the subwatershed and the manhole at which the force main terminates.

Using the force main length, the friction head caused by the force main pipe can determined using the Darcy-Weisbach formula. The friction head is added to the static head to determine the total head on the pump. The flow from the subwatershed, together with the total head are then used in selection the appropriate pump for that subwatershed.

Chapter 5 - Wet Well Design, Pump Selection, and Network Pipe Design

5.1 Introduction

The procedure used in the design of wet wells and selection of pumps is the subject of this chapter. The information used in the design has been gathered from various literature on wet well design, pump selection, and sewer network pipe design. These source includes information from a wet well design and pump selection outline provided by Vogelsong (1994), and others by Gray et al (1992), Tchobanoglous (1981), and Sanks (1989).

5.2 Wet Well Size Calculations

The wet well size or volume for each subwatershed that requires pumping is determined by the average flow at that subwatershed outfall and the chosen cycle time. The wet well volume is also influenced by the method of pump operation, this may be constant speed or variable speed (Tchobanoglous 1981). If the pump is driven by a variable speed motor which varies the pumping rate to match the inflow of sewage into the wet well, the required size for the wet well will be small. If a constant speed pump is used, then the size of the wet well must be adequate to prevent short cycling of pump. Short cycling of pump refers to frequent starting and stopping of the pump.

According to Tchobanoglous (1981), the required volume of storage for the wet well, V, in gallons is given by:

$$V = \frac{t_c Q_p}{4} \qquad (5.2.1)$$

where t_c is the cycle time in minutes, and Q_p is the pump capacity in gal/min.

Equation 5.2.1 is derived by Tchobanoglous (1981) as follows:

Step 1:

a. The time required to fill the wet well when the pump is off is given by:

$$t_f = \frac{V}{Q_i} \tag{5.2.2}$$

where Q_i is the inflow rate.

b. The time the pump takes to empty the wet well is:

$$t_e = \frac{V}{Q_p - Q_i} \tag{5.2.3}$$

c. The total cycle time is therefore defined by:

$$t_c = t_f + t_e \tag{5.2.4}$$

also the same as:

$$t_c = \frac{V}{Q_i} + \frac{V}{Q_p - Q_i} \tag{5.2.5}$$

Step 2.

To determine the value of inflow, Q_i, into the wet well that will make the cycle time, t_c, minimum,

a. Rearrange the terms in Equation 5.2.5 to get:

$$t_c(Q_iQ_p - Q_i^2) = V(Q_p - Q_i) + VQ_i \tag{5.2.6}$$

leading to:

$$t_c(Q_iQ_p - Q_i^2) = VQ_p \tag{5.2.7}$$

and

$$\frac{V}{t_c} = Q_i - \frac{Q_i^2}{Q_p} \tag{5.2.8}$$

b. To determine the Q_i that will make t_c a minimum and the term V/t_c a maximum, differentiate the term with respect to Q_i and set the resulting derivative to zero:

$$\frac{d(V/t_c)}{dQ_i} = 1 - \frac{2Q_i}{Q_p} = 0 \tag{5.2.9}$$

From equation 5.2.9,

$$Q_p = 2Q_i \tag{5.2.10}$$

c. To verify that V/t_c is a maximum, find the second derivative:

$$\frac{d^2(V/t_c)}{dQ_i^2} = -\frac{2}{Q_p} \tag{5.2.11}$$

Since the second derivative is negative, the term V/t_c is a maximum. This implies that when $Q_p = 2Q_i$, V/t_c is maximum, or for any preselected value of V/t_c the maximum wet well volume required occurs when $Q_i = Q_p / 2$.

Step 3.

Solve for the wet well volume. Substitute $Q_i = Q_p / 2$ in equation 5.2.5 to get:

$$t_c = \frac{V}{Q_p/2} + \frac{V}{Q_p - Q_p/2} \tag{5.2.12}$$

or

$$\frac{t_c Q_p}{2} = 2V \tag{5.2.13}$$

from which we get Equation 5.2.1.

The time to fill is the time it takes for the wet well to fill from the low water level (LWL) to the high water level mark (HWL) and the time to empty is vice versa.

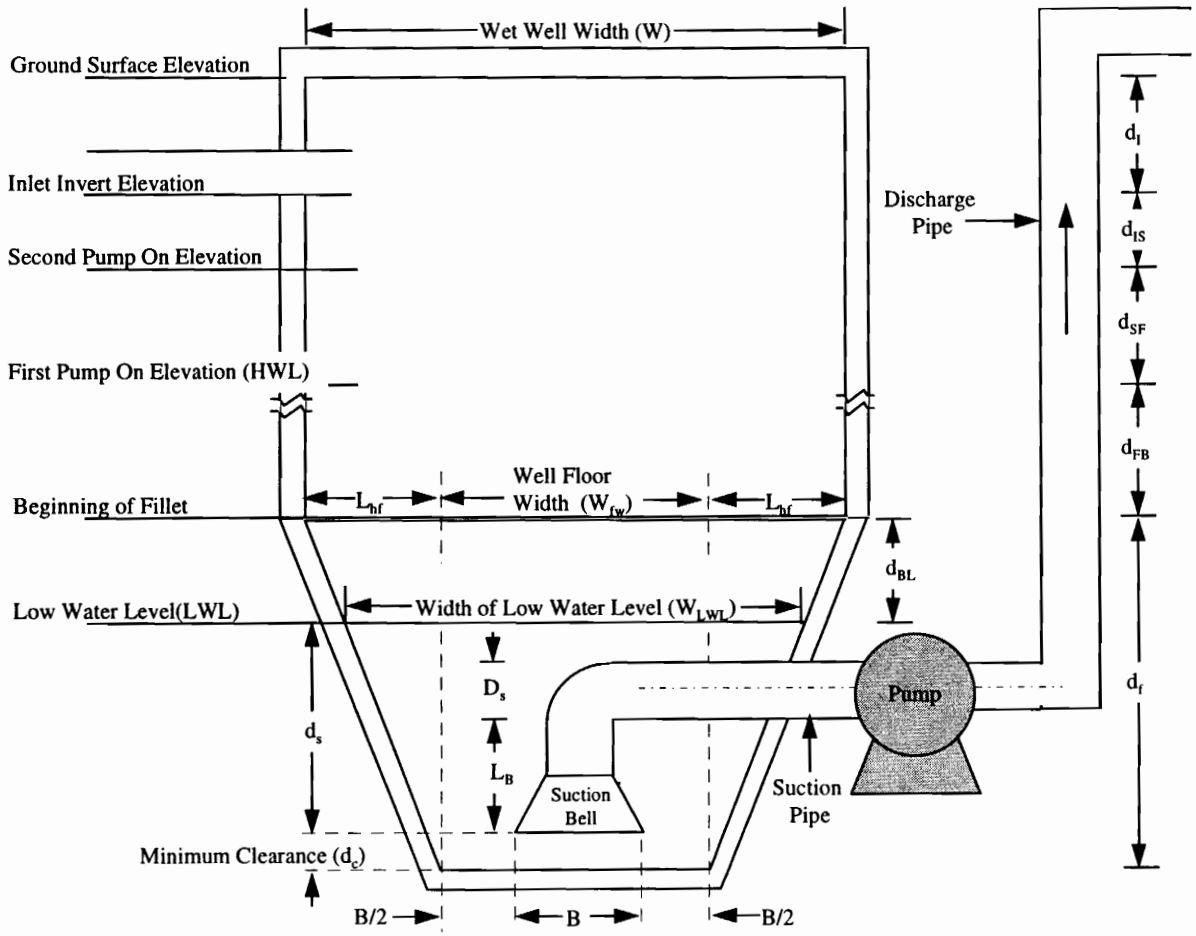


Figure 5.2.1 Wet Well Description

The required volume, V, refers to the volume of liquid contained between the first pump on elevation or HWL and low water elevation or LWL (see Figure 5.2.1). Tchobanoglous (1981) recommends that the cycle time should not be less than 15 minutes for pump motors between 15 and 75 kW (20 and 100 hp), and 20 to 30 minutes for pump motors between 75 and 200 kW. For motors larger than 200 kW, the manufacturer should be contacted for the acceptable range of cycle time.

From Equation 5.2.9, the minimum cycle time for a single pump operation occurs when the inflow Q_i is exactly half the pump capacity, Q_p (Tchobanoglous, 1981). Under this condition, when flows are larger than the average, the pump stays on for a longer period and stays off for a shorter period. When flows are less than the average, the pump stays on for a shorter periods and stays off for longer periods. From Equation 5.2.9, t_e can be express as follows:

$$t_e = \frac{V}{2Q_i - Q_i} \tag{5.2.14}$$

$$t_e = \frac{V}{Q_i} = t_f \tag{5.2.15}$$

This implies that for the minimum cycling time for single speed pump the time to fill and the time to empty are equal. The town of Blacksburg requires a maximum retention time, t_f, of 20 minutes. For the purpose of this research, a t_f of 10 minutes is used. This time was chosen as it would yield a minimum cycle time of 20 minutes thus satisfying both the recommended minimum cycle time by Tchobanoglous for most pump sizes and the town requirements.

For the slope of the sides of the fillet, the state of Virginia Water Control Board Requires minimum slope of 1:1 in order to prevent the accumulation of material (Sewerage, 1977). The required depth of wet well is therefore a variable

parameter which will depend on the volume of water to be collected in the fillet part of the wet well and the vertical part of it (see Figure 5.2.1).

The velocity of flow in the suction pipe should not be less than 2 ft/s or greater than 6 ft/s (obtained from Vogelsong (1994)). Sanks (1989) recommends that suction velocities be below 5 ft/s. The size of the suction pipe is therefore selected to meet these requirements. With the pump capacity Q_p ($Q_p \cong 2 * Q_i$) known, for a selected suction pipe size D_s (in feet), the velocity in the suction pipe, V_s, can be calculated from:

$$V_s = \frac{4Q_p}{\pi D_s^2} \qquad (5.2.16)$$

If the velocity is less than 2 ft/s or greater than 6 ft/s, a different diameter of suction pipe is chosen. The parameters of the flare for the suction, which include: its thickness (T), radius of curvature of the bell mouth (R), its length (L_B), and the diameter of the mouth (B) are determined based on the diameter of the suction pipe (see Figure 5.2.2).

Based on the velocity in the suction pipe, the minimum depth of submergence (d_s) (Figure 5.2.1) of the mouth of the suction pipe flare is determined from the equation:

$$d_s = 0.585*V_s - 0.16 \qquad (5.2.17)$$

Equation 5.2.4 was derived from a chart in Vogelsong (1994) that shows the relationship between the required minimum depth of submergence and the suction velocity. This minimum submergence refers to the length of suction pipe that should always be beneath the water surface. The minimum clearance depth (d_c) (Figure 5.2.1) of the mouth of the suction pipe from the wet well floor is chosen as 3B/8 as suggested by Sanks (1989). Sanks recommends that the clearance between the wet well floor and inlet of the suction pipe be between B/4 and B/2

where B is the diameter of the suction pipe. The value 3B/8 is chosen because it lies between the range of B/4 and B/2 recommended by Sanks (1989).

The ASCE/WPCF manual recommends that the distance from the edge of the suction bell to the foot of the wet well fillet (see Figure 5.2.1) be B/2. This leads to the wet well floor width, W_{fw} defined as:

$$W_{fw} = 2 * B \qquad (5.2.18)$$

The horizontal distance from the foot of the fillet to the vertical wall of the wet well, L_{hf} is:

$$L_{hf} = (W - W_{fw}) / 2 \qquad (5.2.19)$$

and the depth of the fillet part of the wet well, d_f is:

$$d_f = L_{hf} * Slope \qquad (5.2.20)$$

The volume of water contained from the beginning of the fillet to the wet well floor, Vol_f is:

$$Vol_f = [(W_{fw} + W) / 2 * d_f] * L \qquad (5.2.21)$$

The width of the low water level, W_{LWL} is:

$$W_{LWL} = W - 2 * (d_s + d_c) / Slope \qquad (5.2.22)$$

The volume of water contained between the low water level and the wet well floor, Vol_{LWL}, is:

$$Vol_{LWL} = [((W_{LWL} + W_{fw}) / 2) * (d_s + d_c)] * L \qquad (5.2.23)$$

The volume between the beginning of fillet and the low water level, Vol_{BL}, is:

$$Vol_{BL} = Vol_f - Vol_{LWL} \qquad (5.2.24)$$

and the depth between the beginning of fillet and the low water level, d_{BL}, is:

$$d_{BL} = d_f - (d_s + d_c) \qquad (5.2.25)$$

Also, the volume contained between the beginning of the fillet and the first pump on elevation or the high water level mark, Vol_{FB}, is:

$$Vol_{FB} = V - Vol_{BL} \qquad (5.2.26)$$

from which the depth between the high water level and the beginning of fillet, d_{FB}, can be derived as:

$$d_{FB} = Vol_{FB} / (L * W) \qquad (5.2.27)$$

Other depths making up the total wet well depth are defined as:

d_{SF} = depth between the second pump on elevation and the first pump elevation.

d_{IS} = depth between the inlet invert elevation and the second pump on elevation

d_I = depth of the inlet invert from the ground surface

The wet well depth, d_w is therefore defined as:

$$d_w = d_I + d_{IS} + d_{SF} + d_{FB} + d_{BL} + d_s + d_c \qquad (5.2.28)$$

5.3 Pump Selection

In selecting the pump for each subwatershed, the main goal is to maximize pump efficiency and minimize the power required by the pump. The specified total dynamic head H_T and the flow rate Q_p are used as the duty required (DR) for selecting the pump. A pump will operate at its best efficiency if it is operating at its design point (Douglas et al., 1979). The design point is uniquely defined by the type number. Two type numbers are used in selecting the pump: required type number (n_{sr}) (also known as the specific speed) and pump type number (n_s). The required type number is defined by:

$$n_{sr} = \frac{NQ_p^{0.5}}{(gH_T)^{0.75}} \tag{5.3.1}$$

where N is the nominal speed of the pump in revolutions per second; g is the acceleration due to gravity in ft/s^2; Q_p is the pump capacity or the required pumping rate in ft^3/s; and H_T is the total dynamic head in ft.

The pump type number is defined by:

$$n_s = \frac{K_Q^{0.5}}{K_H^{0.75}} \tag{5.3.2}$$

where K_Q and K_H represent the flow coefficient and the head coefficients respectively at the maximum efficiency point of the pump. The coefficients are defined as:

$$K_Q = \frac{Q_p}{ND_{imp}^3} \tag{5.3.3}$$

and

$$K_H = \frac{gH_T}{N^2 D_{imp}^2} \tag{5.3.4}$$

where D_{imp} is the impeller diameter.

For selecting the appropriate pump, $n_s \approx n_{sr}$.

5.3.1 Determination of the Total Dynamic Head

The total dynamic head is defined by a combination of the static head, head losses due to friction in the suction and discharge pipes, minor losses in pipes, and the velocity heads in the suction and discharge pipes. The static suction head, h_s, is the difference in elevation between the suction liquid level (when liquid level is at minimum depth of submergence) and the centerline of the pump impeller. Referring to Figure 5.2.1, it can be defined as

$$h_s = d_s - L_B - D_s/2 \tag{5.3.5}$$

The static discharge head, h_d is the difference in elevation between the maximum discharge liquid level and the centerline of the pump impeller, defined from Figure 5.2.1 as

$$h_d = d_s - D_s/2 - L_B + d_{BL} + d_{FB} + d_{SF} + d_{IS} + d_I + \Delta E \tag{5.3.6}$$

where ΔE is the difference in elevation between the ground surface elevation at the wet well and the point of discharge of the force main.

The total dynamic head, H_T is defined as

$$H_T = h_{stat} + h_{fs} + \Sigma h_{ms} + h_{fd} + \Sigma h_{ms} + \frac{V_d^2}{2g} \tag{5.3.7}$$

where H_T is the total dynamic head,

h_{stat} is the total static head defined as $h_d - h_s$

h_{fs} is the friction loss on the suction side of the pump;

h_{fd} is the friction loss on the discharge side of the pump;

h_{ms} is the minor losses on the suction side of the pump;

h_{md} is the minor losses on the discharge side of the pump;

V_d is the flow velocity in the discharge pipe.

In this research, the pump selection is based on a procedure described by Douglas et al. (1979). In the procedure, three or four pairs of K_Q, and K_H values, and their corresponding efficiencies are stored for each pump in a data file. The K_H and K_Q are related by the parabolic equation

$$K_H = C - B*(K_Q) - A*(K_Q)^2, \quad (5.3.8)$$

where A, B, and C are constant coefficients. Using the pairs of the K_Q and K_H values, the coefficients A, B and C are determined for each pump through a multiple regression procedure. These coefficients are used later to calculate the K_H for any chosen K_Q. A quadratic relation is also developed between the K_Q value and the pump efficiencies so that for any chosen K_Q, the corresponding efficiency of the pump can be determined.

The system curves used for this project were curves for a Gorman-Rupp pump included in Vogelsong (1994). The curves are for the same pump operating at speeds ranging from 650 rpm to 1550 rpm. The impeller diameter is 12.375 inches. For each pump speed, the K_Q and K_H values were generated. For each of these curves, the pump type numbers are calculated and stored in a pump data file for later use. Also, from the duty required, the required type number is calculated for each speed.

In deciding which pump to select, the goal is to select the pump such that the pump type number, n_s, is close to or equals the required type number, n_{sr}. If a match can not be achieved for a particular pump, it means that the pump will not operate at its maximum efficiency. In order to determine the pump type number n_s' that will match the required type number, the K_Q value is varied by an iteration procedure. In each case, the corresponding K_H is determined from the parabolic equation relating K_Q and K_H. The n_s' is then calculated from both K_Q and K_H. This new type number is then compared with the required type number. The type

number n_s' closest to the required type number is chosen and the efficiency of the pump at type number n_s' is then calculated. The power P, required for the pump is calculated by:

$$P = \frac{\gamma Q_p H_T}{\eta} \tag{5.3.9}$$

where η is the pump efficiency and γ is the specific gravity of the wastewater. The procedure is repeated for all the other pumps in the pump data file. The pump with least power requirement is chosen as the pump for the subwatershed.

5.4 Design of pipes in each subwatershed

The design of pipes is accomplished using the GSDPM3, the gravity sewer design program developed by Gray et al (1992). GSDPM3 uses the Manning's equation to determine the require diameter for each pipe and also calculates the required slope of that pipe.

5.4.1 Input Data for GSDPM3.

The information that is input into the GSDPM3 program includes: street elevations of manhole locations, pipe lengths, Manning's coefficient, maximum allowable infiltration, type of material (PVC, Vitrified Clay, or Reinforced concrete), Means Cost Index, year of design, number of pipes in the system, peak factor, number of connections to each pipe, number of persons per connection, and method of aligning of pipes across inverts. Also additional design information such as the Manning's n, the maximum allowable infiltration, required minimum depth of cover and the type of pipe material. The manual for GSDPM3 (See Appendix A) gives further details on the input parameters.

5.4.2 Summary of Design Steps in GSDPM3.

Set i = 1. For i = 1 to n. Where n is the total number of pipes in the system,

1. Determine ground slope for each of the pipes.
2. Determine peak flow from the number of direct connections, indirect connections, and flow from pipes upstream of pipe i.
3. Determine size of pipe i using Manning's equation.
4. If the absolute value of the difference between the ground slope and the minimum allowable slope for pipe i is less than or equal to 0.0001, then goto 6 else goto 5.

5. Optimize pipe slope (Choose pipe slope that is close to the minimum allowable slope for pipe i while meeting the minimum depth of cover requirement specified by local authorities).

6. Calculate pipe capacity and corresponding pipe diameter, Dcap.

7. At peak flow, is the depth of flow less than or equal to 0.7(Dcap)? If yes, then goto 8. Else redo the pipe size. Goto 4.

8. Determine drop at upper manhole and invert elevation at upper manhole of pipe i. If the drop is greater than 2 ft, recommend using drop stack manhole (Figure 5.4.1 shows an illustration of a drop stack manhole).

9. Calculate invert elevation at lower manhole of pipe i.

10. Is i = n? if yes then goto 12. Else goto 11.

11. i = i + 1, goto 1.

12. Print results

13. End

The GSDPM3 program produces the diameters for each of the pipes, their associated slopes, invert elevations at each manhole, the depth of each manhole. Also, GSDPM3 provides and economic report which details the cost of manholes, pipes, and trenching in each subwatershed. The minimum pipe diameter used by GSDPM3 is 8 inches.

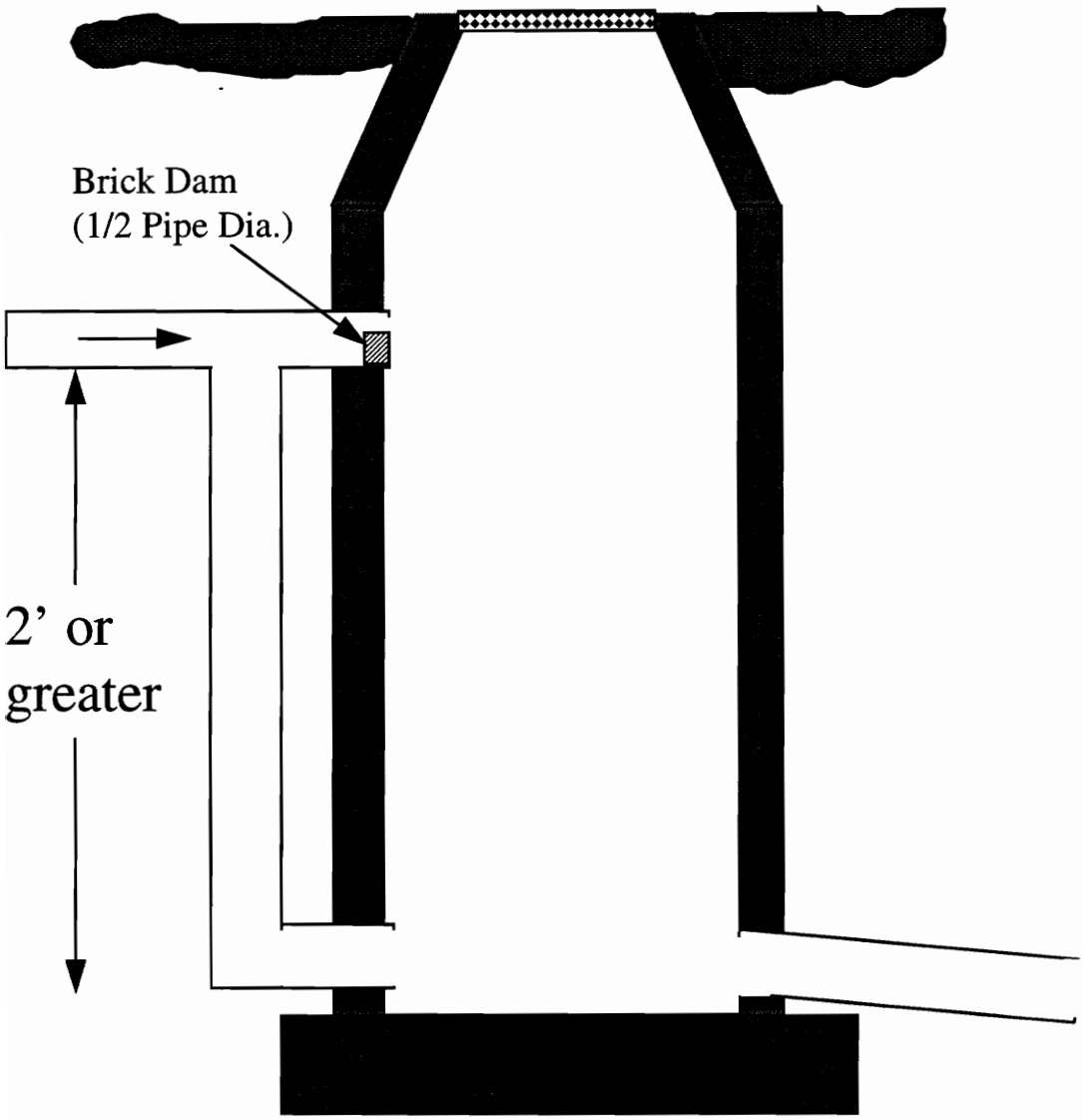


Figure 5.4.1- Illustration of Drop Stack Manhole

Chapter 6 - Integration of Design Components and Determination of Costs.

6.1 Introduction

This chapter addresses the manner and order in which the various components of the design procedure are brought together to accomplish the overall design. In this chapter, the relationship between the network layout selection, pipe design, pump selection and wet well design are highlighted. Also the procedure used in the calculation of the cost of the various components in the system design is discussed. Appendix D contains the code listings for the various FORTRAN, C, and ARC Macro Language (AML) programs.

6.2 Integration of Components

6.2.1 Renumbering of Subwatersheds

In the sewer design program GSDPM3, because pipes in upstream subwatersheds have to be designed before those in the downstream subwatersheds, the subwatersheds have to be renumbered so that the upstream subwatersheds occur before the downstream ones. The subwatersheds are numbered WP(1),WP(2),...,WP(N) as follows:

1. The most upstream subwatershed is allocated the number 1 (m=1).
2. Proceed downstream numbering subwatersheds consecutively.
3. When a junction is encountered, check if there are any unnumbered subwatersheds upstream of it. If there exists some unnumbered subwatersheds, follow the unnumbered path to its most upstream subwatershed and start consecutive numbering downstream until return to junction is established. If any other unnumbered path from the same junction exists, repeat step 3. Else go to step

2. At the end of the numbering procedure, the highest number which is N should be the subwatershed with the main outfall, WP(N).

6.2.2 Integration of the Design Components

The overall design procedure after the network has been generated is outlined in the flowchart (see Figure 6.2.1). In Figure 6.2.1, the following steps are performed. The subwatershed delineation, the network selection, and pump station location are determined. The sewer design program then starts with the most upstream subwatershed with its number m = 1. The flow for each pipe in subwatershed WP(m) is estimated based on the number of connections to the upstream manhole of that pipe. The flow estimates, together with the elevations of manhole locations and pipe lengths are passed on to GSDPM3 for the design of pipe diameters and slopes. If m = N, all the subwatersheds have been considered and the procedure ends. Else, the downstream subwatershed WP(k) into which WP(m) drains is determined. The total flow from the outlet of WP(m) is transferred to an optimal manhole in WP(k) through the force main. The wet well and force main are designed for WP(m). Also the appropriate pump size is selected for WP(m).

The variable m is then set to m = m+1, and the design is performed for the new subwatershed WP(m). The process continues until the pipes have been designed for all the subwatersheds m=1 to m=N.

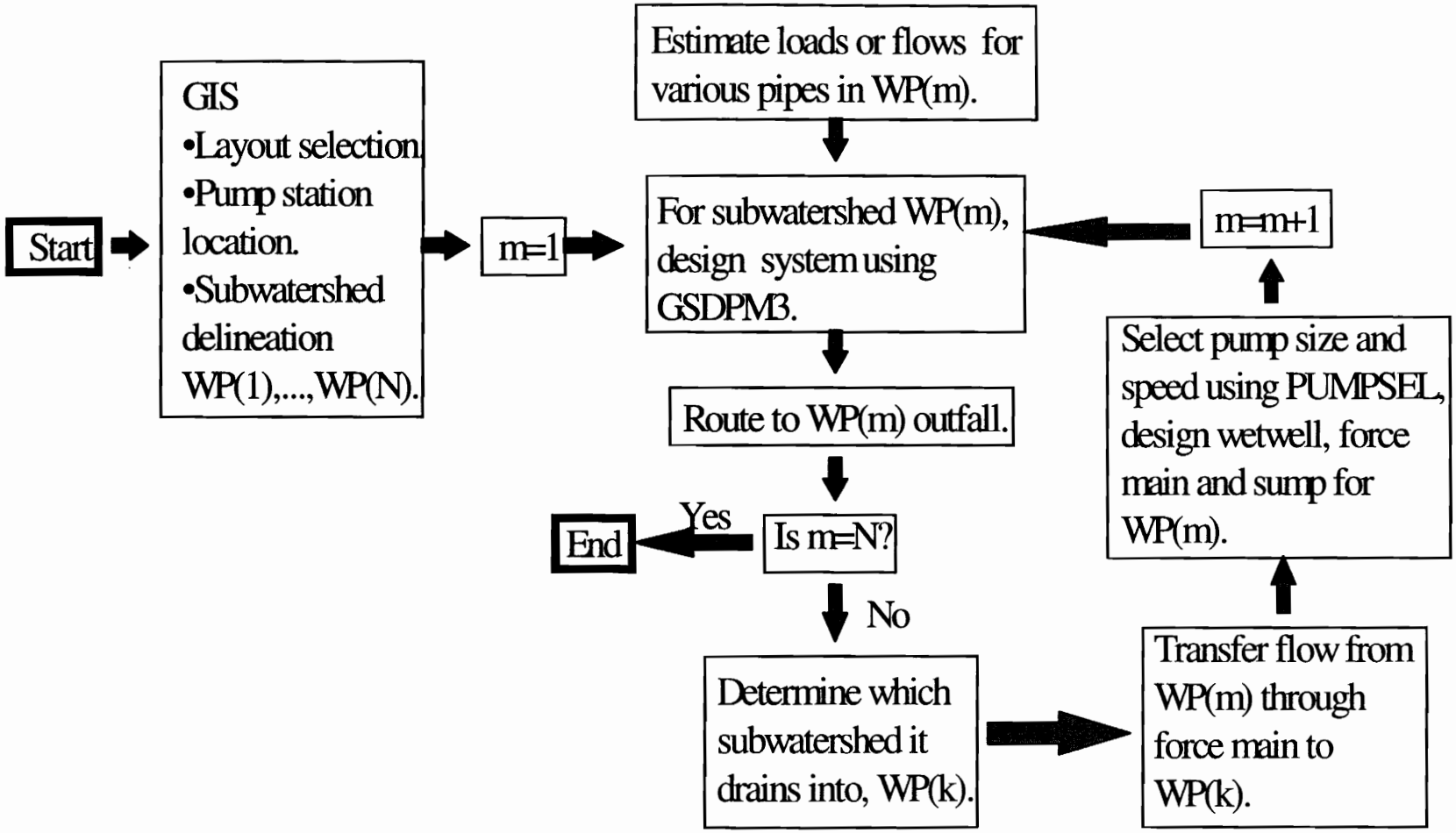


Figure 6.2.1. Flow Chart For Design Procedure.

6.3 Cost Determination

For a comparison of the different alternatives, system design costs are estimated based on the 1995 Means Cost Index. See Appendix A for a further discussion on the Means Cost Index. The costs are determined for the force mains, wet wells, pumps and motors, and networks in each subwatershed.

6.3.1 Force Main Costs

The costs associated with the force main installation are the trenching costs, backfilling, and pipe installation costs. For example, if the soil type is heavy soil or clay, the trenching costs based on the 1995 Means Index is $42 per cu. yard for trenching up to 6 ft and $55 per cu yard for trenching up to 12 ft deep. The total trenching cost is the volume of earth removed multiplied by the unit cost per cubic yard depending on the trench depth.

Pipe bedding costs are estimated based on the class C type bedding as specified in the ASCE/WPCF sewer design manual (Design, 1970). Two types of materials are used when backfilling according to the Class C option (see Figure 6.3.1). These are concrete and granular material. The cost of backfilling is the sum of the volume of concrete used multiplied by the unit cost per cubic yard of concrete plus volume of granular material used multiplied by the unit cost of granular material per cubic yard.

The pipe cost is estimated based on an equation developed from a regression analysis of the costs in the Means Index. The equation for the pipe cost pc is:

$$pc = flength * 0.0611 * D^2 - 0.2047 * D + 2.2088, \quad (6.3.1)$$

where D is the force main diameter and flength is the total force main length. This cost includes the cost of the pipe, together with its installation cost. The constants 0.0611, -0.2047, and 2.2088 were determined by regression based on the various

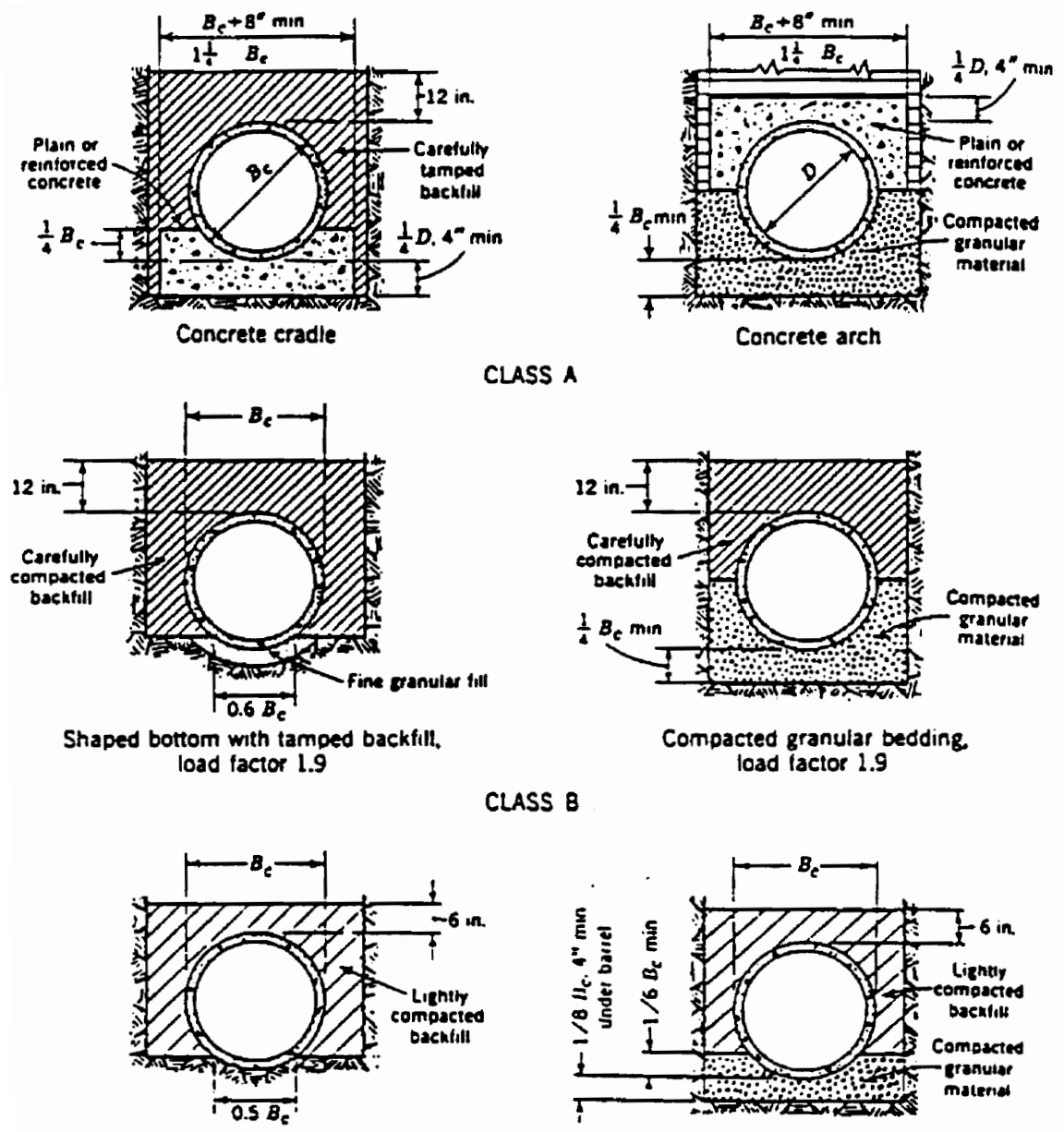


Figure 6.3.1- Trench Bedding Classes

Source: ASCE/WPCF Sewer Design Manual (Design, 1970)

pipe diameters and their corresponding pipe cost. Equation 6.3.1 was derived for use by the program to interpolate pipe costs given the pipe diameters.

6.3.2 Wet Well Costs

The wet well cost is determined based on its capacity. An equation generated based on costs from the means index is used to determine the price for the wet well. The cost of the wet well wc is defined as:

$$wc = cap * 5.1282 + 1615.4, \quad (6.3.2)$$

where cap is the capacity of the wet well in gallons.

6.3.3 Pump Costs

The cost associated with the pump are: the capital cost of the pump, the cost of the motor and the cost of operating the pump. The cost of operating the pump is reflected in the power being consumed by the pump. The capital cost of the pump, Pcap, is determined by the equation:

$$Pcap = Q_p * 49.625 + 37375, \quad (6.3.3)$$

where Q_p is the capacity of the pump in gallons per minute.

The cost of the motor, cm, is also given by:

$$cm = -0.0044 * P^3 + 1.3369 * P^2 + 18.608 * P + 770.55, \quad (6.3.4)$$

where P is the power required by the pump in horse power (hp).

6.3.4 Cost of Pipe Networks in Each Subwatershed

The costs of installation of pipes and manholes in each subwatershed is calculated by the GSDPM3 which based the costs on the 1991 Means Costs Index.

Chapter 7 - Application

7.1 Introduction

The sewer design program is applied to design sewer systems in two sections of Blacksburg.

7.2 Test Area 1

7.2.1 Problem Description

The first section is bounded to the north by Route 460 Bypass, to the south by Price's Fork Road, to the east by Main Street, and to the West by Tom's Creek Road. Zoning in this section includes: high and medium density multi-unit residential apartments and some as restricted single unit apartments. The terrain in this area is quite hilly with elevations ranging from 2096 ft to 2251 ft. The total area is about 0.46 sq. mi. Figure 7.2.1 shows the topography and the layout of the streets in the area. The zoning distribution can be seen in Figure 7.2.2. Figure 7.2.3 shows property lots and existing and future locations of buildings. The goal is to determine the required sewer network layout for the area and design the pipes in the network. Figure 7.2.4 contains the desired locations for manholes in the area and Table 7.2.1 contains the number of direct connection to each manhole and also an indication of the desired outlet or outfall for the whole system. In Table 7.2.1, the desired outfall or outlet has an OUT attribute value of 1. All the other manholes have their OUT attribute values = 0. Figure 7.2.5 also indicates the various numbers of the manholes. The chosen outlet is manhole number 145 (see Figure 7.2.5).

The sewer design is based on the following conditions:

Number of person per connection or household = 3.5

Flow per person = 100 gallons per day

Peak factor = 2.5

Allowable infiltration = 500 gpd per inch diameter per mile

Maximum allowable depth of manhole = 15 ft

Minimum depth of cover = 3 ft.

Manning's n = 0.013.

According to standards set by the town of Blacksburg, the minimum sewer pipe diameter is 8 inches.

According to the standards of the WPCF/ASCE annual, manhole spacing should not be more than 400 ft.

7.2.2 Results and Discussion:

Figure 7.2.6 shows the network generated from the 145 manhole locations which resulted in two subwatersheds that are connected by a force main. Subwatershed 2, is only comprised of 5 manholes. From the output of the run which is summarized on Table 7.2.2, the user can see the head on the pump in subwatershed 2. The value of the head on the pump is 6.43 ft which is far below the set allowable maximum manhole depth limit. From its value, it is decided that a pump is not necessary. A gravity connection would suffice to link the 5 isolated manholes to subwatershed 1. An investigation of this possibility can be done by taking a look at the force main path output file (see Table 7.2.3).

A close examination of the force main path indicates the path of the force main is following a downward slope from the source to the destination. In the table, the column 'point' refers to either manholes or just ordinary points that make up the force main path. For example, the force main is leading from manhole 31 to 57 hence those two are at the start and finishing points. The points

labeled '0' are just one of either the grid points or the buffer vertices that are used in the force main path determination. They are not manholes but are just intermediary points between the source of the force main to its destination. As can be seen from the table, the elevation along the force main path goes from one of higher elevation to a lower elevation. This indicates that it may be possible to drain subwatershed 2 to subwatershed 1 via a gravity connection. If this is the case, why does the program indicate that a pump is needed at that particular location?

The answer to the above question is that, the program placed a pump at that location because in order to connect the manholes in subwatershed 2 to subwatershed 1 by gravity connection, the 15 ft maximum depth of manhole would be violated. The depth violation results from the fact that the positions where the manholes were initially placed resulted in a network in which gravity connection is not possible without the 15 ft depth violation.

To investigate the possibility of using a gravity connection, another run of the program is made, this time with two additional manholes placed along the path of the force main (see Figure 7.2.7). This results in the program being able to design a gravity sewer network for the whole area with (see Figure 7.2.8). Table 7.2.4 contains an excerpt form the GSDPM3 output file. This table lists the design parameters and summaries of the gravity sewer design by GSDPM3. Table 7.2.5 also lists the actual dimensions of the various design parameters. It lists the various pipe diameters, slopes, depths of cover, pipe capacities etc.

The results above indicate that the program results very much depend on the users location of the manholes. Inappropriately placed manholes can result in the program indicating the need of pumps. Fortunately, the force main path determination part of the program helps to detect those cases and gravity sewers

may be adopted with suitable manhole relocations or additions. Manholes can therefore placed along the force main path and the program rerun.

Table 7.2.1 Desired Manholes and Their Attributes

MH#	OUT	CON
1	0	36
2	0	5
3	0	5
4	0	36
5	0	36
6	0	5
7	0	5
8	0	36
9	0	36
10	0	60
11	0	60
12	0	36
13	0	36
14	0	36
15	0	5
16	0	36
17	0	5
18	0	5
19	0	5
20	0	5
21	0	5
22	0	36
23	0	5
24	0	60
25	0	36
26	0	5
27	0	60
28	0	36
29	0	60
30	0	60
31	1	5
32	0	5
33	0	36
34	0	5
35	0	5
36	0	5

MH#	OUT	CON
37	0	5
38	0	5
39	0	5
40	0	36
41	0	5
42	0	60
43	0	5
44	0	5
45	0	5
46	0	36
47	0	5
48	0	5
49	0	5
50	0	20
51	0	5
52	0	5
53	0	5
54	0	5
55	0	5
56	0	5
57	0	5
58	0	5
59	0	5
60	0	5
61	0	5
62	0	5
63	0	5
64	0	5
65	0	5
66	0	5
67	0	5
68	0	5
69	0	5
70	0	5
71	0	5
72	0	5

MH#	OUT	CON
73	0	5
74	0	5
75	0	5
76	0	5
77	0	5
78	0	5
79	0	5
80	0	5
81	0	5
82	0	5
83	0	5
84	0	5
85	0	5
86	0	5
87	0	20
88	0	5
89	0	5
90	0	5
91	0	5
92	0	5
93	0	5
94	0	5
95	0	5
96	0	5
97	0	5
98	0	5
99	0	5
100	0	5
101	0	5
102	0	5
103	0	100
104	0	5
105	0	5
106	0	5
107	0	5
108	0	5

MH#	OUT	CON
109	0	5
110	0	5
111	0	36
112	0	5
113	0	36
114	0	36
115	0	5
116	0	5
117	0	5
118	0	5
119	0	36
120	0	5
121	0	5
122	0	5
123	0	5
124	0	5
125	0	5
126	0	36
127	0	5
128	0	36
129	0	5
130	0	5
131	0	36
132	0	5
133	0	36
134	0	5
135	0	36
136	0	5
137	0	5
138	0	5
139	0	5
140	0	5
141	0	5
142	0	5
143	0	5
144	0	5
145	1	5

Table 7.2.2 Summarized Output File

```
***********   Begin connectivity #1   ***********

For subwatershed connectivity #1, the general costs are summarized
 as follows:

For subwatershed 1
Costs of pipes and excavation in subwatershed = $  484816.12
For subwatershed[2]
Waste from subwatershed 2 is pumped to manhole #57 in subwatershed 1
Force main trenching cost                      = $   11331.89
Force main Bedding cost                        = $     371.59
Force main pipe installation cost              = $    1579.57
Force main backfill cost                       = $    3641.37
Pump power consumption cost(per year)          = $      98.52
Pump cost                                      = $   30400.64
Pump Motor cost                                = $     606.70
Wetwell cost                                   = $    1375.51
Costs of pipes and excavation in subwatershed = $   62164.29

 Total cost for subwatershed 2 is $  111570.09

Total system cost for the 2 subwatersheds is $  596386.25
 Out of the 1 different connectivities, connectivity #1 produces the
Least Cost Overall Design

For subwatershed[1]
Total Flow at Outfall  =     927.66 gal/min

For subwatershed[2]
Waste from subwatershed 2 is pumped to manhole #57 in subwatershed 1
Force main Size         =       6.00 inches
Force main Length       =     631.92 ft
```

Wetwell Capacity	=	14.04 ft^3
Total Flow at Outfall	=	26.25 gal/min
Pump power required	=	0.07 hp
Head on pump	=	6.43 ft

Table 7.2.3 Output of force main path.

Point	X-Coord (ft)	Y-Coord (ft)	Elevation (ft)
31	11220825.3	3716128.93	2188.17
0	11220774.7	3715896.40	2184.53
0	11220774.7	3715628.48	2179.31
57	11220799.2	3715487.35	2178.57

Table 7.2.4 Design Parameters

MANNINGS ROUGHNESS COEFFICIENT	=	.013
MAXIMUM INFILTRATION	=	500.0 G/IN.DIA./MI./DAY
NUMBER OF PIPES IN SYSTEM	=	147
DESIRED INVERT AT OUTLET OF SYSTEM	=	2038.63 FT
MINIMUM ALLOWABLE DEPTH OF COVER	=	3.0 FT
NUMBER OF PERSONS PER SERVICE CONNECTION	=	2.7
AVERAGE SEWAGE FLOW PER PERSON	=	100.0 GPD
SEWAGE FLOW PEAK FACTOR	=	2.0

MANHOLE DROPS ARE FOUND BY ALIGNING CROWN EL.

PIPE SLOPES ARE DESIGNED ACCORDING TO TEN STATES STANDARDS.

PRELIMINARY ECONOMIC REPORT *
(MINIMUM INVERT DESIGN) *

PIPE SIZE (IN)	LENGTH (FT)	INSTALLED PIPE COST ($)	MATERIAL TYPE
8.0	19428.49	175853.06	PVC SDR-35
10.0	3188.62	28587.66	PVC SDR-35
18.0	3498.15	125666.49	PVC SDR-35
TOTAL=	26115.26	$ 330107.22	

MH DEPTH (FT)	NUMBER	INSTALLED MH COST ($)
0- 6	131	133063.33
7- 8	2	3054.65
8- 9	1	2536.21
9-10	2	3966.93
10-11	1	2135.85
11-12	3	8596.85
12-13	3	10340.94
13-14	1	3028.72
14-15	2	7275.01
15-16	2	9275.87

```
   ----------------------------------------
      TOTAL=     148     $ 183274.34
      TOTAL INSTALLED COST = $ 513381.56

DIRECTLY CONNECTED POPULATION (PERSONS) = 5343.30
COST PER DIRECTLY CONNECTED PERSON ($) =   96.08

NUMBER OF DIRECT CONNECTIONS            = 1979.00
COST PER DIRECT CONNECTION ($)          =  259.41
COST PER FOOT OF PIPE ($/LF)            =   19.66
FEET OF PIPE PER DIR. CONNECTION (LF)   =    13.2
AVERAGE PIPE SIZE (IN)                  =     9.6
TOTAL EXCAVATION (CU YD)                =  25287.
AVERAGE INVERT DEPTH (FT)               =    4.91
NOTE: COST CALCULATIONS ARE FOR R.S. MEANS
HISTORICAL COST INDEX OF  221.6 FOR THE YEAR 1995.
AND THE CITY COST INDEX OF   78.0
```

Table 7.2.5 GSDPM3 Results

PIPE NUM.	M.H. NUMBER Upp.	Low.	PIPE DIA. (IN)	PIPE LENGTH (FT)	SLOPE (FT/FT)	COVER Upp. (FT)	Low. (FT)	GROUND EL. Upp. (FT)	Low. (FT)	INVERT EL. Upp. (FT)	Low. (FT)	DESIGN FLOW (CFS)	PIPE CAP. (CFS)	ACTUAL VEL. (FPS)	FULL VEL. (FPS)	ACTUAL DEPTH (IN)	INFILT. (CFS)
1	143	145	8	92.9	0.104	3.0	3.0	2068	2059	2065	2055	0.0043	3.92	1.6	11.2	0.16	0.0001
2	145	146	8	121.8	0.057	3.0	3.0	2059	2052	2055	2048	0.01	2.9	2	**	0.35	0.0001
3	141	142	8	53.3	0.089	3.0	3.0	2066	2061	2062	2058	0.0042	3.62	1.4	10.3	0.16	6E-05
4	142	144	8	146.7	0.047	3.0	3.0	2061	2055	2058	2051	0.01	2.63	1.8	**	0.35	0.0002
5	139	136	8	42.7	0.056	3.0	3.0	2060	2057	2056	2054	0.0042	2.87	1.1	8.2	0.16	5E-05
6	136	140	8	100.7	0.011	3.0	3.0	2057	2056	2054	2052	0.01	1.27	1.1	**	0.48	0.0001
7	128	130	8	170.4	0.079	3.0	3.0	2085	2071	2081	2068	0.0303	3.4	3.1	9.7	0.55	0.0002
8	130	132	8	88.7	0.135	3.0	3.0	2071	2059	2068	2056	0.06	4.45	4.5	**	0.66	0.0001
9	113	116	8	148.9	0.114	3.0	3.0	2101	2084	2097	2080	0.0303	4.09	3.4	11.7	0.48	0.0002
10	116	119	8	145.2	0.096	3.0	3.0	2084	2070	2080	2066	0.0603	3.76	4	10.8	0.7	0.0002
11	119	121	8	104.2	0.111	3.0	3.0	2070	2058	2066	2054	0.0645	4.04	4.3	11.6	0.7	0.0001
12	121	123	8	135.2	0.035	3.0	3.0	2058	2054	2054	2050	0.0946	2.26	3.2	6.5	1.13	0.0002
13	91	87	18	142.5	0.001	3.0	3.0	2103	2103	2099	2099	0.0046	4.04	0.3	2.3	0.36	0.0004
14	61	68	8	158.4	0.075	3.0	3.0	2144	2132	2141	2129	0.0044	3.33	1.3	9.5	0.16	0.0002
15	68	73	8	183.6	0.076	3.0	3.0	2132	2118	2129	2115	0.0086	3.34	1.8	9.6	0.25	0.0002
16	73	82	8	217.2	0.030	3.0	3.0	2118	2112	2115	2108	0.0128	2.11	1.6	6	0.42	0.0003
17	82	87	8	204.7	0.043	3.0	3.0	2112	2103	2108	2099	0.017	2.51	2.1	7.2	0.48	0.0002
18	87	92	18	260.2	0.063	3.0	3.0	2103	2087	2099	2082	0.0258	26.5	2.1	15	0.36	0.0007
19	45	38	8	169.5	0.005	3.0	3.0	2163	2162	2159	2158	0.0044	0.84	0.6	2.4	0.42	0.0002
20	22	29	10	148.1	0.003	3.0	3.0	2172	2172	2168	2168	0.0303	1.22	0.9	2.2	1.09	0.0002
21	29	38	10	180.4	0.053	3.0	3.0	2172	2162	2168	2158	0.0805	5.06	3.4	9.3	0.88	0.0003
22	38	33	10	230.9	0.029	3.0	3.0	2162	2155	2158	2151	0.0889	3.77	2.9	6.9	1.06	0.0003
23	33	30	10	182.7	0.066	3.0	3.0	2155	2143	2151	2139	0.1189	5.65	4.2	10.3	1.01	0.0003
24	16	13	8	191.2	0.025	3.0	3.0	2169	2164	2165	2160	0.0303	1.92	2	5.5	0.7	0.0002
25	13	18	10	141.8	0.003	3.0	5.4	2164	2166	2160	2160	0.0604	1.16	1.1	2.1	1.53	0.0002
26	18	24	10	157.9	0.079	5.4	3.0	2166	2151	2160	2147	0.0646	6.19	3.6	11.3	0.69	0.0002
27	24	30	10	184.6	0.044	3.0	3.0	2151	2143	2147	2139	0.1147	4.63	3.6	8.5	1.09	0.0003
28	40	30	8	205.7	0.018	3.0	3.0	2147	2143	2143	2139	0.0303	1.63	1.8	4.6	0.74	0.0002
29	30	32	10	359.5	0.003	3.0	4.1	2143	2143	2139	2138	0.3139	1.16	1.8	2.1	3.54	0.0005

PIPE NUM.	M.H. NUMBER Upp.	Low.	PIPE DIA. (IN)	PIPE LENGTH (FT)	SLOPE (FT/FT)	COVER Upp. (FT)	Low. (FT)	GROUND EL. Upp. (FT)	Low. (FT)	INVERT EL. Upp. (FT)	Low. (FT)	DESIGN FLOW (CFS)	PIPE CAP. (CFS)	ACTUAL VEL. (FPS)	FULL VEL. (FPS)	ACTUAL DEPTH (IN)	INFILT. (CFS)
30	32	37	10	171.4	0.079	4.1	3.0	2143	2128	2138	2125	0.3178	6.19	5.9	11.3	1.53	0.0003
31	4	2	8	164.9	0.031	3.0	3.0	2175	2170	2172	2167	0.03	2.15	2.2	**	0.66	0.0002
32	1	2	8	213.4	0.004	3.0	8.7	2165	2170	2162	2161	0.0303	0.77	1.1	2.2	1.08	0.0003
33	2	3	8	166.6	0.004	8.7	11.1	2170	2172	2161	2160	0.0645	0.77	1.3	2.2	1.58	0.0002
34	3	5	8	181.8	0.028	11.1	3.0	2172	2159	2160	2155	0.0687	2.02	2.7	5.8	1.01	0.0002
35	5	6	8	288.8	0.004	3.0	10.7	2159	2165	2155	2154	0.0989	0.77	1.5	2.2	1.94	0.0003
36	6	7	8	227.8	0.022	10.7	3.0	2165	2153	2154	2149	0.103	1.79	2.8	5.1	1.3	0.0003
37	7	9	8	189.7	0.031	3.0	3.0	2153	2147	2149	2143	0.1072	2.12	3.2	6.1	1.23	0.0002
38	9	11	8	175.7	0.044	3.0	3.0	2147	2139	2143	2135	0.1372	2.55	3.9	7.3	1.25	0.0002
39	11	19	8	208.7	0.004	3.0	3.4	2139	2139	2135	2135	0.1874	0.77	1.8	2.2	2.69	0.0002
40	26	21	8	149.1	0.010	3.0	3.0	2152	2151	2149	2147	0.0044	1.21	0.8	3.5	0.35	0.0002
41	8	10	8	158.4	0.004	3.0	5.4	2166	2168	2163	2162	0.0303	0.77	1.1	2.2	1.08	0.0002
42	10	15	8	138.8	0.083	5.4	3.0	2168	2154	2162	2151	0.0804	3.48	4.1	9.9	0.83	0.0002
43	15	21	8	161.9	0.021	3.0	3.0	2154	2151	2151	2147	0.0846	1.76	2.6	5	1.2	0.0002
44	21	25	8	141.0	0.083	3.0	3.0	2151	2139	2147	2135	0.0929	3.48	4.3	10	0.91	0.0002
45	25	19	8	269.7	0.004	3.0	3.7	2139	2139	2135	2134	0.1231	0.77	1.6	2.2	2.16	0.0003
46	19	27	8	165.9	0.004	3.7	6.0	2139	2140	2134	2134	0.3144	0.77	2.1	2.2	3.57	0.0002
47	27	35	8	180.2	0.026	6.0	3.0	2140	2133	2134	2129	0.3645	1.95	4.3	5.6	2.35	0.0002
48	12	20	8	196.9	0.019	3.0	3.0	2163	2159	2159	2155	0.0303	1.66	1.8	4.8	0.74	0.0002
49	20	28	8	202.1	0.034	3.0	3.0	2159	2152	2155	2148	0.0345	2.24	2.4	6.4	0.7	0.0002
50	28	36	8	181.0	0.034	3.0	3.0	2152	2146	2148	2142	0.0646	2.25	2.8	6.4	0.94	0.0002
51	36	43	8	238.7	0.040	3.0	3.0	2146	2136	2142	2132	0.0688	2.43	3	6.9	0.91	0.0003
52	43	35	8	201.9	0.017	3.0	3.0	2136	2133	2132	2129	0.1189	1.57	2.7	4.5	1.49	0.0002
53	35	37	8	261.3	0.016	3.0	3.0	2133	2128	2129	2125	0.4874	1.55	3.9	4.4	3.08	0.0003
54	37	46	10	225.1	0.009	3.0	3.0	2128	2127	2125	2123	0.8092	2.04	3.5	3.7	4.38	0.0003
55	47	46	8	232.9	0.005	3.0	3.0	2128	2127	2124	2123	0.0304	0.88	1.2	2.5	1.01	0.0003
56	50	46	8	236.9	0.052	3.0	3.0	2139	2127	2135	2123	0.0045	2.76	1.1	7.9	0.16	0.0003
57	46	60	10	277.5	0.049	3.0	3.0	2127	2113	2123	2109	0.8477	4.87	6.7	8.9	2.81	0.0004
58	60	69	10	195.3	0.003	3.0	3.5	2113	2113	2109	2108	0.8517	1.16	2.3	2.1	6.36	0.0003
59	69	75	10	192.2	0.031	3.5	3.0	2113	2106	2108	2102	0.8559	3.89	5.7	7.1	3.19	0.0003
60	75	83	10	210.4	0.003	3.0	5.9	2106	2109	2102	2102	0.8601	1.16	2.3	2.1	6.4	0.0003

PIPE NUM.	M.H. NUMBER Upp.	Low.	PIPE DIA. (IN)	PIPE LENGTH (FT)	SLOPE (FT/FT)	COVER Upp. (FT)	Low. (FT)	GROUND EL. Upp. (FT)	Low. (FT)	INVERT EL. Upp. (FT)	Low. (FT)	DESIGN FLOW (CFS)	PIPE CAP. (CFS)	ACTUAL VEL. (FPS)	FULL VEL. (FPS)	ACTUAL DEPTH (IN)	INFILT. (CFS)
61	94	89	8	238.2	0.004	3.0	3.0	2133	2132	2129	2128	0.0045	0.77	0.6	2.2	0.42	0.0003
62	89	84	8	260.0	0.039	3.0	3.0	2132	2121	2128	2118	0.0212	2.41	2.2	6.9	0.55	0.0003
63	63	72	8	187.8	0.032	3.0	3.0	2158	2152	2155	2149	0.0044	2.18	1.2	6.2	0.25	0.0002
64	72	78	8	179.4	0.037	3.0	3.0	2152	2146	2149	2142	0.0086	2.32	1.6	6.6	0.35	0.0002
65	78	74	8	244.2	0.040	3.0	3.0	2146	2136	2142	2132	0.0128	2.44	1.8	7	0.42	0.0003
66	74	84	8	370.6	0.039	3.0	3.0	2136	2121	2132	2118	0.0171	2.39	2	6.8	0.48	0.0004
67	84	81	8	141.2	0.019	3.0	3.0	2121	2119	2118	2115	0.0419	1.68	2	4.8	0.87	0.0002
68	57	53	8	212.4	0.047	3.0	3.0	2175	2165	2171	2161	0.0044	2.62	1	7.5	0.16	0.0003
69	53	56	8	206.9	0.117	3.0	3.0	2165	2141	2161	2137	0.0086	4.14	2.2	11.8	0.25	0.0002
70	56	64	8	145.3	0.039	3.0	3.0	2141	2135	2137	2132	0.0127	2.39	1.8	6.8	0.42	0.0002
71	64	77	8	318.1	0.051	3.0	3.0	2135	2119	2132	2115	0.0171	2.75	2.1	7.8	0.42	0.0004
72	77	81	8	145.1	0.004	3.0	3.2	2119	2119	2115	2115	0.0211	0.77	1	2.2	0.91	0.0002
73	81	83	8	265.8	0.037	3.2	3.0	2119	2109	2115	2105	0.07	2.34	2.9	**	0.91	0.0003
74	83	92	10	330.8	0.058	5.9	3.0	2109	2087	2102	2083	0.9313	5.28	7.3	9.7	2.83	0.0005
75	92	95	18	68.1	0.001	3.0	3.1	2087	2087	2082	2082	0.9602	3.65	1.7	2.1	6.3	0.0002
76	95	99	18	107.8	0.001	3.1	5.3	2087	2089	2082	2082	0.9645	3.65	1.7	2.1	6.32	0.0003
77	93	100	8	151.4	0.004	3.0	3.6	2106	2106	2103	2102	0	0.77	0.6	**	0.42	0.0002
78	98	103	8	187.3	0.045	3.0	3.0	2113	2104	2109	2101	0.0044	2.57	1	7.3	0.16	0.0002
79	105	103	8	240.2	0.086	3.0	3.0	2125	2104	2121	2101	0.0838	3.55	4.2	10.2	0.85	0.0003
80	103	100	8	242.0	0.004	3.0	5.8	2104	2106	2101	2100	0.0922	0.77	1.5	2.2	1.87	0.0003
81	100	99	8	198.0	0.075	5.8	3.0	2106	2089	2100	2085	0.1	3.31	4.3	**	0.96	0.0002
82	99	108	18	340.5	0.019	5.3	3.0	2089	2080	2082	2076	1.0696	14.4	4.8	8.1	3.33	0.0009
83	108	111	18	180.1	0.016	3.0	3.0	2080	2077	2076	2073	1.0733	13.3	4.5	7.5	3.48	0.0005
84	111	110	18	111.1	0.001	3.0	3.0	2077	2077	2073	2073	1.0773	3.65	1.8	2.1	6.71	0.0003
85	70	79	8	276.4	0.088	3.0	3.0	2142	2117	2138	2114	0.0045	3.59	1.4	10.3	0.16	0.0003
86	79	86	8	231.7	0.033	3.0	3.0	2117	2110	2114	2106	0.0086	2.19	1.5	6.3	0.35	0.0003
87	86	90	8	165.0	0.101	3.0	3.0	2110	2093	2106	2090	0.0127	3.84	2.6	11	0.35	0.0002
88	90	96	8	205.2	0.032	3.0	3.0	2093	2087	2090	2083	0.017	2.17	1.8	6.2	0.48	0.0002
89	54	58	8	103.6	0.060	3.0	3.0	2129	2123	2125	2119	0.0043	2.97	1.2	8.5	0.16	0.0001
90	58	62	8	121.7	0.043	3.0	3.0	2123	2117	2119	2114	0.01	2.51	1.7	**	0.35	0.0001
91	39	41	8	222.9	0.097	3.0	3.0	2146	2124	2142	2120	0.0044	3.77	1.5	10.8	0.16	0.0003

PIPE NUM.	M.H. NUMBER Upp.	Low.	PIPE DIA. (IN)	PIPE LENGTH (FT)	SLOPE (FT/FT)	COVER Upp. (FT)	Low. (FT)	GROUND EL. Upp. (FT)	Low. (FT)	INVERT EL. Upp. (FT)	Low. (FT)	DESIGN FLOW (CFS)	PIPE CAP. (CFS)	ACTUAL VEL. (FPS)	FULL VEL. (FPS)	ACTUAL DEPTH (IN)	INFILT. (CFS)
92	41	44	8	169.4	0.096	3.0	3.0	2124	2108	2120	2104	0.0086	3.75	2	10.7	0.25	0.0002
93	44	52	8	254.0	0.004	3.0	13.7	2108	2118	2104	2103	0.0128	0.77	0.8	2.2	0.7	0.0003
94	52	62	8	199.3	0.004	13.7	14.4	2118	2117	2103	2102	0.0295	0.77	1.1	2.2	1.08	0.0002
95	62	66	8	135.8	0.004	14.4	6.3	2117	2109	2102	2102	0.0419	0.77	1.2	2.2	1.28	0.0002
96	66	71	8	112.2	0.004	6.3	4.3	2109	2106	2102	2101	0.0461	0.77	1.2	2.2	1.33	0.0001
97	34	31	8	145.5	0.028	3.0	3.0	2134	2130	2131	2127	0.0043	2.02	1.1	5.8	0.25	0.0002
98	14	17	8	182.0	0.004	3.0	13.8	2136	2146	2132	2131	0.0303	0.77	1.1	2.2	1.08	0.0002
99	17	23	8	202.4	0.004	13.8	8.1	2146	2139	2131	2130	0.0345	0.77	1.1	2.2	1.15	0.0002
100	23	31	8	152.1	0.024	8.1	3.0	2139	2130	2130	2127	0.0386	1.9	2.1	5.4	0.78	0.0002
101	31	42	8	220.5	0.018	3.0	3.0	2130	2126	2127	2123	0.047	1.61	2	4.6	0.94	0.0003
102	42	48	8	208.2	0.004	3.0	3.0	2126	2126	2123	2122	0.047	0.77	1.2	2.2	1.33	0.0002
103	48	55	8	132.1	0.004	3.0	9.1	2126	2131	2122	2121	0.0469	0.77	1.2	2.2	1.33	0.0002
104	49	51	8	236.4	0.034	3.0	3.0	2150	2142	2147	2139	0.0045	2.22	1.2	6.3	0.25	0.0003
105	51	55	8	156.0	0.072	3.0	3.0	2142	2131	2139	2127	0.01	3.26	1.7	**	0.25	0.0002
106	55	59	8	174.3	0.023	9.1	3.0	2131	2121	2121	2117	0.0595	1.85	2.4	5.3	0.98	0.0002
107	59	65	8	167.8	0.032	3.0	3.0	2121	2116	2117	2112	0.0637	2.18	2.8	6.2	0.94	0.0002
108	65	67	8	144.7	0.023	3.0	3.0	2116	2112	2112	2108	0.0679	1.86	2.5	5.3	1.05	0.0002
109	67	71	8	141.0	0.041	3.0	3.0	2112	2106	2108	2103	0.072	2.46	3.1	7	0.94	0.0002
110	71	76	8	150.0	0.004	4.3	10.7	2106	2112	2101	2101	0.1222	0.77	1.6	2.2	2.16	0.0002
111	76	80	8	160.9	0.004	10.7	12.2	2112	2113	2101	2100	0.1264	0.77	1.6	2.2	2.2	0.0002
112	80	85	8	170.1	0.004	12.2	4.3	2113	2104	2100	2099	0.1305	0.77	1.6	2.2	2.24	0.0002
113	85	88	8	141.5	0.070	4.3	3.0	2104	2093	2099	2090	0.1347	3.21	4.6	9.2	1.13	0.0002
114	88	96	8	221.2	0.030	3.0	3.0	2093	2087	2090	2083	0.139	2.09	3.4	6	1.4	0.0003
115	96	101	8	155.1	0.010	3.0	3.0	2087	2085	2083	2081	0.1598	1.19	2.4	3.4	1.99	0.0002
116	101	104	8	111.3	0.004	3.0	4.9	2085	2087	2081	2081	0.1639	0.77	1.7	2.2	2.51	0.0001
117	104	109	8	87.5	0.048	4.9	3.0	2087	2081	2081	2077	0.1681	2.66	4.2	7.6	1.36	0.0001
118	109	107	8	123.3	0.018	3.0	3.0	2081	2078	2077	2075	0.1723	1.62	3	4.6	1.76	0.0001
119	97	102	8	174.2	0.012	3.0	3.0	2095	2093	2092	2090	0.0044	1.32	0.9	3.8	0.35	0.0002
120	102	107	8	148.5	0.100	3.0	3.0	2093	2078	2090	2075	0.0085	3.84	2.1	11	0.25	0.0002
121	107	106	8	94.1	0.014	3.0	3.0	2078	2077	2075	2073	0.1848	1.41	2.8	4	1.96	0.0001
122	106	110	8	132.2	0.004	3.0	3.6	2077	2077	2073	2073	0.189	0.77	1.8	2.2	2.71	0.0002

PIPE NUM.	M.H. NUMBER Upp.	Low.	PIPE DIA. (IN)	PIPE LENGTH (FT)	SLOPE (FT/FT)	COVER Upp. (FT)	Low. (FT)	GROUND EL. Upp. (FT)	Low. (FT)	INVERT EL. Upp. (FT)	Low. (FT)	DESIGN FLOW (CFS)	PIPE CAP. (CFS)	ACTUAL VEL. (FPS)	FULL VEL. (FPS)	ACTUAL DEPTH (IN)	INFILT. (CFS)
123	110	112	18	253.6	0.012	3.6	3.0	2077	2074	2072	2069	1.2707	11.4	4.3	6.4	4.06	0.0007
124	112	114	18	144.7	0.001	3.0	3.2	2074	2074	2069	2069	1.2746	3.65	1.9	2.1	7.35	0.0004
125	114	115	18	83.6	0.073	3.2	3.0	2074	2067	2069	2063	1.2786	28.5	8.1	16.1	2.59	0.0002
126	115	117	18	103.0	0.066	3.0	3.0	2067	2060	2063	2056	1.3088	27.1	7.9	15.3	2.7	0.0003
127	126	124	8	100.7	0.125	3.0	3.0	2083	2070	2079	2066	0.0043	4.29	1.7	12.3	0.16	0.0001
128	124	120	8	145.5	0.074	3.0	3.0	2070	2059	2066	2055	0.0085	3.3	1.8	9.4	0.25	0.0002
129	120	117	8	133.9	0.004	3.0	4.8	2059	2060	2055	2055	0.0127	0.77	0.8	2.2	0.7	0.0002
130	117	123	18	200.3	0.025	4.8	3.0	2060	2054	2054	2049	1.3257	16.8	5.7	9.5	3.42	0.0005
131	123	127	18	168.4	0.031	3.0	3.0	2054	2048	2049	2044	1.4242	18.4	6.2	10.4	3.39	0.0004
132	127	132	18	252.4	0.001	3.0	14.4	2048	2059	2044	2044	1.4286	3.65	1.9	2.1	7.82	0.0007
133	132	135	18	128.0	0.001	14.4	11.1	2059	2056	2044	2043	1.4927	3.65	2	2.1	8.01	0.0003
134	137	134	8	122.4	0.014	3.0	3.0	2080	2078	2076	2075	0.0302	1.42	1.6	4	0.8	0.0001
135	131	129	8	138.8	0.053	3.0	3.0	2087	2079	2083	2076	0.0043	2.78	1.1	7.9	0.16	0.0002
136	118	122	8	111.3	0.079	3.0	3.0	2108	2100	2105	2096	0.0043	3.4	1.4	9.7	0.16	0.0001
137	122	125	8	127.4	0.049	3.0	3.0	2100	2093	2096	2090	0.0085	2.69	1.8	7.7	0.35	0.0002
138	125	129	8	160.5	0.087	3.0	3.0	2093	2079	2090	2076	0.0127	3.56	2.4	10.2	0.35	0.0002
139	129	134	8	234.8	0.004	3.0	3.0	2079	2078	2076	2075	0.0212	0.77	1	2.2	0.91	0.0003
140	134	133	8	137.9	0.081	3.0	3.0	2078	2067	2075	2064	0.0553	3.45	3.6	9.9	0.7	0.0002
141	133	135	8	136.3	0.082	3.0	3.0	2067	2056	2064	2052	0.09	3.48	4.2	**	0.87	0.0002
142	135	138	18	101.4	0.001	11.1	5.4	2056	2050	2043	2043	1.6079	3.65	2	2.1	8.35	0.0003
143	138	140	18	148.9	0.001	5.4	11.5	2050	2056	2043	2043	1.6122	3.65	2	2.1	8.37	0.0004
144	140	144	18	408.4	0.001	11.5	10.4	2056	2055	2043	2043	1.6254	3.65	2	2.1	8.41	0.0011
145	144	146	18	124.6	0.001	10.4	7.6	2055	2052	2043	2042	1.6372	3.65	2	2.1	8.45	0.0003
146	146	147	18	162.8	0.026	7.6	3.0	2052	2043	2042	2038	1.6498	17.1	6.1	9.7	3.77	0.0004
147	147	9999	18	8.0	0.001	3.0	3.0	2043	2043	2038	2038	1.6536	3.65	2	2.1	8.49	2E-05

7.3 Test Area 2 (Application to Wyatt Farms)

The program was also applied to a pump station designed the area of Blacksburg called the Wyatt Farms residential area.

7.3.1 Introduction:

Lusk et al. (1995) worked on designing a pump station for the Wyatt Farms residential area in the Town of Blacksburg. The area originally had two pump stations which were operating at full capacity. The new pump station is to replace the two existing ones. The actual complete project was done by Anderson and Associates Inc., an engineering consulting firm in Blacksburg.

As part of the pump station design, Lusk et al. had to choose the path to be followed by the force main from the pumping station to the destination of pumping. The purpose of this test is to compare the force main path chosen by Lusk et. al (1995) to the results produced by the GIS based optimal force main path determination program developed in this research.

7.3.2 Description of Test Site:

The area of concern is the Wyatt Farms residential area in Blacksburg, VA. According to Lusk et. al., (1995) the projected saturation level number of homes in the area is 669 with each home producing a flow of 400gpd (infiltration included). This results in an average flow of 186 gpm and a peak flow of 465 gpm using a peak factor of 2.5.

7.3.3 Materials Used:

The data used included a 2 foot contour map of the area, a map of right of ways, and property boundaries. Another map layer was created which contained

the source where the water is to be pumped from and the destination. Apart from the layer containing the source and destination points, all the others were obtained in AutoCAD format. Figure 7.3.1 shows the property lots and right of ways for the streets. It also indicates the source and destination points, together with the topography of the area.

7.3.4 Data Conversion Procedure:

Because the data was in AutoCAD format, the initial task was to convert it into ARC/INFO format. This was done by first saving the layers in DXF format in AutoCAD and then exporting the DXF files into ARC/INFO. ARC/INFO was then used to covert the DXF files into ARC/INFO layers or coverages.
The topographic layer was subsequently converted to a triangular irregular network or TIN.

7.3.5 Results and Discussion:

The property lots were used as prohibited areas. This implies that, it is not desirable to allow any force main path to go through these lots. The path of the force main obtained from the run is compared with the existing path as shown on Figure 7.3.1. The existing path refers to the path used by Lusk et. al. (1995). The force main path determined by the GIS based program is clearly different from the existing one (the path used by Lusk et. al., 1995). The resulting path, referred to as 'force main path1' on Figure 7.3.1, has a length of 2690 ft as compared to the path chosen by Lusk et. al. which has a length of 2910 ft. Also, a plot of the profiles of the two paths (see Figure 7.3.2) shows that the GIS based force main path determination program in general avoids undulating paths.

The path obtained by the GIS based program was also compared to a case where the existing streets are marked as prohibited areas. This is based on the assumption that it is not desirable to dig up the already existing streets and the 'force main path1' may probably not have been chosen because of the cost involved in the attainment of easements. The results obtained when the streets were marked as prohibited is shown in 'force main path2' in Figure 7.3.1. In this case, the path chosen by the program had a length of 2968 ft. This length is about 58 ft more than the length of the path used by Lusk et. al (1995). This increase in length may be resulting from the fact that the modified length concept upon which the path finding program is based places a heavier penalty on variations in the vertical rise than variation in the horizontal plane. Therefore by trying to avoid changes in elevation, the length of the path is increased. The profile view of the two paths shown in Figure 7.3.3 depicts this trend.

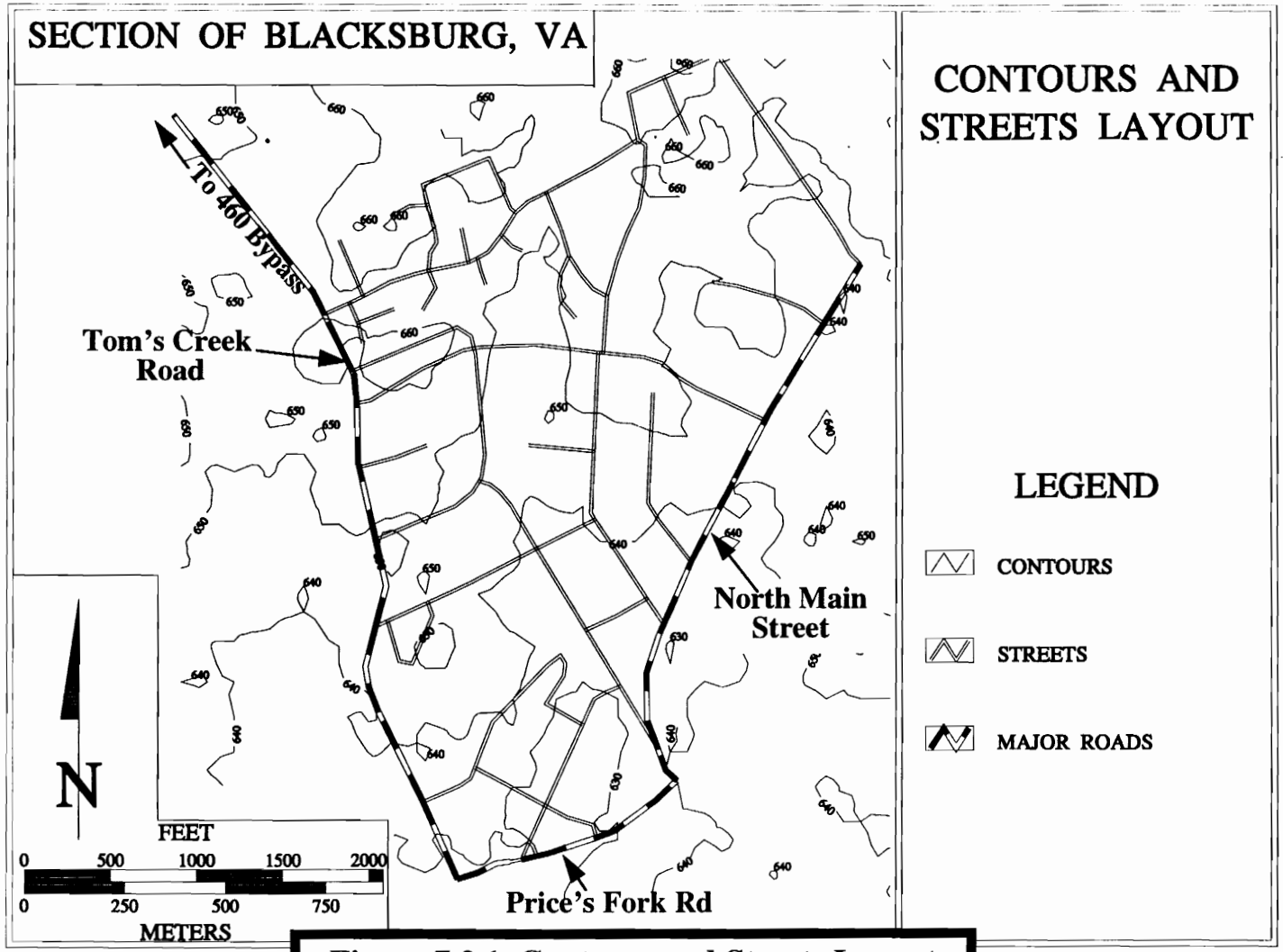


Figure 7.2.1 Contours and Streets Layout

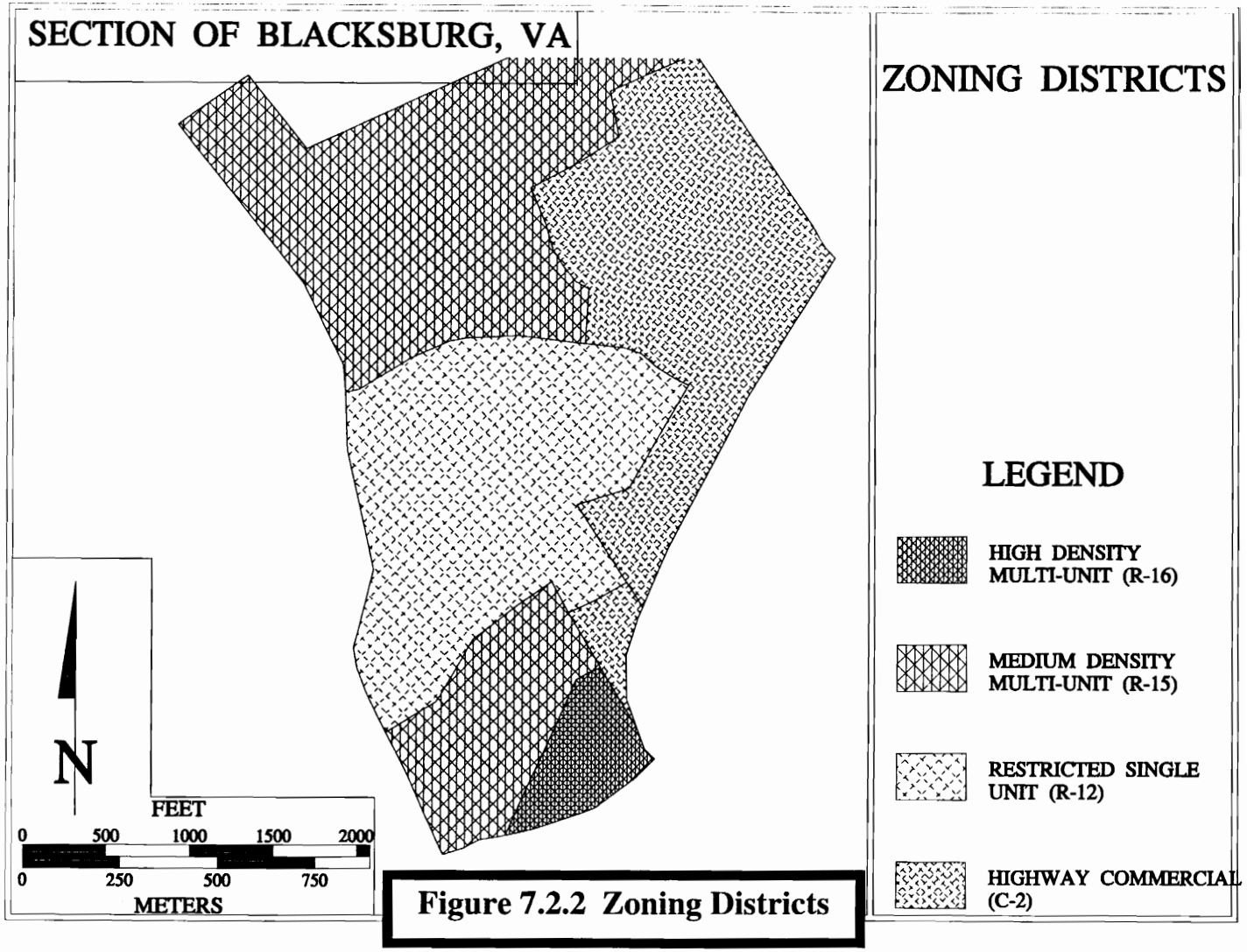


Figure 7.2.2 Zoning Districts

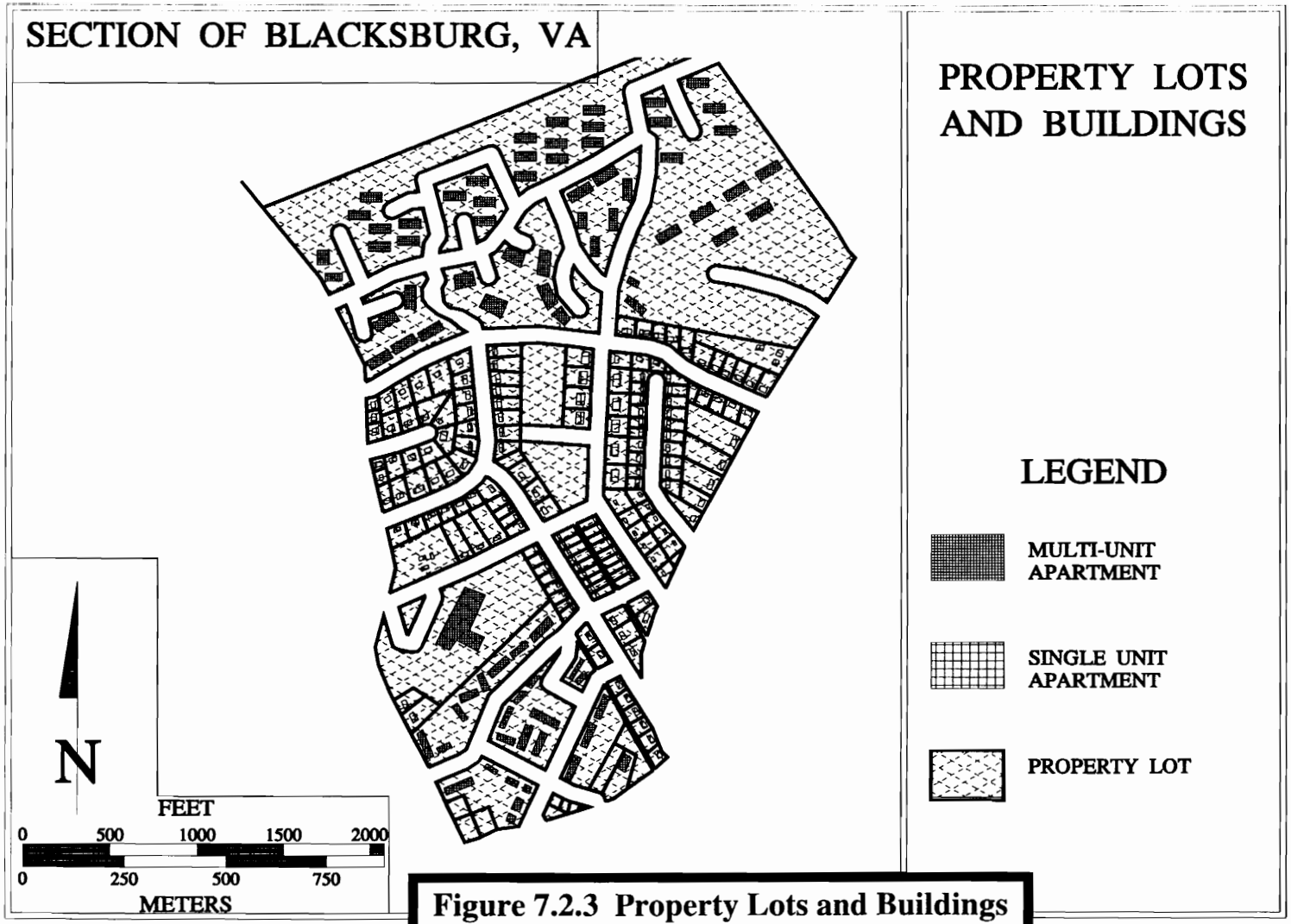


Figure 7.2.3 Property Lots and Buildings

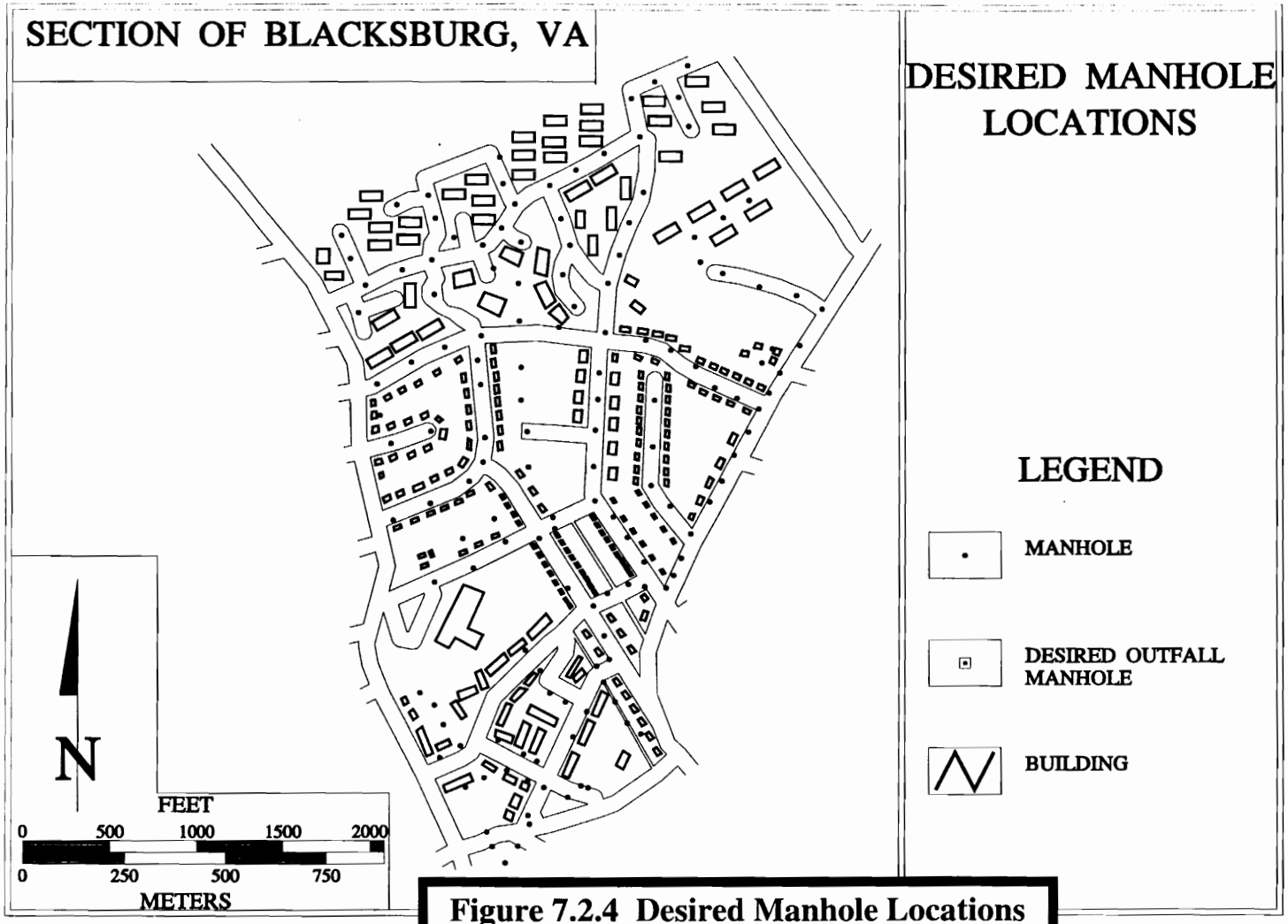


Figure 7.2.4 Desired Manhole Locations

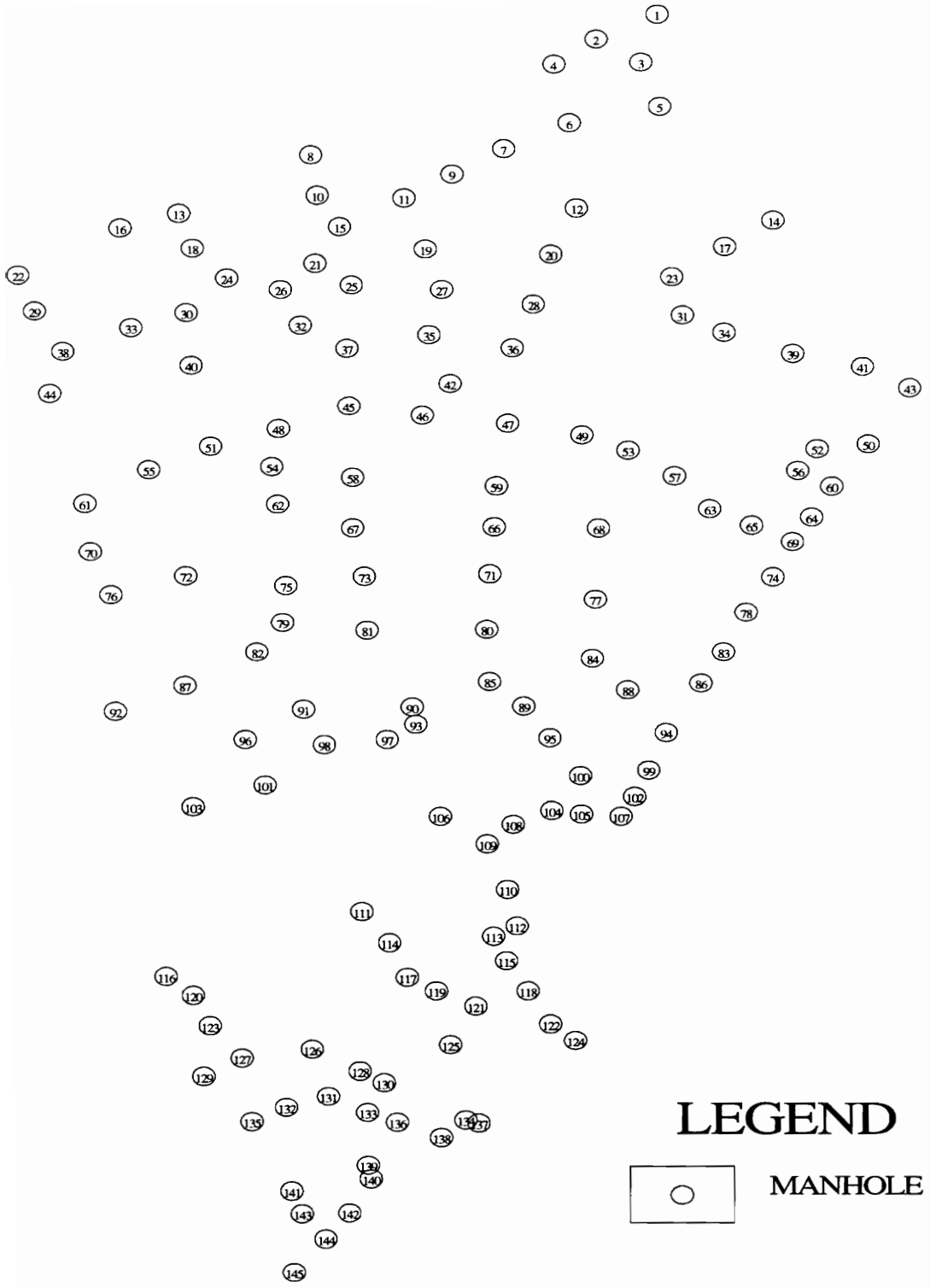


Figure 7.2.5 Manhole and Manhole Numbers

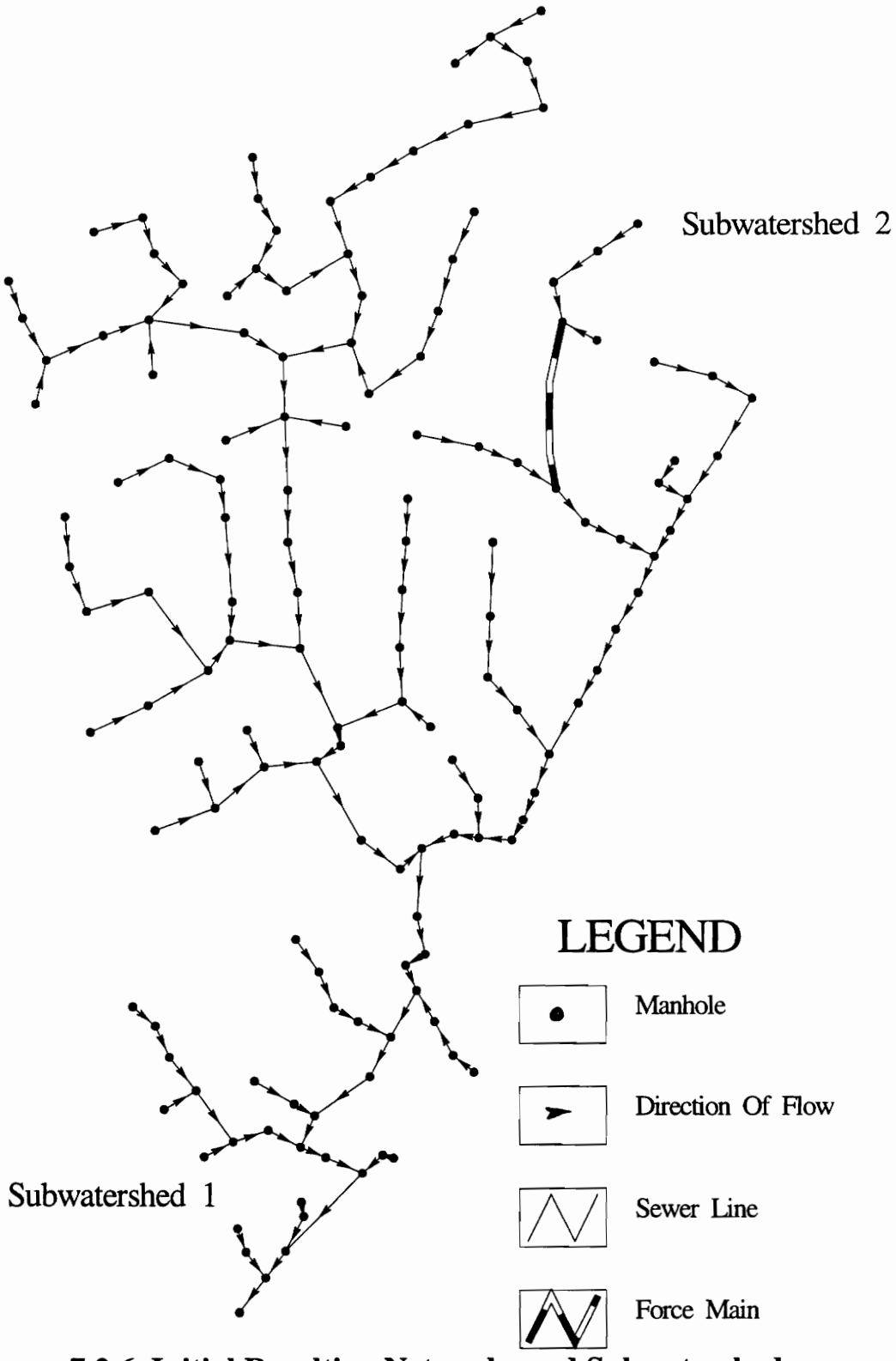


Figure 7.2.6 Initial Resulting Networks and Subwatersheds

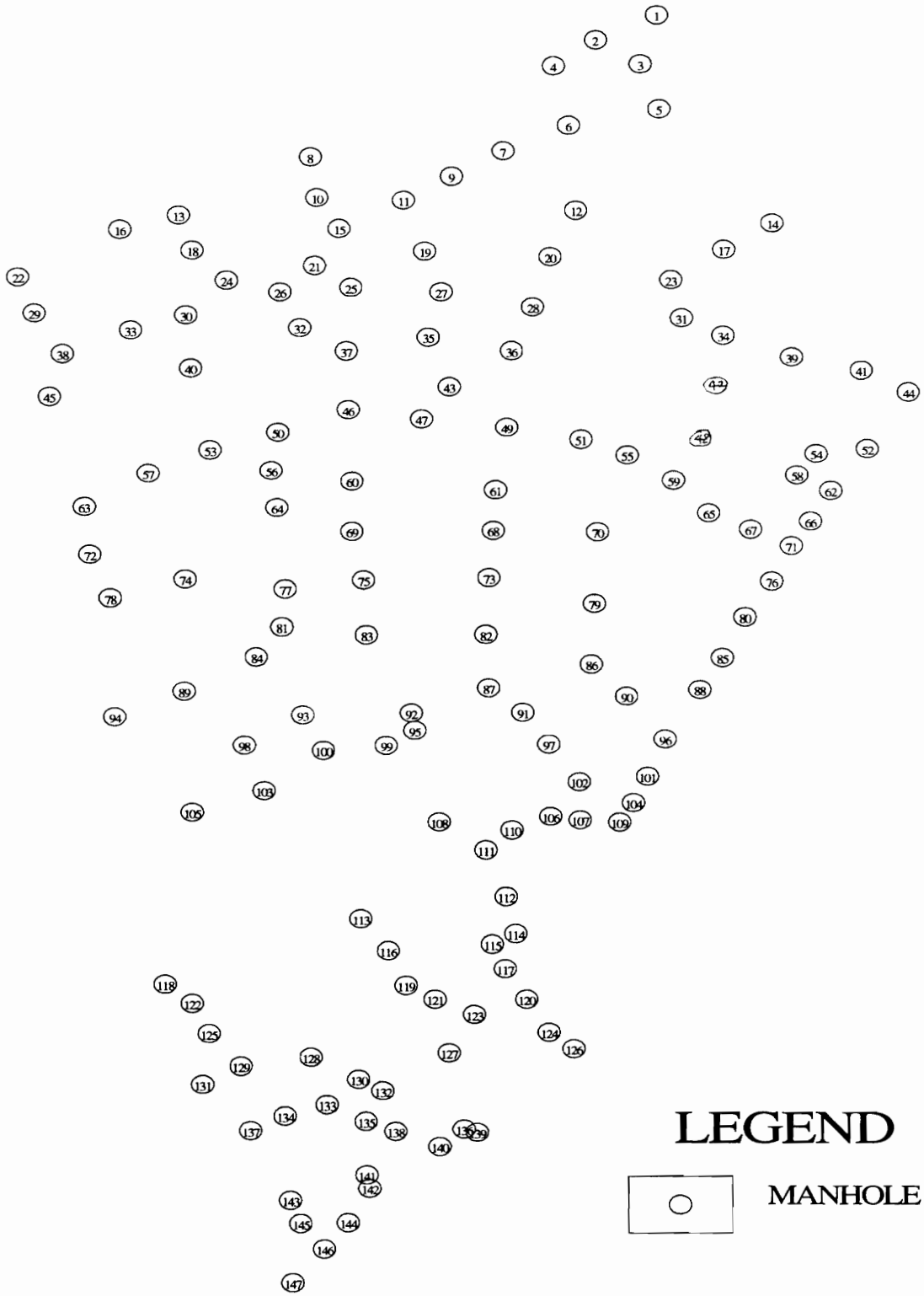


Figure 7.2.7 Two Extra Manholes Added

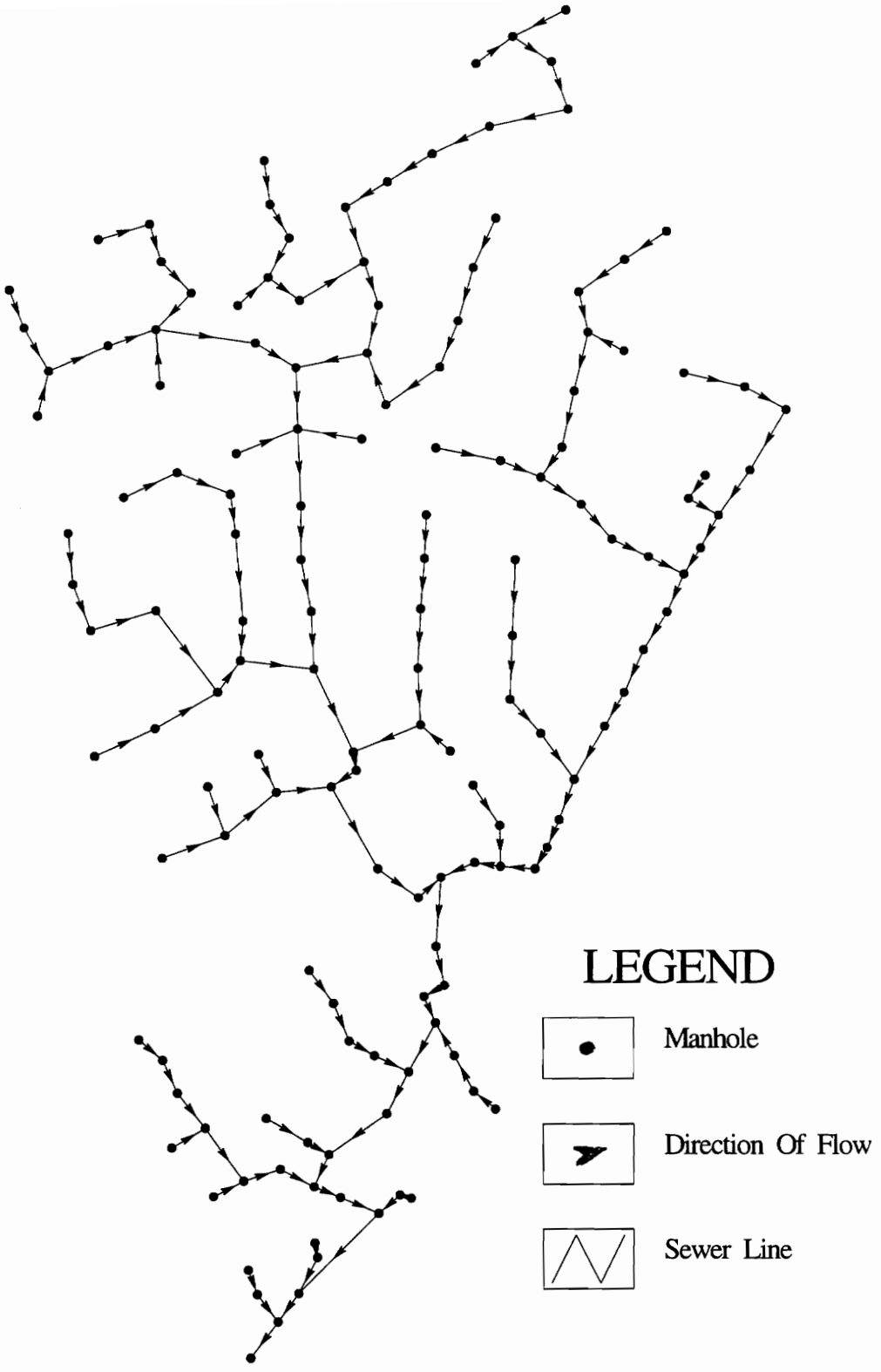


Figure 7.2.8 Final Network Layout

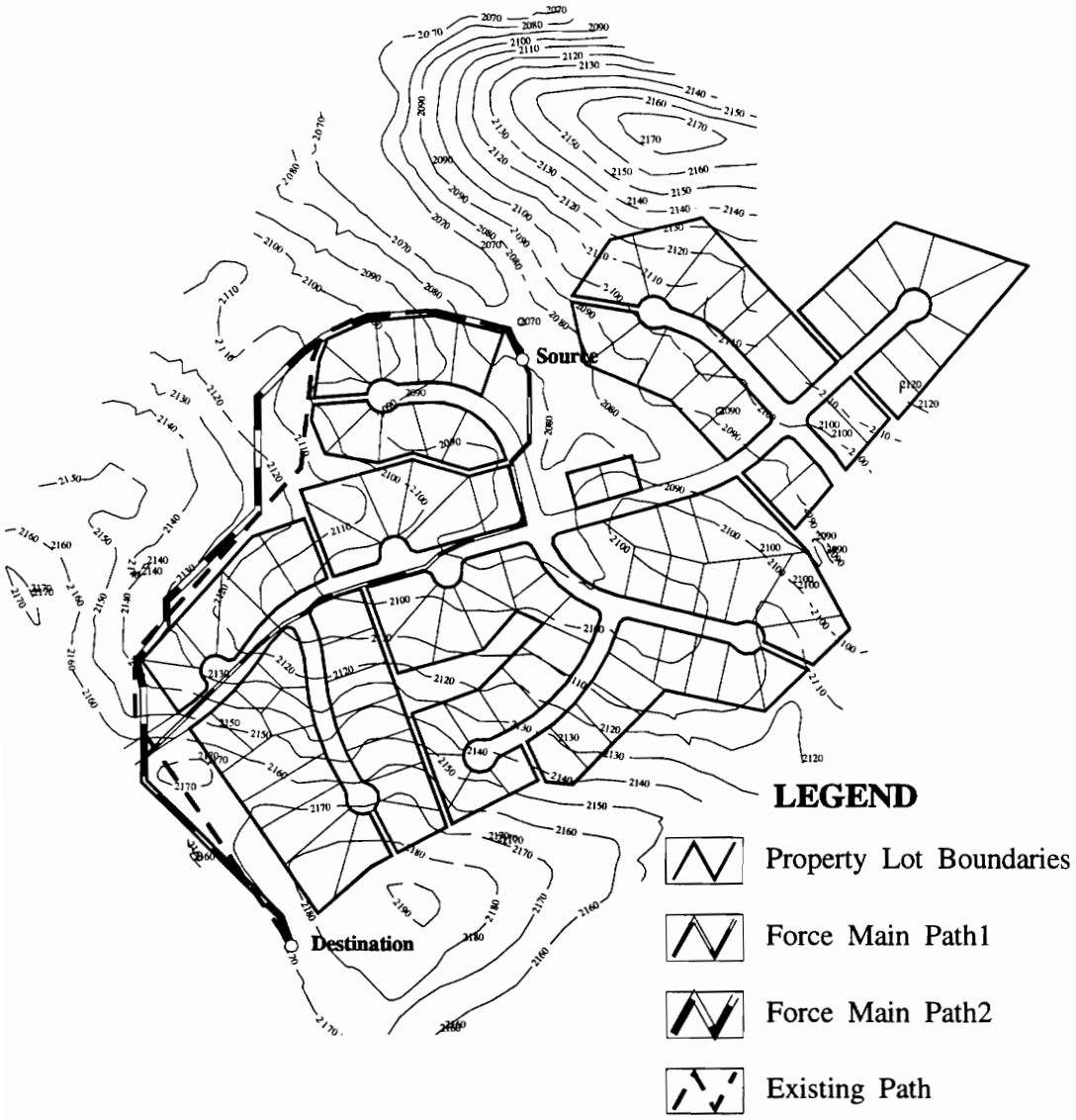


Figure 7.3.1 Various Paths for Force Main

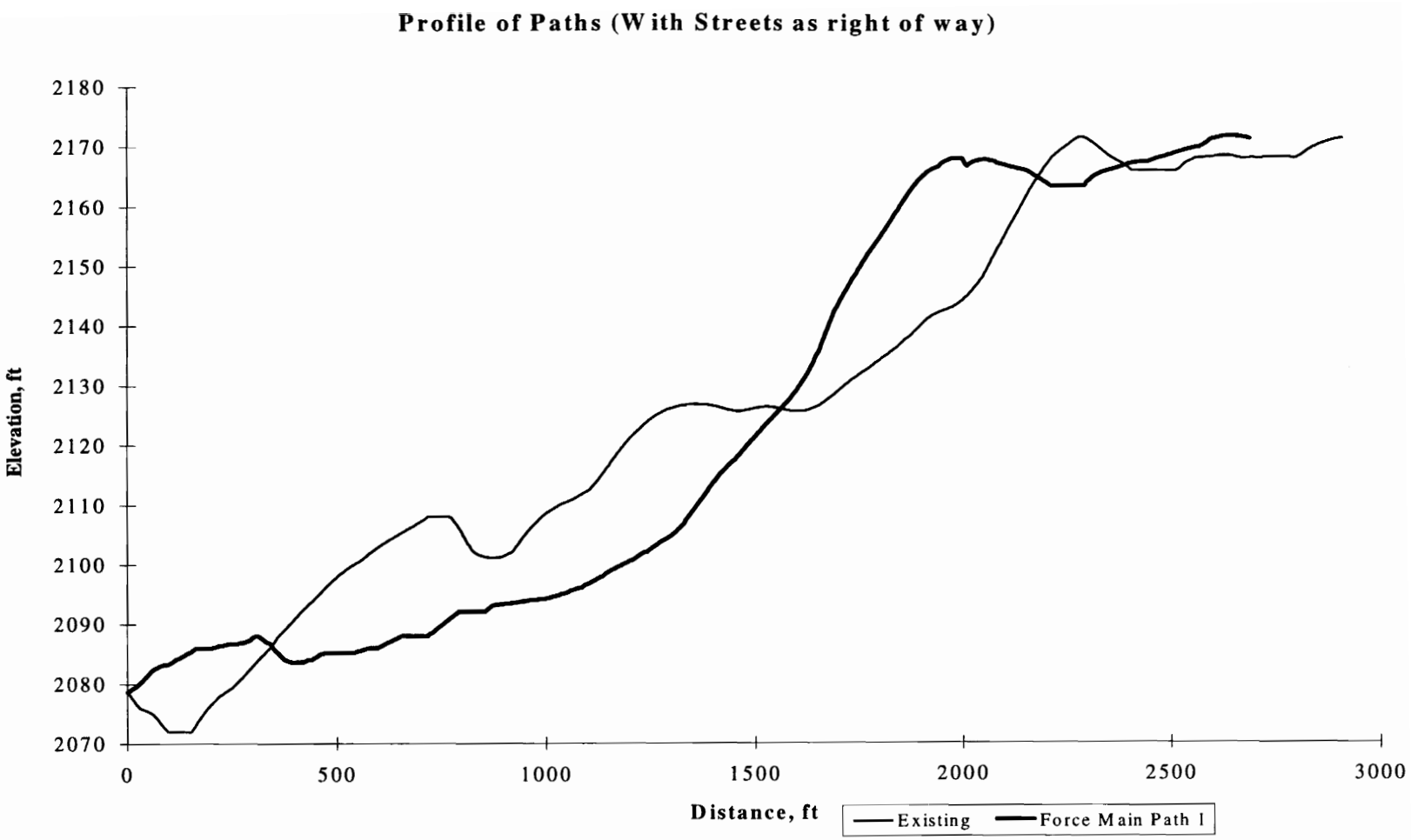


Figure 7.3.2 Profile of paths (I)

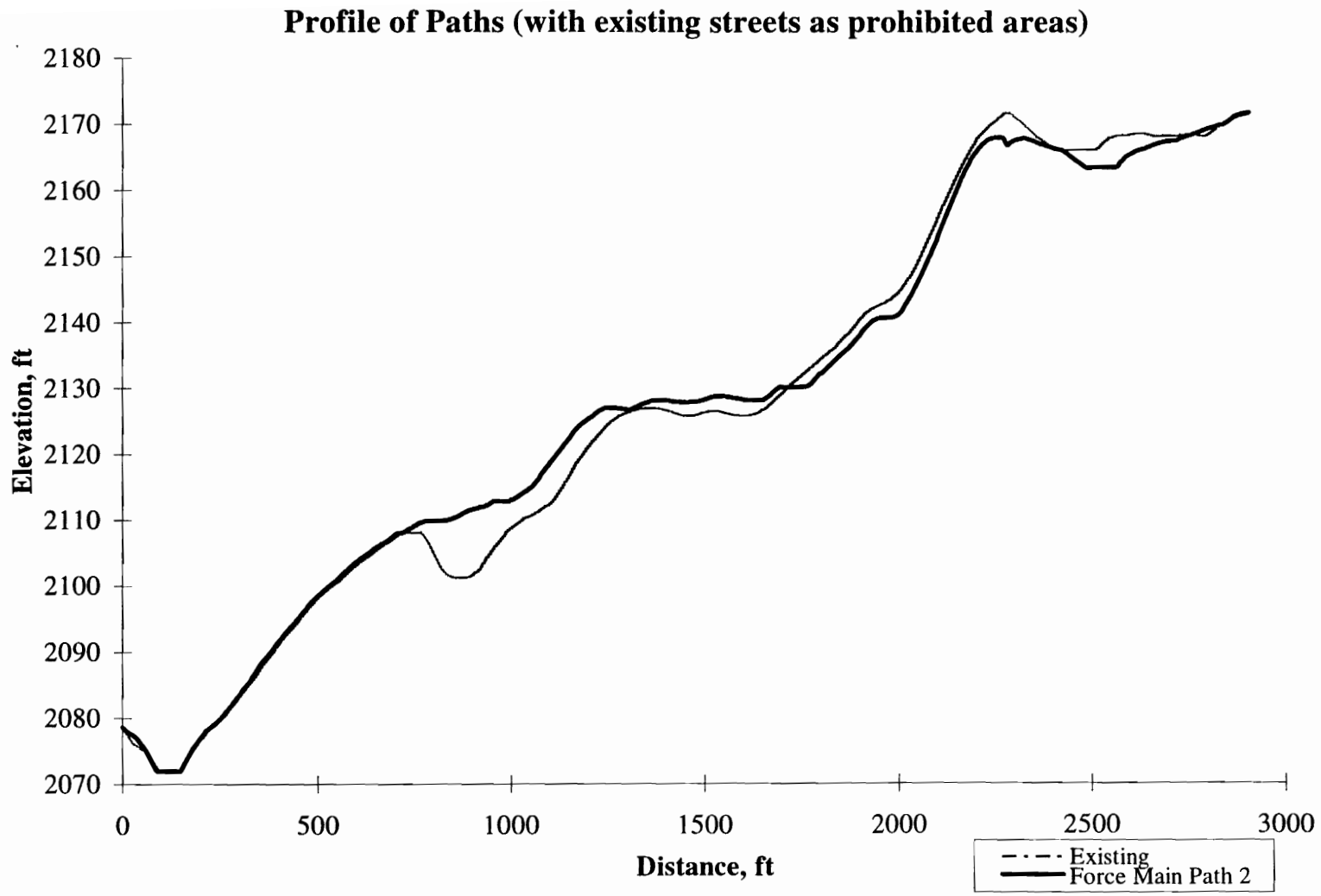


Figure 7.3.3 Profile of Paths (II)

Chapter 8 -Conclusions and Recommendations

8.1 Conclusions

The approach described in this paper successfully demonstrates the versatility in the use of GIS as a tool to design sewer networks. The advantage of the GIS to aid in the integration and management of the required data for the system design is also clear.

The ability of the GIS based program to determine the locations of pump stations and also to determine the force main paths based only on the user's input of manhole locations sets this work apart from previous research in this area. Also, with this tool, it is possible for a planner or engineer to develop a comprehensive sewer network for a large area. What-if analysis can be easily made by just altering such constraints as the manhole locations and the prohibited area sizes. Further annexation possibilities can be analyzed long before their need arises.

The graphics capability of the GIS also allows for immediate visualization of the results and any modifications required can be easily made without having to reconsult maps for information. With the program, the user has the option and opportunity of modifying the input such as the manhole locations to determine if there could be any further improvements on the results.

The results from the test area 1 in which the program initially delineated two subwatersheds shows that some further modification needs to be made to the program. It is currently relying on the user's judgment and expertise to insert manholes along the path of force mains to determine if it may be possible to use a gravity connection in certain cases instead of a force main. As can be seen from the results of the force main path determination in the Wyatt Farms problem (test

area 2), the program is a handy tool that is able to aid in the selection of a path in the design of force mains.

The advantage in using GIS in the design is that designers do not have to deal with bulky maps. It makes it easier to know what to look for when actual surveys are to be made for the design project. Also the computer does overlays that are used to determine elevations which would otherwise have been determined by a manual interpolation between contour lines or by additional surveying. Alternative designs can be quickly analyzed by changing the manhole locations and rerunning the program. For the force main path determination, decisions concerning whether to obtain an easement or buy certain properties can be made based on the paths chosen by the program and a comparison of the costs of construction and maintenance and cost of obtaining the easements.

The GIS also facilitates the determination of the path to be followed by the force main and sewer pipes so as to avoid areas that have been designated as prohibited areas.

The input information required by such sewer design programs as GSDPM3 are easily generated. With known manhole locations, the GIS package can extract their elevations and lengths of pipes from the given topographic map. The approach of sewer design described in this paper enables the designer to design sewer systems just by choosing the desired locations of the manholes.

The major benefit of using GIS is the ability to quickly evaluate costs, and technical objectives in the design process and to eliminate a large number of alternative configurations. The graphics capability of the GIS also enhances the comprehension of the design procedure and makes it easier to determine existing structures or features that may be obstacles to the design of the system. With the

completion of the project, city engineers and planners should have a versatile tool for design of sewer networks and pump stations.

8.2 Limitations

ARC/INFO, even though very sophisticated in terms of the tools it offers, can be slow to learn (has a long learning curve). Though this program was developed using ARC/INFO, the methods described here can be applied to any other vector based GIS package. Most of the subprograms and subroutines used in the main program were developed on a 386 DX-40 MHz computer with an 8 MB RAM. The program uses a lot of dimensioned arrays. It was discovered that, in many cases, the 8 MB was insufficient for the program to run. The UNIX based Sun Workstation which is the main computer on which the program runs together with ARC/INFO, has 48 MB of RAM. This RAM size was found to be very sufficient in most cases.

Many of the procedures used which require many iteration and loops were found to be much much faster when run outside ARC/INFO. For example, the Dijkstra shortest path finding algorithm which is used in solving the shortest path problem is provided as part of the NETWORK module of ARC/INFO. Because of the many repetitions in the finding of the shortest path, it was found to be much faster to do the path finding by custom coding the Dijkstra's algorithm in C language and compiling it on the UNIX platform. The compiled version was faster than using the one in the network module.

It must be stressed that the program depends on the areas polygons representing prohibited areas for the determination of the path so that it does not pass through any of these areas. Also, the grid spacing which is dependent on available computer memory is also a deciding factor in the path chosen by the

program. The program seems to be limited by the fact that it depends on the spacing of the grid used in deciding the path. The spacing of the grid therefore determines the resulting path. A finer grid would result in a better path. The use of a finer grid spacing is also limited by the computer memory as a finer grid spacing would require more grid points which will take up more memory space to process.

8.3 Recommendations

Maps that contain soil and geologic information in the area for which the sewer is to be design could also be included to enable a better estimation of costs of trenching and also to guide the program to choose better paths. The program seems to be dependent on the chosen location of the manholes and this tends to dictate the end result. A better solution would depend on a better placement of desired manhole locations. This could be achieved by the process of interactively changing manhole locations and running the program each time to see whether a better solution would be achieved. A suggestion for further research would be to devise a method which is able to efficiently alter the manhole locations and rerun the program again. The procedure used by Walters (1985) in varying the manhole locations for a more efficient sewer network design could also be combined with the GIS based program to help in the design of a more efficient network. Also strategically placing the manholes so that most sewers are design in the right of ways or streets and along the boundaries of property lots would also decrease the need to obtain extra easements for sewer lines.

The resulting force main path chosen in any case also is affected by the grid spacing used. A smaller grid spacing requires more computer memory. The use of pointers (a concept in programming) could help improve paths chosen by the force

main path determination program. Use of pointers will help the program to use the available computer memory more efficiently and hence decrease the memory usage by the program. Future work should focus on the use of the concept of pointers to make memory usage more efficient thus allowing for the use of finer grid sizes.

The program could also be made more intelligent by specifying conditions under which gravity connections could be used instead of force mains thus relieving the user from having to interactively inspecting and modifying the output results.

References

ARC/INFO User Guides. ESRI (Environmental Systems Research Institute). Redlands, 1992.

Bakken, Darrell J. and Charline M. Avey. "Integration of AM/FM/GIS with MODELING/DESIGN on Large Utility PC Network." Computing in Civil Engineering and Geographic Information Systems Symposium. ASCE, New York. 1992

Charalambous, C. and A. A. Elimam. "Heuristic Design of Sewer Networks." *Journal of Environmental Engineering* 116 (3), 1181-1199, 1977.

Crouch, Gary. Personal Interview. Anderson and Assoc. May 1995.

Dajani, Jarir S., Yakir Hasit, and Stephen McCullers. "Mathematical Programming in Sewer Network Design." *Engineering Optimization*, (3), 27-35, 1977.

Design and Construction of Sanitary and Storm Sewers," ASCE Manual of Engineering Practice, No. 37, New York, 1970.

Douglas, J. F., J. M. Gasiorek, and J. A. Swaffield. Fluid Mechanics. Pitman. London. 1979.

"GIS Leads to More Efficient Route Planning." *Oil and Gas Journal.* Vol. 91 (17) PenWell. 1993.

Gray, Donald D., Mark A. Pacheco, Richard L. Coffman, and John D. Quaranta. Gravity Sewer Design Program Version 3.0 M: User's Guide. West Virginia University. Morgantown, 1992.

Guisset, Pierre. "Introduction of Fictitious Discharges and Use of a Reduced Network for the Optimal Design of urban Storm Sewers." Third

International Conference on Urban Drainage, Chalmers University, Sweden, 1984.

Lusk, Howard, Daniel McGinnis, Jeff McInnis, and Justing Nyland. *"Wyatt Farms Pump Station."* Unpublished Class Project Report. Virginia Tech. 1995.

Lui, Guiyi and Robert G. S. Matthew. "New Approach for optimisation of Urban Drainage Systems." *Journal of Environmental Engineering* 116 (5), 927-944, 1990.

Orth, H. B. Nandy, and W. I. Rabbani. "Design of Urban Drainage Networks by a Combined Dynamic Programming and Branch-and-Bound Approach." Third International Conference on Urban Drainage, Chalmers University, Sweden, 1984.

Phillips, Don T., A. Ravindran, and James Solberg. *"Operations Research: Principles and Practice."* Wiley, New York. 1976.

Przybyla, John and Cherie L. Kiesler. *"Extending GIS Capabilities For Enhanced Sewer System Modeling."* Civil Engineering Applications of Remote Sensing and Geographic Information Systems. ASCE, New York, 1991.

"Recommended Standards for Sewage Works: Policies for the Review and Approval of Plans and Specifications For Sewage Collection and Treatment." Health Education Service, Albany, 1978.

Sanks, Robert L. Ed. Pumping Station Design. Butterworths. Boston. 1989.

"Sewer Regulations Pursuant to Section 62.I-44.I9(8) of the Code of Virginia (1950), As Amended." State Department of Health and State Water Control Board, Commonwealth of Virginia Waters, Richmond, 1977.

Tchobanoglous, George. "Wastewater Engineering: Collection and Pumping of Wastewater." McGraw-Hill, New York. 1981.

Tekeli, Sahim and Huseyin Belkaya. "Computerized Layout Generation for Sanitary Sewers." *Journal of Water Resources Planning and Management,* ASCE, Vol. 112, No. 4, 1986.

Vogelsong, David. "An Outline for Wet Well Design and Pump Selection." Unpublished. Blacksburg, 1994.

Waier, Philip R. Means Open Shop Building Construction Cost Data. 11th ed. 1995. R. S. Means Company . Kingston. 1994.

Walters, G. A. "The Design of the Optimal Layout for a Sewer Network." *Engineering Optimization*, (9), 37-50, 1985.

Appendix A.

(Brief Explanation of the Means Index, Adapted from GSDPM3 Users Guide)

The Means Index Costs obtainable from Means Costs estimation book allows for estimation of costs of construction. It allows also for the estimation of cost of one year based on the known costs from another year. Means Historical Cost Index values are tabulated in the Means Construction Cost Data publications Reference section, and the input values are listed under the Quarterly City Cost Index - Actual column. Means City Cost Index as tabulated in the Means Construction Cost Data publication Reference section. Input index values are listed as the specific city's total weighted average and are based on a national average of 100 percent.
Sample Historical Cost Index Values and CITYIDX values based on Means 1992 Historical and City Cost Index data are:

CITY	YEAR	MEANSIDX	CITYIDX
Charleston, WV	1992	224.8	97.9
Pittsburgh, PA	1991	221.6	109.5
Youngstown, OH	1990	215.9	95.1
Columbus, OH	1980	144.0	97.5

The Means cost values for materials and installation are based on a national average. Adjustments can be made to apply the cost values to specific regions of the United States and Canada and for historical comparisons by using the Means Historical and City cost indexing system. The values for numerous cities and for previous years, going as far back as 1942, are listed in the appendix of the Means Cost Data books. The city cost indexes are listed as percentages of a national average = 100 percent and the historical cost indexes are based on January 1, 1975 = 100. The cost index values are ratios applied against the total estimated project costs. Adjustments for cost index values for cities not listed by Means can be made by evaluating the cost index data available for nearby cities and applying sound judgement and experience to select a reasonable cost index value.

Time adjustment using the Historical Cost Indexes:

$$\frac{\text{Index for Year A}}{\text{Index For Year B}} * \text{Cost in Year B} = \text{Cost in Year A}$$

Location adjustment using the City Cost Indexes:

$$\frac{\text{Index for City A}}{\text{Index For City B}} * \text{Cost in City B} = \text{Cost in City A}$$

Adjustment from the National Average:

$$\frac{\text{National Average Cost} * \text{Index for City}}{100} = \text{Cost in City A}$$

Appendix B.

An overview of ARC/INFO
(Adapted from ARC/INFO Users Guide, 1992)

Brief Review of ARC/INFO

ARC/INFO is a GIS software made up of a set of tools for creating, displaying, and managing computerized maps in vector format,(i.e. lines, points and polygons) or raster format (grid cells identified by rows and columns). It uses a topological and relational DBMS model. The ARC/INFO data model uses the principles of graph theory and topology to organize all cartographic information into a spatial network of features: points, lines, polygons and their attributes. Complex groupings of features are indexed with additional relationships in the data base.

ARC/INFO creates and manages 3-dimensional surfaces using a triangular irregular network (TIN) data structure. ARC/INFO tools are computer programs that perform four kinds of GIS functions:

1) Input functions
- coverage automation: this follows seven basic steps.
    a) prepare map for digitizing
    b) digitize map
    c) identify and correct digitizing errors
    d) define features and build topology
    e) identify and correct topological errors
    f) assign attributes to coverage features
    g) identify and correct attribute coding errors

- conversion to ARC-format: conversion of preexisting digital data to ARC/INFO format.
  e.g. DEM or AutoCAD format to ARC/INFO format.
- updating

2) Analysis functions
-thematic
-co-ordinate modification
-geometric
-selection/aggregation

3) Data Management function
- Data Base design
- processing control

4) Output function
- map display
- Reporting
- conversion to non-ARC format (for use in other programs)

ARC/INFO is composed of two subsystems:
a) ARC, a set of tools for managing cartographic data.
b) INFO, a generalized Relational Data Base Management System that is used for managing map attribute data.

The COVERAGE is a set of features, where each features has a location and possibly attributes.

Storage of coverage features in ARC/INFO:
Coverage features are stored , referenced, and managed as sets of coverage files. These files are organized into a directory and contain coordinate, topology, and attribute information for set of coverage features. Thematic attributes are managed in parallel with the locational information so that feature attributes can be referenced via the INFO directory.

The topological relationships between arcs, nodes, and polygons in a coverage are internally tracked by ARC/INFO through an internally assigned sequence number.

Example Arc information:
The basic information for each arc includes:
- the arc's unique internal # (id number)
- the number of points defining the arc (vertices)
- the x and y coordinates of each point defining the arc.

Arc topology is defined for each arc in terms of its to-node and from-node and polygon to the left and right of the arc. Nodes and polygons are identified by their unique internal sequence numbers.
ARC attributes are stored in ARC attribute table (AAT), POLYGON or POINT attributes stored in a POLYGON or POINT attribute tables (PAT). For polygon information, ARC/INFO uses the polygon topology to find out which arcs define the polygon and gets the coordinates of those arcs. All the attribute tables are accessible through from INFO.

Relating Attributes to Coverage Features
PAT and AAT files are used to store additional items for a particular set of map features. The additional items are related to feature # and/or feature ID.

The thematic information that must be stored in the coverage attribute file is limited to:
1) Spatial information (e.g., polygon area, arc length)
2) Numbers used to keep track of features and to link information to the features (i.e., features internal number and user ID).

Thematic information can also be stored outside of the coverage in another INFO file.

Possible coverage contents for each feature type
- tic features can have only coordinate information
- arc features can have coordinate information , topology, and an attribute file in the coverage.
-nodes can have coordinate information an d topology
- polygon can have only topology and attribute file
- label points as points features can have coordinate information and an attribute file.
- label points as polygon labels can have coordinate information and topology relative to polygons.

Coverage Files Accessible with INFO
- INFO is used to store thematic information for a coverage in tabular files.
- INFO data files that are used for accessing thematic information stored with the coverage include:
1) POLYGON or POINT Attribute Table (cover.PAT)
2)ARC Attribute Table (cover.AAT)
3) NODE Attribute Table (cover.NAT)
3) TIC file (cover.tic)
4) Boundary File (cover.BND)
-the boundary file defines the minimum and maximum extent of tic, arc, and Label point features in the coverage.
-the BND of a coverage is used for locationally indexing coordinate features in a coverage and is used for setting the MAPEXTENT (extent of coverage of the map) in ARCPLOT.

Three ways in which INFO data files are used by ARC:
1) To store additional coverage attributes. This INFO file and the coverage attribute file (PAT or AAT) usually share at least one common item so the additional INFO file item can be related to coverage features.
2) As a Lookup table which is used to create categories for a code value to be associated with each of the item categories. This file is used by ARC commands to lookup the code value associated with each item category.
3) To define ARCPLOT symbol definition files-SHADESET, LINESET, TEXTSET, and MARKERSET.

In ARC/INFO a fuzzy tolerance is used to define the resolution of a coverage. The fuzzy tolerance is the minimum distance separating arc coordinates in a coverage and arc coordinates within this distance are snapped together.
- the default fuzzy tolerance for a coverage is 0.002 coverage units.
- the resolution of a coverage is characterized by this precision and its relationship to the initial scale used for data entry (i.e. digitizing)
Example,

map scale = 1:63,360
maximum possible resolution of the digitized arcs is:
1 inch = 1 mile = 5280 feet
0.002 = 5280 0.002 = 10.56 ft

i.e., every distance within 10.56 ft will be snapped together.

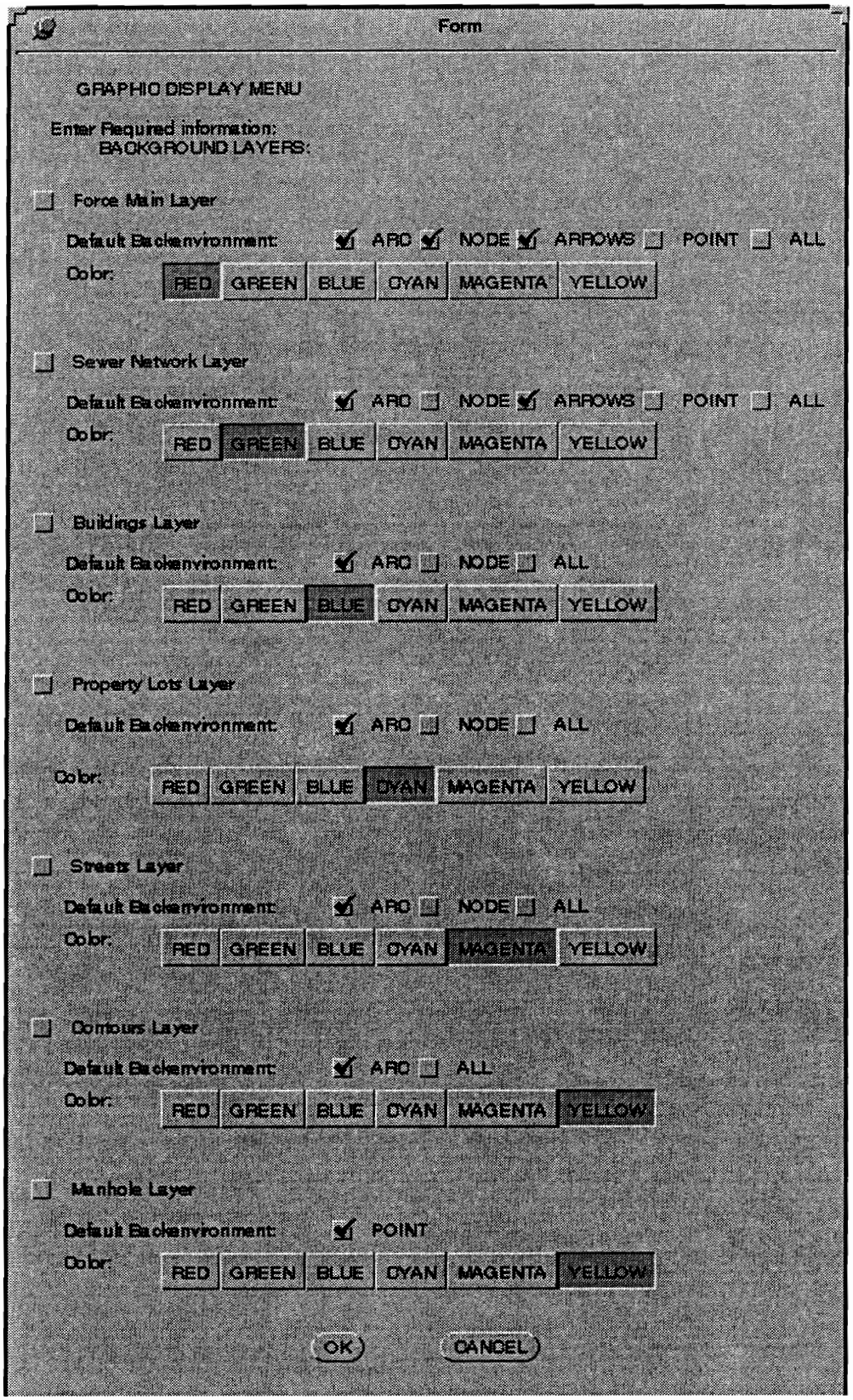
Other Tools Available In ARC/INFO:

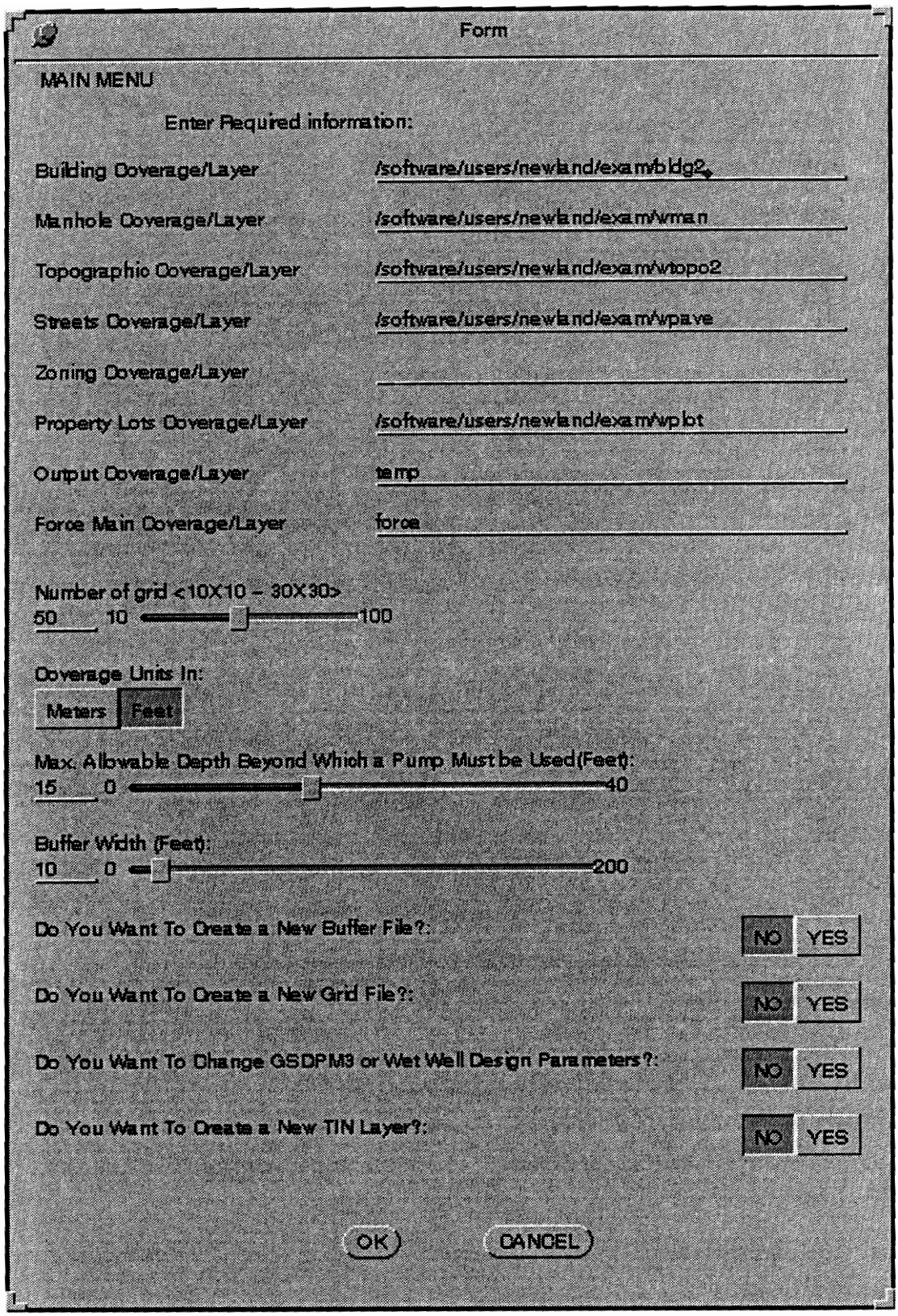
Besides being able to handle analysis and data in vector format, ARC/INFO also contains a module called GRID which can be used for cell based raster analysis. In the raster format, features are represented by grid cells as opposed to points, lines and polygons in the vector format.

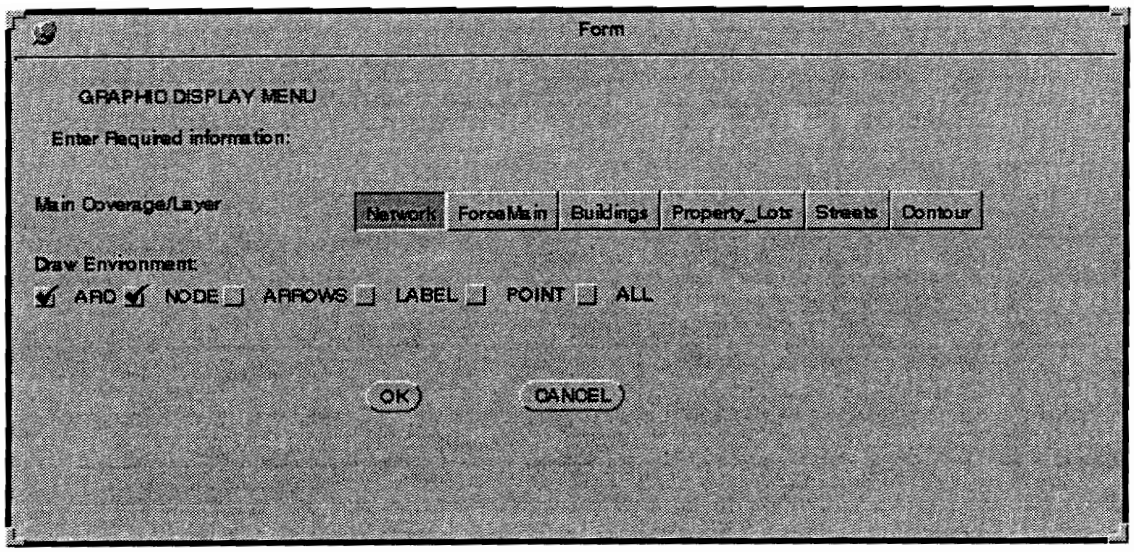
Two other modules which are vector based are also available for certain specific analysis. The NETWORK module is used in modeling network flows, path tracing analysis, shortest path analysis, etc. It is a very useful tool for analyzing transportation systems. The COGO module is useful for inputting surveying field data, and performing other surveying related analysis. It also acts as an interface for converting field data from surveying data loggers to ARC/INFO format.

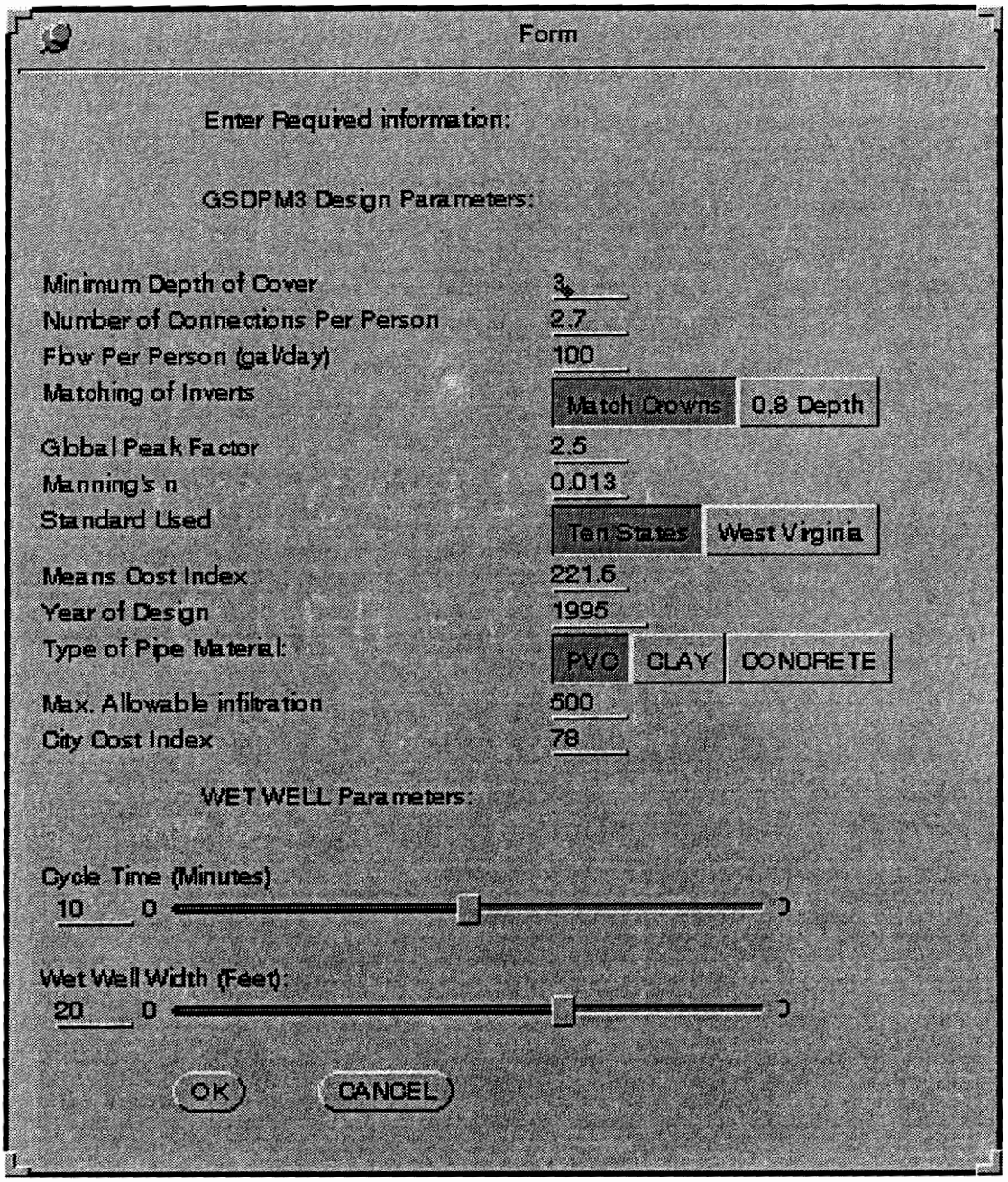
Appendix C

(Samples of Graphic User Interface Menus)


Form
GRAPHIC DISPLAY MENU
Enter Required information:
BACKGROUND LAYERS:
Force Main Layer
Default Backenvironment:
ARC
NODE
ARROWS
POINT
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Sewer Network Layer
Default Backenvironment:
ARC
NODE
ARROWS
POINT
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Buildings Layer
Default Backenvironment:
ARC
NODE
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Property Lots Layer
Default Backenvironment:
ARC
NODE
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Streets Layer
Default Backenvironment:
ARC
NODE
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Contours Layer
Default Backenvironment:
ARC
ALL
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
Manhole Layer
Default Backenvironment:
POINT
Color:
RED
GREEN
BLUE
CYAN
MAGENTA
YELLOW
OK
CANCEL


Form
MAIN MENU
Enter Required information:
Building Coverage/Layer
/software/users/newland/exam/bldg2
Manhole Coverage/Layer
/software/users/newland/exam/wman
Topographic Coverage/Layer
/software/users/newland/exam/wtopo2
Streets Coverage/Layer
/software/users/newland/exam/wpave
Zoning Coverage/Layer
Property Lots Coverage/Layer
/software/users/newland/exam/wplot
Output Coverage/Layer
temp
Force Main Coverage/Layer
force
Number of grid <10X10 - 30X30>
50 10 100
Coverage Units In:
Meters Feet
Max. Allowable Depth Beyond Which a Pump Must be Used(Feet):
15 0 40
Buffer Width (Feet):
10 0 200
Do You Want To Create a New Buffer File?: NO YES
Do You Want To Create a New Grid File?: NO YES
Do You Want To Change GSDPM3 or Wet Well Design Parameters?: NO YES
Do You Want To Create a New TIN Layer?: NO YES
OK
CANCEL


Form
GRAPHIC DISPLAY MENU
Enter Required information:
Main Coverage/Layer
Network
ForceMain
Buildings
Property_Lots
Streets
Contour
Draw Environment:
ARC
NODE
ARROWS
LABEL
POINT
ALL
OK
CANCEL


Form
Enter Required information:
GSDPM3 Design Parameters:
Minimum Depth of Cover
3
Number of Connections Per Person
2.7
Flow Per Person (gal/day)
100
Matching of Inverts
Match Crowns
0.8 Depth
Global Peak Factor
2.5
Manning's n
0.013
Standard Used
Ten States
West Virginia
Means Cost Index
221.6
Year of Design
1995
Type of Pipe Material:
PVC
CLAY
CONCRETE
Max. Allowable infiltration
500
City Cost Index
78
WET WELL Parameters:
Cycle Time (Minutes)
10
0
Wet Well Width (Feet):
20
0
OK
CANCEL

Appendix D

(Computer Programs)

Introduction:

This appendix tries to describe the various programs and their roles in the overall system. The main program is started by running the *START.AML* program. This program calls all the startup menu: *BUTTON.MENU*. It also calls the *GSDPM3.MENU*. Below is a description of each of the program codes. The codes have been grouped according to types: C programs, AML programs, MENU programs, and FORTRAN.

C PROGRAMS

Program Name: **COSTS.C**
Executable Form : COSTS.EXE
Parent Program(s) : FORCEMOD.C
Child Programs(s) : NONE
Accompanying Files : COSTS.IN, COSTS.OUT, COSTS.AUX
Brief Description of Purpose : Determines costs of components in the system based on the 1995 Means Costs. Index.

Program Name: **DESIGN.C**
Executable Form : DESIGN.EXE
Parent Program(s) : FORCEMOD.C
Child Programs(s) : GSDPM3.C
Accompanying Files : UNIT.DAT, GSDPM3.AUX, GSDPM3.IN, GSDPM3.OUT, CONN.OUT, WSHD.OUT, NODES3.DAT
Brief Description of Purpose : Designs pipes in subwatershed in order: from upstream subwatersheds downwards to the downstream subwatersheds.

Program Name: **FGRID.C**
Executable Form : FGRID.EXE
Parent Program(s) : FGRID.AML
Child Programs(s) : NONE
Accompanying Files : FGRID.DAT, FGRID.OUT, FGRID.OU
Brief Description of Purpose : Generates Digital elevation grid used for shortest path analysis.

Program Name: **FORCEARR.C**
Executable Form : FORCEARR.EXE
Parent Program(s) : TESTGEN2.AML
Child Programs(s) : NONE
Accompanying Files : NODES2.DAT, FORCEMOD.DAT

Brief Description of Purpose : Arranges the subwatershed in the order in which their pipes are to be designed.

Program Name: **FORCEMOD.C**
Executable Form : FORCEMOD.EXE
Parent Program(s) : TESTGEN2.AML
Child Programs(s) : FORCEM.C, FPATH.C, DESIGN.C, PUMPSEL.FOR, COSTS.C
Accompanying Files : GSDPM3.AUX, UNIT.DAT, FORCEMOD.DAT, NODES2.DAT, FORCENOD.DAT, FORCEM.IN, FORCEM.OUT, FPATH.IN, FPATH.OUT, NODES3.DAT, WHSD.OUT, DESIGN.OUT, PUMPSEL.IN, PUMPSEL.OUT, COSTS.IN, COSTS.OUT, FORCE.OUT, FORCEMOD.OUT, FLARE.DAT.
Brief Description of Purpose : Mainly coordinates optimal force main path determination, pump selection, and wetwell design.

Program Name: **FPATH.C**
Executable Form : FPATH.EXE
Parent Program(s) : FORCEMOD.C
Child Programs(s) : INTGEN.C, SHORTP.C
Accompanying Files : FPATH.IN, FPATH.OUT, NODES2.DAT, INTSECT.IN, FGRID.DAT, PNODES.DAT, SHORTP.IN, BOUND.DAT, SHORTP.OUT
Brief Description of Purpose : Determines optimal force main path and destination.

Program Name: **GENSHTP4.C**
Executable Form : GENSHTP.EXE
Parent Program(s) : TESTGEN2.AML
Child Programs(s) : NONE
Accompanying Files : NODES.DAT, GENSHTP.DAT, GENSHTP.AUX, GSDPM3.AUX, GENNUM.DAT, GENSHTP.ARC, GENSHTP1.ARC, UNIT.DAT
Brief Description of Purpose : Does subwatershed delineation and pump station location determination.

Program Name: **INTGEN.C**
Executable Form : INTGEN.EXE
Parent Program(s) : FPATH.C
Child Programs(s) : NONE
Accompanying Files : PNODES.DAT, INTSECT.OUT, PERSGRID.OUT
Brief Description of Purpose : Used to delete lines passing through prohibited areas.

Program Name: **SHORTP.C**
Executable Form : SHORTP.EXE
Parent Program(s) : FPATH.C
Child Programs(s) : NONE
Accompanying Files : SHORTP.IN, UNIT.DAT, SHORTP.OUT, INTSECT.OUT
Brief Description of Purpose : Shortest path determination program based on Dijkstra's algorithm.

Program Name: **COMPILE.C**
Executable Form : COMPILE.EXE
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : ALL COMPILABLE FILES
Brief Description of Purpose : Compiles all compilable files.

AML PROGRAMS

Program Name: **BOUND.AML**
Executable Form : BOUND.AML
Parent Program(s) : START.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Generates buffers around prohibited area polygons.

Program Name: **FGRID.AML**
Executable Form : FGRID.AML
Parent Program(s) : START.AML
Child Programs(s) : FGRID.C
Accompanying Files : FGRID.DAT, FGRID.OU
Brief Description of Purpose : Generates grid used in shortest path determination.

Program Name: **GRADISP.AML**
Executable Form : GRADISP.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : EDITC.MENU, BACKC.MENU
Brief Description of Purpose : Used for graphic display of results.

Program Name: **PLAT.AML**
Executable Form : PLAT.AML
Parent Program(s) : START.AML
Child Programs(s) : NONE
Accompanying Files : POLY.DAT
Brief Description of Purpose : Extracts coordinates of prohibited area polygons for later use in deleting links that pass throug them.

Program Name: **POLY.AML**
Executable Form : POLY.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Miscellaneous.

Program Name: **PROFILE.AML**
Executable Form : PROFILE.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Determines profile of a given line by superimposing it on a TIN represent the terrain. Works in ARCPLOT.

Program Name: **PROFILE2.AML**
Executable Form : PROFILE2.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Extract information from an INFO file and prints it to an ASCII file.

Program Name: **START.AML**
Executable Form : START.AML
Parent Program(s) : NONE
Child Programs(s) : BUTTON.MENU, BOUND.AML, FGRID.AML, GSDPM3.MENU
Accompanying Files : UNIT.DAT
Brief Description of Purpose : Main startup file which calls all other programs.

Program Name: **TEMP.AML**
Executable Form : TEMP.AML
Parent Program(s) : TESTGEN2.AML
Child Programs(s) : NONE
Accompanying Files : GENSHTP.ARC, GENSHTP1.ARC
Brief Description of Purpose : Works with TESTGEN2.AML to display results.

Program Name: **TESTGEN1.AML**
Executable Form : TESTGEN1.AML
Parent Program(s) : START.AML
Child Programs(s) : TESTGEN2.AML, TESTGEN2A.AML
Accompanying Files : NONE
Brief Description of Purpose : Preliminary processing.

Program Name: **TESTGEN2.AML**
Executable Form : TESTGEN2.AML
Parent Program(s) : TESTGEN1.AML
Child Programs(s) : GENSHTP.C, FORCEARR.C, FORCEMOD.C
Accompanying Files : GENSHTP.DAT, GENSHTP.AUX, NODES.DAT
Brief Description of Purpose : Further processing.

Program Name: **TESTGEN2A.AML**
Executable Form : TESTGEN2A.AML
Parent Program(s) : TESTGEN1.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Further processing.

Program Name: **TEXTDISP.AML**
Executable Form : TEXTDISP.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : FORCEMOD.OUT
Brief Description of Purpose : Used for text display of results.

Program Name: **WPLOT.AML**

Executable Form : WPLOT.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Miscellaneous. An example of a plot file used in ARCPLOT

Program Name: **WYATTDXF.AML**
Executable Form : WYATTDXF.AML
Parent Program(s) : NONE
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Used to convert DXF file to ARC/INFO coverage.

MENU PROGRAMS

Program Name: ***BACKC.MENU***
Executable Form : BACKC.MENU
Parent Program(s) : GRADISP.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Used to specify the desired attributes of the backcover of the display.

Program Name: ***BUTTON.MENU***
Executable Form : BUTTON.MENU
Parent Program(s) : START.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : The first menu that comes up on startup. Used to specify files to be used, grid spacing, etc.

Program Name: ***EDITC.MENU***
Executable Form : EDITC.MENU
Parent Program(s) : GRADISP.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Used to specify the desired attributes of the editcover of the display.

Program Name: ***GSDPM3.MENU***
Executable Form : GSDPM3.MENU
Parent Program(s) : START.AML
Child Programs(s) : NONE
Accompanying Files : NONE
Brief Description of Purpose : Used to specify the parameters to be used in the GSDPM3 program.

FORTRAN PROGRAMS

Program Name: ***PUMPSEL.FOR***
Executable Form : PUMPSEL.EXE
Parent Program(s) : FORCEMOD.C
Child Programs(s) : NONE
Accompanying Files : PUMPSEL.IN, PUMPSELCOSTS.IN, COSTS.OUT, COSTS.AUX
Brief Description of Purpose : Selects appropriate pump based on procedure by Douglas et. al

COSTS

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
FILE *inp1,*inp2,*out;
int i,life,outlet,j,a,n,m,k,pts,b,t,fr;
float year,gencost,area1,area2,mat1vol,mat2vol,mat1unitcost;
float mat2unitcost,ci,mat1cost,mat2cost;
float flength,pipecost,wellcost,frac,cost_php,thick;
float outdiam,diam,trenchcost,rdepth,trenchvol,trenchunitcost;
float depth,width,cityidx,histidx,inf_rate,tr,tc,tf,powercost;
float totpumpcost,cap,depthbelow,d_mat1,pumpcost,cost,pi;
float totpower,pumppower,systemcost;

inp1=fopen("costs.in","r");
inp2=fopen("costs.aux","r");
out=fopen("costs.out","w");
fscanf(inp1,"%f",&pumppower);
fscanf(inp1,"%f",&diam);
fscanf(inp1,"%f",&cap);
fscanf(inp1,"%f",&flength);

fscanf(inp2,"%d",&life);
fscanf(inp2,"%f",&trenchunitcost);
fscanf(inp2,"%f",&mat1unitcost);
fscanf(inp2,"%f",&mat2unitcost);
fscanf(inp2,"%f",&rdepth);
fscanf(inp2,"%f",&thick);
fscanf(inp2,"%f",&tr);
fscanf(inp2,"%f",&tf);
fscanf(inp2,"%f",&tc);
fscanf(inp2,"%f",&cost_php);
fscanf(inp2,"%f",&inf_rate);
fscanf(inp2,"%f",&cityidx);
fscanf(inp2,"%f",&histidx);

/*
Trenching based on 1995 means BCCD page 46, depths up to 12ft
unitcost for up to 6ft (assuming heavy soil,or clay) = 42 per cu yard
unitcost for up to 12ft (assuming heavy soil,or clay) = 55.50 per cu
yard
Assuming pipe = 1 inch thick
For Bedding, using Class C part2 from ASCE
concrete(mat1) and granular (mat2)
*/
pi=acos(-1);
outdiam=(diam+2*thick)/12.0;
width=outdiam+8/12.0;
depth= rdepth+outdiam;
trenchvol=width*depth*flength/9;
ci=cityidx/100*histidx/100;
trenchcost=trenchvol*trenchunitcost*ci;

/*
Bedding
*/
depthbelow=outdiam/8.0;
```

```
if(depthbelow<4/12.0)
  depthbelow=4/12.0;

frac=1/6.0;

d_mat1=outdiam*frac+depthbelow;

/* Area of segment */
area1=(asin(1-frac)/4-0.5*sqrt(2*frac-frac*frac)*(1-
frac))*pow(outdiam,2);
area2=pi*pow(outdiam,2)/8.0-area1;
mat1vol=(width*d_mat1-area1)*flength/9;
mat2vol=((width*(1-frac)*outdiam-
(area2+pi*pow(outdiam,2)/8))+rdepth*width)*flength/9;
mat1cost=mat1vol*mat1unitcost*ci;
mat2cost=mat2vol*mat2unitcost*ci;
/*
pump
1 hp = 1.341 KW
$ 0.75 per KWh = 1.00575 per hph
time = 30 year;
run tr/tc fraction of cycle time
2.5% inflation rate
*/
totpower=pumppower*tr/tc*24*365;
cost=0;
for (i=1;i<=life;i++) {
year=cost_php*pow((1+inf_rate/100.0),i-1);
cost=cost+year;
}

powercost=cost*totpower*ci;
pumpcost=cap*49.625+37375;
pumpcost=pumpcost*ci;
gencost=-
0.0044*pow(pumppower,3)+1.3369*pow(pumppower,2)+18.608*pumppower+770.55;
gencost=gencost*ci;
totpumpcost=gencost+powercost+pumpcost;

/* Wetwell */
wellcost=5.1282*cap+1615.4;
wellcost=wellcost*ci;
/* Pipe cost (PVC) */
pipecost=flength*(0.0611*pow(diam,2) - 0.2047*diam+2.2088);
pipecost=pipecost*ci;
systemcost=totpumpcost+wellcost+pipecost+trenchcost+mat1cost+mat2cost;

fprintf(out,"%f\n",trenchcost);
fprintf(out,"%f\n",mat1cost);
fprintf(out,"%f\n",mat2cost);
fprintf(out,"%f\n",powercost);
fprintf(out,"%f\n",pumpcost);
fprintf(out,"%f\n",gencost);
fprintf(out,"%f\n",wellcost);
fprintf(out,"%f\n",pipecost);
fprintf(out,"%f\n",systemcost);

/*

printf("%f trecost\n",trenchcost);
```

```
printf("%f mat1cost\n",mat1cost);
printf("%f mat2cost\n",mat2cost);
printf("%f powercost\n",powercost);
printf("%f pumpcost\n",pumpcost);
printf("%f gencost\n",gencost);
printf("%f wellcost\n",wellcost);
printf("%f pipecost\n",pipecost);
printf("%f syscost\n",systemcost);
*/
}
```

DESIGN

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
struct card {
int n,pn[430],umh[430],lmh[430];float finv,finel,cov,ppc,qpp,gpf,meth;
float cost,len[430],con[430],icon[430],el[430],pf[430];
} wshed[50];
struct card1 {int id,fn,tn,number,b;float x1,x2,y1,y2,el,len,con,icon;}
l[500];
struct card2 {int num,b,p;float elev,x,y,con,icon;} nd[500];
struct card3 {int frw,tow,frnd,tond;} ws[300];
main()
{
FILE *inp1,*inp2,*inp3,*inp4,*out,*out1,*aux,*unit;
int wshds,outfall,n,a,c,d,b,nds,e,num,legs,k,units;
float p,dx,dy,gsdpm3;
float cov,ppc,qpp,meth,gpf,con;
float manns,stand,meansidx,year,infilt,matrl,cityidx;
char input[30],output[30];
/*** Reading of auxilary file    ***/
aux=fopen("gsdpm3.aux","r");
unit=fopen("unit.dat","r");
fscanf(aux,"%f",&cov);fscanf(aux,"%f",&ppc);
fscanf(aux,"%f",&qpp);fscanf(aux,"%f",&meth);
fscanf(aux,"%f",&gpf);fscanf(aux,"%f",&manns);
fscanf(aux,"%f",&stand);fscanf(aux,"%f",&meansidx);
fscanf(aux,"%f",&year);fscanf(aux,"%f",&infilt);
fscanf(aux,"%f",&matrl);fscanf(aux,"%f",&cityidx);
fscanf(unit,"%d",&units);
/*** End of auxilary file  ***/
inp2=fopen("nodes3.dat","r");
inp1=fopen("conn.out","r");
inp3=fopen("wshd.out","r");
fscanf(inp3,"%d",&wshds);
for (a=1;a<=wshds;a++) {
fscanf(inp3,"%d",&ws[a].frw);fscanf(inp3,"%d",&ws[a].tow);
fscanf(inp3,"%d",&ws[a].frnd);fscanf(inp3,"%d",&ws[a].tond);
}

fscanf(inp1,"%d",&legs);
for (a=1;a<=legs;a++) {
fscanf(inp1,"%d",&l[a].id);fscanf(inp1,"%d",&l[a].fn);
fscanf(inp1,"%d",&l[a].tn);fscanf(inp1,"%d",&l[a].number);
fscanf(inp1,"%d",&l[a].b);
}
fscanf(inp2,"%d",&nds);
for (a=1;a<=nds;a++) {
fscanf(inp2,"%d",&nd[a].num);fscanf(inp2,"%d",&nd[a].b);
fscanf(inp2,"%d",&nd[a].p);fscanf(inp2,"%f",&nd[a].elev);
fscanf(inp2,"%f",&nd[a].x);fscanf(inp2,"%f",&nd[a].y);
fscanf(inp2,"%f",&nd[a].con);fscanf(inp2,"%f",&nd[a].icon);
if(units==1) {
nd[a].elev=nd[a].elev/.3048;nd[a].x=nd[a].x/.3048;nd[a].y=nd[a].y/.3048;
}
}
for (k=1;k<=legs;k++) {
 for (a=1;a<=nds;a++) {
```

```
   if (l[k].fn == nd[a].num) {
    l[k].el = nd[a].elev;l[k].x1 = nd[a].x;
    l[k].y1 = nd[a].y;l[k].con = nd[a].con;
    l[k].icon = nd[a].icon;}
   if (l[k].tn == nd[a].num)
    {l[k].x2 = nd[a].x;
    l[k].y2 = nd[a].y;}
  }
  dx=l[k].x2-l[k].x1;dy=l[k].y2-l[k].y1;
  l[k].len=sqrt(dx*dx+dy*dy);
 }
 b=1;for (k=1;k<=legs;k++) {if (l[k].b>b) {b=l[k].b;}}
 c=0;
 for (k=1;k<=b;k++) {
  num=1;outfall=0;
  for (a=1;a<=legs;a++)
   if ((l[a].b==k)&&(l[a].number>num))
    num=l[a].number;
  for (a=1;a<=nds;a++)
   if ((nd[a].b==k)&&(nd[a].p==1))
     outfall=a;
  wshed[k].n=num+1;
  wshed[k].finv=nd[outfall].elev-(cov+1);
 wshed[k].finel=nd[outfall].elev;
  wshed[k].cov=cov; wshed[k].ppc=ppc; wshed[k].qpp=qpp;
  wshed[k].gpf=gpf; wshed[k].meth=meth; e=num+1;
  for (a=1;a<=num;a++) {
   for (c=1;c<=legs;c++) {
    if ((l[c].number==a)&&(l[c].b==k)) {
     wshed[k].pn[a]=a;
     wshed[k].umh[a]=l[c].fn;
     wshed[k].lmh[a]=l[c].tn;
     wshed[k].len[a]=l[c].len;
     wshed[k].con[a]=l[c].con;
     wshed[k].icon[a]=l[c].icon;
     wshed[k].el[a]=l[c].el;
     wshed[k].pf[a]=gpf;
    }
   }
  }
  wshed[k].pn[e]=e;
  wshed[k].umh[e]=wshed[k].lmh[num];
  wshed[k].lmh[e]=9999;
  wshed[k].len[e]=8;
  wshed[k].con[e]=nd[outfall].con;
  wshed[k].icon[e]=nd[outfall].icon;
  wshed[k].el[e]=nd[outfall].elev;
  wshed[k].pf[e]=gpf;
 }


 for (d=1;d<=wshds+1;d++) {
 k=ws[d].frw;
 if (d==wshds+1)
  k=1;
  out1=fopen("gsdpm3.in","w");
  fprintf(out1,"%s%d%s\n","subwat",k,".dat");
  fprintf(out1,"%s%d%s\n","subwat",k,".out");
  fclose(out1); out1=fopen("gsdpm3.in","r");
```

Appendix D:

```
 fscanf(out1,"%s",&input); fscanf(out1,"%s",&output);
 fclose(out1);

 out=fopen(input,"w");
 fprintf(out,"%s %d\n","Data for Subwatershed",k);
 fprintf(out,"%-10.4f%-10.0f%-10.1f%-10.0f%-10.1f%-10.0f%-
10.1f\n",manns,stand,meansidx,year,infilt,matrl,cityidx);
 fprintf(out,"%-10d%-10.2f%-10.2f%-10.1f%-10.1f%-10.1f%-10.0f%-
10.1f\n",wshed[k].n,wshed[k].finv,wshed[k].finel,wshed[k].cov,wshed[k].p
pc,wshed[k].qpp,wshed[k].gpf,wshed[k].meth);
 num=wshed[k].n;
 for (a=1;a<=num;a++) {
  fprintf(out,"%-10d%-10d%-10d%-10.2f%-10.2f%-10.2f%-10.2f%-
10.1f\n",wshed[k].pn[a],wshed[k].umh[a],wshed[k].lmh[a],wshed[k].len[a],
wshed[k].con[a],wshed[k].icon[a],wshed[k].el[a],wshed[k].pf[a]);
 }
 fclose(out);
 system("gsdpm3.exe");
 inp4=fopen("gsdpm3.out","r");
 fscanf(inp4,"%f",&wshed[k].cost);
 fclose(inp4);
 if (k==1)
  goto hey;

hey:;
fclose(out1);
}
out1=fopen("design.out","w");
for (k=1;k<=b;k++)
  fprintf(out1,"%f\n",wshed[k].cost);

fclose(inp1);fclose(inp2);fclose(inp3);fclose(out);
}
```

FGRID

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
FILE *inp,*out,*out1;
int a,b,c,n;
float dx,dy,xmin,ymin,xmax,ymax,x,y,offset,incr,d1,d2;
inp=fopen("fgrid.dat","r");
out=fopen("fgrid.out","w");
out1=fopen("fgrid.ou","w");
fscanf(inp,"%d",&n);
fscanf(inp,"%f",&xmin);
fscanf(inp,"%f",&ymin);
fscanf(inp,"%f",&xmax);
fscanf(inp,"%f",&ymax);
dx = xmax - xmin;
dy = ymax - ymin;
d2 = dx;
if (dy < dx)
  d2 = dy;
offset = d2 / (n*2);

xmin = xmin + offset;
ymin = ymin + offset;
xmax = xmax - offset;
ymax = ymax - offset;

dx = xmax - xmin;
dy = ymax - ymin;
d1=dy;
if (dy < dx)
  d1 = dx;
incr = d1/n;
x = xmin-incr;
c=0;
for (a=1;a<=n;a++) {
 x = x + incr;
 y = ymin - incr;
 for (b=1;b<=n;b++) {
  y = y + incr;
  if ((x<=xmax)&&(y<=ymax)) {
    c++;
    fprintf(out,"%d %15.4f %15.4f\n",c,x,y);
  }
 }
}
fprintf(out,"END\n");
fprintf(out1,"%12.8f\n",incr);
fclose(out);
fclose(inp);
fclose(out1);
}
```

FORCEARR

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
struct card2 {int id,next,wshed,out;float x,y,elev,con;} node[2000];
main()
{
FILE *inp1,*out;int a,c,k,b,m,t,fr,travel[2000];
int pts,wshed,outnum,dest;float     len,q[100][100],dm,dmin1,dx,dy;
inp1=fopen("nodes2.dat","r");out=fopen("forcemod.dat","w");
fscanf(inp1,"%d",&pts);
for (a=1;a<=pts;a++) {
fscanf(inp1,"%d",&node[a].id);fscanf(inp1,"%d",&node[a].wshed);
fscanf(inp1,"%d",&node[a].out);
fscanf(inp1,"%f",&node[a].elev);
fscanf(inp1,"%f",&node[a].x);
fscanf(inp1,"%f",&node[a].y);
fscanf(inp1,"%f",&node[a].con);
}
/* determine # of subwatersheds */
wshed=0;
for(a=1;a<=pts;a++){if(node[a].wshed>wshed){wshed=node[a].wshed;}}
/* End determine # of subwatersheds */

for (a=1;a<=wshed;a++){
 travel[a] = 0; for(k=1;k<=wshed;k++){q[a][k]=999999;}
}

/* determine distance from outlet to nearest of subwatersheds */
for (b=2;b<=wshed;b++) {
 for (a=1;a<=pts;a++)
  if ((node[a].wshed==b)&&(node[a].out==1)) {outnum=a;break;}

 for (c=1;c<=wshed;c++) {dm=9999999;
  for (a=1;a<=pts;a++)
   if (c!=b)
    if (node[a].wshed==c) {
     dx=node[outnum].x-node[a].x;dy=node[outnum].y-node[a].y;
     len=sqrt(pow(dx,2)+pow(dy,2));
     if (len<dm) {dm=len;dest=a;}
    }

  if (c!=b)
   q[node[outnum].wshed][node[dest].wshed]=dm;
 }
}

/* End determine distance from outlet to nearest of subwatersheds */

/* determine order of linking */
travel[1] = 1;
for (k=1;k< wshed;k++) {
dmin1 = 1000000000;
for(m=1;m<=wshed;m++)
 if(travel[m] == 1)
  for(a=1;a<=wshed;a++)
   if((travel[a]==0)&&(q[a][m]!=999999)&&(dmin1>q[a][m]))
    {dmin1 = q[a][m];t = m;fr = a;}
travel[fr] = 1;
```

```
fprintf(out,"%d %d \n", fr, t);
}
fprintf(out,"%d \n",0);
/* End determine order of linking */
fclose(out);fclose(inp1);
}
```

FORCEMOD

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
struct loop1 {
 struct card {
  int b,num,f,t,next,path[1600],frmanh,tomanh,nodesalong,number;
  float
dirf,cumf,diam,length,eqlength,gsdpm3_cost,my_cost,ftrenchcost,fbedcost,
x[400],y[400],z[400];
  float fbackfillcost,
powercost,pumpcost,gencost,wellcost,pipecost,maxelev,initelev,TDH;
  struct wetw{
   float
ctime,len,fill_d,minsub,clear,suc_d,flare_d,suc_v,wetfloor,s_volume,t_vo
lume,depth,stath,thead;
  }wetwell;
  struct pumps {
   float num,power,impdiam,speed,req_head,av_head, effeciency;
  } pump;
 } nd[80];
float totalcost;
struct lnk {
 int b,num,f,t,number,trav;
   float dirf,cumf;
 } link[80];

struct card1 {
 int b,num,p;
 float elev,x,y,con,icon;
} node[1600];
} loop[5];

int lp, number, g,cheap;
main()
{
void wetwell(int lp,int k,float ctime,float Qi,float cov,float L,float
W,float Slope, float forced,float forcev, float dIS,float dSF);
void print(int wsheds);
void up(int z, int legs);
FILE *inp,*inp1,*inp2,*inp3,*out,*aux,*out2,*fce,*unit;
int check,val,i,j,k,a,b,c,d,e,ndb,nodes,nds,id,temp,units;
float pi,cov, ppc, qpp, meth, gpf, manns, stand, meansidx, year, infilt,
matrl, cityidx, ctime,len;
float head,flo,eql,dia,dx,dy,dz,cost,maxelev;
float L=10,W=10,Slope=1,forcev=4,dIS=0.5,dSF=1;
pi=acos(-1);
aux=fopen("gsdpm3.aux","r");
unit=fopen("unit.dat","r");
fscanf(unit,"%d",&units);

fscanf(aux,"%f",&cov);fscanf(aux,"%f",&ppc);
fscanf(aux,"%f",&qpp);fscanf(aux,"%f",&meth);
fscanf(aux,"%f",&gpf);fscanf(aux,"%f",&manns);
fscanf(aux,"%f",&stand);fscanf(aux,"%f",&meansidx);
fscanf(aux,"%f",&year);fscanf(aux,"%f",&infilt);
fscanf(aux,"%f",&matrl);fscanf(aux,"%f",&cityidx);
fscanf(aux,"%f",&ctime);fscanf(aux,"%f",&len);
```

```
inp3=fopen("forcemod.dat","r");
check=9999;
lp=1;
while(check!=0) {
/**** reading of nodes data ***/
inp=fopen("nodes2.dat","r");
fscanf(inp,"%d",&nodes);
for (a=1;a<=nodes;a++) {
fscanf(inp,"%d",&loop[lp].node[a].num);fscanf(inp,"%d",&loop[lp].node[a]
.b);
fscanf(inp,"%d",&loop[lp].node[a].p);fscanf(inp,"%f",&loop[lp].node[a].e
lev);
fscanf(inp,"%f",&loop[lp].node[a].x);fscanf(inp,"%f",&loop[lp].node[a].y
);
fscanf(inp,"%f",&loop[lp].node[a].con);
loop[lp].node[a].icon=0;
}
fclose(inp);
/**** end of reading of nodes data ***/

/***** Determination of Number of Subwatersheds *****/
b=1;
for (a=1;a<=nodes;a++)
 if (loop[lp].node[a].b>b)
  b=loop[lp].node[a].b;
/** End of Determination of Number of Subwatersheds */

/*** Summation of subwatershed flows ***************/
inp2=fopen("forcenod.dat","w");
fprintf(inp2,"%d\n",b);
for (a=1;a<=b;a++) {
 for (e=1;e<=nodes;e++)
  if (loop[lp].node[e].b==a)
   loop[lp].nd[a].dirf=loop[lp].nd[a].dirf+loop[lp].node[e].con;
fprintf(inp2,"%d %10.4f\n",a, loop[lp].nd[a].dirf*ppc*qpp/24/60);
}
fclose(inp2);

/*** End of Summation of subwatershed flows *********/


/**** reading of super nodes & flow ****/
inp1=fopen("forcenod.dat","r");
fscanf(inp1,"%d",&nds);
for (a=1;a<=nds;a++) {
fscanf(inp1,"%d",&loop[lp].nd[a].num);
fscanf(inp1,"%f",&loop[lp].nd[a].dirf);
loop[lp].nd[a].cumf=loop[lp].nd[a].dirf;
loop[lp].nd[a].next=0;
loop[lp].nd[a].number=0;
}
fclose(inp1);
/**** end of reading of super nodes & flow ****/

/**** reading of links of supernodes ****/
for (a=1;a<nds;a++) {
fscanf(inp3,"%d",&loop[lp].link[a].f);fscanf(inp3,"%d",&loop[lp].link[a]
.t);
loop[lp].link[a].b=0;loop[lp].link[a].trav=0;loop[lp].link[a].number=0;
```

```
}
/**** end of reading of links of supernodes ****/

/**** link rearrangement       ****/
number=0; for (a=1;a<nds;a++) {up(1,nds-1);}

/**** end of link rearrangement****/

/****flow accumulation*****/
for (a=1;a<nds;a++)
 for (d=1;d<nds;d++)
  if (loop[lp].link[d].number==a)
   for (b=2;b<=nds;b++) {
    if (loop[lp].nd[b].num==loop[lp].link[d].f) {
     for (c=1;c<=nds;c++)
      if (loop[lp].nd[c].num==loop[lp].link[d].t) {
       loop[lp].nd[c].cumf=loop[lp].nd[c].cumf+loop[lp].nd[b].cumf;
       loop[lp].nd[b].next=loop[lp].nd[c].num;
       loop[lp].nd[b].number=loop[lp].link[d].number;
      }
    }
   }

/*** end of flow accumulation****/
if(a>=0)
printf("bef forcem\n");

/****  forcemain sizing      ****/
for (a=2;a<=nds;a++) {
inp1=fopen("forcem.in","w");
fprintf(inp1,"%10.2f\n",loop[lp].nd[a].cumf*2*2.23/1000);
fclose(inp1);
system("forcem.exe");
inp1=fopen("forcem.out","r");
fscanf(inp1,"%f",&loop[lp].nd[a].diam);
fclose(inp1);
}
/***end of forcemain sizing ****/

if(a>=0)
printf("bef fpath\n");
/*** Forcemain Path Determination ****/
for (a=2;a<=nds;a++) {
 inp1=fopen("fpath.in","w");
 fprintf(inp1,"%d %d
%f\n",loop[lp].nd[a].num,loop[lp].nd[a].next,loop[lp].nd[a].diam/12);
 fclose(inp1);
 system("fpath.exe");
 inp1=fopen("fpath.out","r");
 fscanf(inp1,"%d",&loop[lp].nd[a].nodesalong);
 fscanf(inp1,"%f",&loop[lp].nd[a].eqlength);
 maxelev=0;
 for (i=1;i<=loop[lp].nd[a].nodesalong;i++) {

fscanf(inp1,"%d",&loop[lp].nd[a].path[i]);fscanf(inp1,"%f",&loop[lp].nd[
a].x[i]);

fscanf(inp1,"%f",&loop[lp].nd[a].y[i]);fscanf(inp1,"%f",&loop[lp].nd[a].
z[i]);
  if (loop[lp].nd[a].z[i]>maxelev)
```

```
    maxelev = loop[lp].nd[a].z[i];
  }
  loop[lp].nd[a].initelev = loop[lp].nd[a].z[1];
  loop[lp].nd[a].maxelev = maxelev;
  loop[lp].nd[a].TDH = maxelev - loop[lp].nd[a].z[1];
  if (units==1)
   loop[lp].nd[a].TDH = loop[lp].nd[a].TDH / 0.3048;

  temp=loop[lp].nd[a].nodesalong;
  loop[lp].nd[a].frmanh=loop[lp].nd[a].path[1];
  loop[lp].nd[a].tomanh=loop[lp].nd[a].path[temp];

  /*** begin icon ****/
  for (k=1;k<=nodes;k++) {
   if (loop[lp].node[k].num==loop[lp].nd[a].tomanh)
    loop[lp].node[k].icon=loop[lp].nd[a].cumf*24*60/ppc/qpp;
  }
  /*** end of icon ****/



  /****  Determination of forcemain length ***/
  loop[lp].nd[a].length=0;
  for (k=1;k<loop[lp].nd[a].nodesalong;k++) {
   dx=loop[lp].nd[a].x[k+1]-loop[lp].nd[a].x[k];
   dy=loop[lp].nd[a].y[k+1]-loop[lp].nd[a].y[k];
   dz=loop[lp].nd[a].z[k+1]-loop[lp].nd[a].z[k];
   loop[lp].nd[a].length=loop[lp].nd[a].length + sqrt(pow(dx,2) +
pow(dy,2) + pow(dz,2));
  }
   if (units==1)
    loop[lp].nd[a].length=loop[lp].nd[a].length/.3048;
  /****  End of Determination of forcemain length ***/


}
if(a>=0)
printf("aft fpath\n");
/*** End of Forcemain Path Determination ****/


out2=fopen("nodes3.dat","w");
fprintf(out2,"%d\n",nodes);

for (k=1;k<=nodes;k++) {
fprintf(out2,"%d %d %d
",loop[lp].node[k].num,loop[lp].node[k].b,loop[lp].node[k].p);
fprintf(out2,"%f %f %f
",loop[lp].node[k].elev,loop[lp].node[k].x,loop[lp].node[k].y);
fprintf(out2,"%f %f\n",loop[lp].node[k].con,loop[lp].node[k].icon);
}
fclose(out2);

printf("yes");

/****    Design by GSDPM3      *****/
out2=fopen("wshd.out","w");
fprintf(out2,"%d\n",nds-1);
for (a=1;a<nds;a++)
 for (k=2;k<=nds;k++)
```

```
  if(loop[lp].nd[k].number==a) {
   fprintf(out2,"%d %d ",loop[lp].nd[k].num,loop[lp].nd[k].next);
   fprintf(out2,"%d %d\n",loop[lp].nd[k].frmanh,loop[lp].nd[k].tomanh);
  }
fclose(out2);
system("design.exe");


/** Read Costs for each subwatershed network **/
out2=fopen("design.out","r");
for (a=1;a<=nds;a++)
  fscanf(out2,"%f",&loop[lp].nd[a].gsdpm3_cost);
fclose(out2);
/** End Read Costs for each subwatershed network **/


/**** End of Design by GSDPM3 *****/
if(a>=0)
printf("aft design\n");
/**** Design of Wetwell *****/
for (a=1;a<nds;a++)
 for (k=2;k<=nds;k++)
  if(loop[lp].nd[k].number==a)

wetwell(lp,k,ctime,loop[lp].nd[k].cumf,cov,L,W,Slope,loop[lp].nd[k].diam
,forcev,dIS,dSF);
/**** end of Design of Wetwell *****/
if(a>=0)
printf("aft wetwell\n");

/**** Selection of pump *****/
for (a=1;a<nds;a++)
 for (k=2;k<=nds;k++)
  if(loop[lp].nd[k].number==a) {
   inp1=fopen("pumpsel.in","w");
   fprintf(inp1,"%s \n","pumpsel.out");
   dia=loop[lp].nd[k].diam/12;
   head=0.022*loop[lp].nd[k].length*pow(forcev,2)/(2*dia*32.2);
   loop[lp].nd[k].TDH = loop[lp].nd[k].TDH + head +
loop[lp].nd[k].wetwell.depth;
   loop[lp].nd[k].TDH = loop[lp].nd[k].TDH +
loop[lp].nd[k].wetwell.thead;
   head=loop[lp].nd[k].TDH;
   fprintf(inp1,"%f\n%f\n",loop[lp].nd[k].cumf,head);
   fclose(inp1);
   system("pumpsel.exe");
   inp1=fopen("pumpsel.out","r");
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.num);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.speed);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.impdiam);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.power);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.effeciency);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.req_head);
   fscanf(inp1,"%f",&loop[lp].nd[k].pump.av_head);
   fclose(inp1);

   /**********Temp Pump Power calculation ********Assuming 60% effic**/
    printf("cumf = %f\n",loop[lp].nd[k].cumf);
```

```
loop[lp].nd[k].pump.power=62.4*((loop[lp].nd[k].cumf*.00223)*2)*loop[lp]
.nd[k].TDH/550/.6;
   printf("power  %f\n",loop[lp].nd[k].pump.power);
   loop[lp].nd[k].pump.req_head=loop[lp].nd[k].TDH;
   /**********  End Temp Pump Power calculation ********Assuming 60%
effic**/


  }
/**** end of selection of pump *****/

/*** Determination of costs for forcemain, trenching & pump  ***/
for (a=2;a<=nds;a++) {
 out2=fopen("costs.in","w");
 fprintf(out2,"%f\n",loop[lp].nd[a].pump.power);
 fprintf(out2,"%f\n",loop[lp].nd[a].diam);
 fprintf(out2,"%f\n",loop[lp].nd[a].cumf);
 fprintf(out2,"%f\n",loop[lp].nd[a].length);
 fclose(out2);
 system("costs.exe");
 out2=fopen("costs.out","r");
 fscanf(out2,"%f",&loop[lp].nd[a].ftrenchcost);
 fscanf(out2,"%f",&loop[lp].nd[a].fbedcost);
 fscanf(out2,"%f",&loop[lp].nd[a].fbackfillcost);
 fscanf(out2,"%f",&loop[lp].nd[a].powercost);
 fscanf(out2,"%f",&loop[lp].nd[a].pumpcost);
 fscanf(out2,"%f",&loop[lp].nd[a].gencost);
 fscanf(out2,"%f",&loop[lp].nd[a].wellcost);
 fscanf(out2,"%f",&loop[lp].nd[a].pipecost);
 fscanf(out2,"%f",&loop[lp].nd[a].my_cost);
 fclose(out2);
}
/*** End determination of costs for forcemain, trenching & pump  ***/


/***  Summation of Costs for Each configuration  *****/
loop[lp].totalcost=0;
for (a=2;a<=nds;a++) {
 loop[lp].totalcost = loop[lp].totalcost+loop[lp].nd[a].my_cost +
loop[lp].nd[a].gsdpm3_cost;
}
 loop[lp].totalcost = loop[lp].totalcost + loop[lp].nd[1].gsdpm3_cost;



/***  End of Summation of Costs for Each configuration  *****/

fscanf(inp3,"%d",&check);
printf("For loop[%d], Total cost = %f \n",lp, loop[lp].totalcost);
lp++;

/*** End of While ****/

}

cost=loop[1].totalcost;
cheap=1;
for (a=2;a<lp;a++)
 if (loop[a].totalcost<cost) {
```

```
   cheap=a;
    cost=loop[a].totalcost;
   }
g=cheap;


/*** Create Force main drawing File ***/
fce=fopen("force.out","w");
for (a=2;a<=nds;a++) {
 fprintf(fce,"%d\n",a-1);
 for (k=1;k<=loop[g].nd[a].nodesalong;k++) {
  fprintf(fce,"%f  %f\n",loop[g].nd[a].x[k],loop[g].nd[a].y[k]);
 }
 fprintf(fce,"END\n");
}
 fprintf(fce,"END\n");
fclose(fce);
/*** End Create Force main drawing File ***/



print(nds);
/*** End of Program ****/
}

void up(int z, int legs)
{
void flip(int p);
int f=0,i=1;
while (i<=legs)
{if ((loop[lp].link[i].f==z)&&(loop[lp].link[i].b==0)) {flip(i);break;}
i++;}i=1;
while (i<=legs) {
if ((loop[lp].link[i].t==z)&&(loop[lp].link[i].trav!=1))
{z=loop[lp].link[i].f;loop[lp].link[i].b=1;f=1;break;}i++;}i=1;
if (f==0)
while (i<=legs) {if (loop[lp].link[i].f==z)
{loop[lp].link[i].trav=1;number++;loop[lp].link[i].number=number;break;}
i++;}
if (f==1) {up(z,legs);}
}


void print(int wsheds)
{
FILE *out2;
int i,j,k;float dia;
out2=fopen("forcemod.out","w");
for (i=1;i<lp;i++) {
fprintf (out2,"\n\n***********   Begin connectivity #%d
***********\n",i);
fprintf (out2,"\nFor subwatershed connectivity #%d, the general costs
are summarised \n as follows:\n",i);
 /* printf(" */
fprintf(out2,"\nFor subwatershed 1\n\n");
fprintf(out2,"Costs of pipes and excavation in subwatershed = $
%10.2f\n",loop[i].nd[1].gsdpm3_cost);

for (j=2;j<=wsheds;j++) {
fprintf(out2,"\nFor subwatershed[%d%]\n\n",j);
```

```
 fprintf(out2,"Waste from subwatershed %d is pumped to manhole #%d in
subwatershed
%d\n",loop[i].nd[j].num,loop[i].nd[j].tomanh,loop[i].nd[j].next);
 fprintf(out2,"Forcemain trenching cost                          = $
%10.2f\n",loop[i].nd[j].ftrenchcost);
 fprintf(out2,"Forcemain Bedding cost                            = $
%10.2f\n",loop[i].nd[j].fbedcost);
 fprintf(out2,"Forcemain pipe installation cost                  = $
%10.2f\n",loop[i].nd[j].pipecost);
 fprintf(out2,"Forcemain backfill cost                           = $
%10.2f\n",loop[i].nd[j].fbackfillcost);
 fprintf(out2,"Pump power consumption cost(per year)             = $
%10.2f\n",loop[i].nd[j].powercost);
 fprintf(out2,"Pump cost                                         = $
%10.2f\n",loop[i].nd[j].pumpcost);
 fprintf(out2,"Pump Motor cost                                   = $
%10.2f\n",loop[i].nd[j].gencost);
 fprintf(out2,"Wetwell cost                                      = $
%10.2f\n",loop[i].nd[j].wellcost);
 fprintf(out2,"Costs of pipes and excavation in subwatershed = $
%10.2f\n",loop[i].nd[j].gsdpm3_cost);
dia=loop[i].nd[j].gsdpm3_cost+loop[i].nd[j].wellcost+loop[i].nd[j].pumpc
ost+loop[i].nd[j].gencost+loop[i].nd[j].powercost+loop[i].nd[j].fbackfil
lcost+loop[i].nd[j].pipecost+loop[i].nd[j].fbedcost+loop[i].nd[j].ftrenc
hcost;
fprintf(out2,"\n Total cost for subwatershed %d is $ %10.2f\n",j,dia);

}

fprintf(out2,"\nTotal system cost for the %d subwatersheds is $
%10.2f",wsheds,loop[i].totalcost);
}
fprintf(out2,"\n Out of the %d different connectivities, connectivity
#%d produces the \n",lp-1,cheap);
fprintf(out2,"Least Cost Overall Design\n");

fprintf(out2,"\n\nFor subwatershed[%d%]\n\n",1);
 fprintf(out2,"Total Flow at Outfall  = %10.2f
gal/min\n",loop[g].nd[1].cumf);

for (j=2;j<=wsheds;j++) {
fprintf(out2,"\nFor subwatershed[%d%]\n\n",j);
 fprintf(out2,"Waste from subwatershed %d is pumped to manhole #%d in
subwatershed
%d\n",loop[g].nd[j].num,loop[g].nd[j].tomanh,loop[g].nd[j].next);
 fprintf(out2,"Forcemain Size         = %10.2f
inches\n",loop[g].nd[j].diam);
 fprintf(out2,"Forcemain Length       = %10.2f
ft\n",loop[g].nd[j].length);
 fprintf(out2,"Wetwell Capacity       = %10.2f
ft^3\n",loop[g].nd[j].wetwell.s_volume);
 fprintf(out2,"Total Flow at Outfall  = %10.2f
gal/min\n",loop[g].nd[j].cumf);
 fprintf(out2,"Pump power required    = %10.2f
hp\n",loop[g].nd[j].pump.power);
 fprintf(out2,"Head on pump           = %10.2f
ft\n",loop[g].nd[j].pump.req_head);
/* fprintf(out2,"Pump Motor cost                                   = $
%10.2f\n",loop[g].nd[j].gencost);
```

```
 fprintf(out2,"Wetwell cost                                          = $
%10.2f\n",loop[g].nd[j].wellcost);
 fprintf(out2,"Costs of pipes and excavation in subwatershed = $
%10.2f\n",loop[g].nd[j].gsdpm3_cost);
*/
}



}



void flip(int p)
{int fn,tn;
tn=loop[lp].link[p].f;fn=loop[lp].link[p].t;loop[lp].link[p].f=fn;loop[l
p].link[p].t=tn;
}

void wetwell(int lp,int k,float ctime,float Qi,float cov,float L,float
W,float Slope, float forced,float forcev, float dIS,float dSF)
{
float flare(int a, float val);
float check_d2 (float d2, float qin2);
float check_d1 (float d1, float qin1);
/* Wet well variables */
float
LB,Ds,Vs,Qp,tf,V,B,Wfw,Lhf,df,Volf,ds,dc,WLWL,VolLWL,VolBL,dBL,VolFB;
float dFB,dI,dw,suclen,dislen,stath,sucfh,disfh,sucmloss,dismloss,thead;
float dia=6,pi=acos(-1);
/*********/
dI=cov+3;tf=ctime/2;Qp=Qi*2;V=Qi*tf/7.48;
/**** Design of Wetwell *****/
if ((Qp*0.41/(dia*dia))>6) {dia = check_d1(dia,Qp);}
if ((Qp*0.41/(dia*dia))<2) {dia = check_d2(dia,Qp);}
Ds=dia;
Vs=4*Qp*(2.23/1000)/(pi*(Ds/12)*(Ds/12));
ds=0.585*Vs-0.16;
B=(flare(3,Ds));
LB=flare(4,Ds);
dc=3*(B/12)/8;
Wfw=2*(B/12);
Lhf=(W-Wfw)/2;
df=Lhf*Slope;
Volf=((Wfw+W)/2*df)*L;
if (ds+dc>df) {goto hey;}
WLWL=W-2*(ds+dc)/Slope;
VolLWL=(((WLWL+Wfw)/2)*(ds+dc))*L;
VolBL=Volf-VolLWL;
dBL=df-(ds+dc);
VolFB=V-VolBL;
hey:;
if (ds+dc>df) {VolFB=V-Volf;}
dFB=VolFB/(L*W);
if (dFB<0) {dFB=0;}
dw=dI+dIS+dSF+dFB+dBL+ds+dc;
if ((ds+dc)>df) {dw=dI+dIS+dSF+dFB+df;}
suclen=LB/12+Ds/12/2+W/2+5;
```

```
dislen=2+(dI+dIS+dSF+dFB+df-(dc+LB/12+Ds/12/2));
stath=dI+dIS+dSF+dFB+df-(dc+ds);
disfh=.02*dislen*forcev*forcev/(2*(forced/12)*32.2);
sucfh=.02*suclen*Vs*Vs/(2*(Ds/12)*32.2);

/*Entloss k=.04 (munson 494) 90 deg bend k = .3 */
/*gate valve k = .15 check k = 2 */

sucmloss=(.04+.3+.15)*Vs*Vs/(2*32.2);
dismloss=(.15+2+.3+.3)*forcev*forcev/(2*32.2);
thead=dismloss+sucmloss+stath+disfh+sucfh;

loop[lp].nd[k].wetwell.depth = dw;
loop[lp].nd[k].wetwell.minsub = ds;
loop[lp].nd[k].wetwell.clear = dc;
loop[lp].nd[k].wetwell.suc_v = Vs;
loop[lp].nd[k].wetwell.suc_d = Ds;
loop[lp].nd[k].wetwell.s_volume=V;
loop[lp].nd[k].wetwell.stath = stath;
loop[lp].nd[k].wetwell.thead = thead;
loop[lp].nd[k].wetwell.flare_d = B;

/**** end of Design of Wetwell *****/

/*** End of File ****/
}

float check_d1 (float d1, float qin1)
{
if ((qin1*0.41/(d1*d1)) > 6)
d1=8;
if ((qin1*0.41/(d1*d1)) > 6)
d1=10;
if ((qin1*0.41/(d1*d1)) > 6)
d1=12;
if ((qin1*0.41/(d1*d1)) > 6)
d1=14;
if ((qin1*0.41/(d1*d1)) > 6)
d1=16;
if ((qin1*0.41/(d1*d1)) > 6)
d1=18;
if ((qin1*0.41/(d1*d1)) > 6)
d1=20;
if ((qin1*0.41/(d1*d1)) > 6)
d1=24;
return d1;
}


float check_d2 (float d2, float qin2)
{
if ((qin2*0.41/(d2*d2)) < 2)
d2=4;
return d2;
}

float flare(int a, float val)
{
struct f {
float size,T,R,B,L,wt;
```

```
} fl[12];
FILE *inp;
float s,t,r,b,l,w,retval;
int k;
inp=fopen("flare.dat","r");
for (k=1;k<=11;k++) {
fscanf(inp,"%f",&fl[k].size);fscanf(inp,"%f",&fl[k].T);
fscanf(inp,"%f",&fl[k].R);fscanf(inp,"%f",&fl[k].B);
fscanf(inp,"%f",&fl[k].L);fscanf(inp,"%f",&fl[k].wt);
}
for (k=1;k<=11;k++)
 if (fl[k].size==val) {
 t=fl[k].T; r=fl[k].R; b=fl[k].B; l=fl[k].L;
 w=fl[k].wt;
 }
if (a == 1)
 retval=t;
if (a == 2)
 retval=r;
if (a == 3)
 retval=b;
if (a == 4)
 retval=l;
if (a == 5)
 retval=w;
fclose(inp);
return retval;
}
```

FPATH

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct card {
int num,b,b1,p,pos;
float x,y,z,elev,con;
} nd[1800],ndbnd[1800];

main()
{
FILE *inp1,*inp,*out,*shrt,*intsect,*shortp,*fgrid;
int val,i,j,k,a,fws,tws,nds,id,nodesalong,nodes,path[1600];
float incr,check,diam,x,y,z,x1[1800],y1[1800],z1[1800],length;
inp1=fopen("fpath.in","r");
fscanf(inp1,"%d",&fws);fscanf(inp1,"%d",&tws);
fscanf(inp1,"%f",&diam);
fclose(inp1);
inp1=fopen("nodes2.dat","r");
fscanf(inp1,"%d",&nds);
for (a=1;a<=nds;a++) {
fscanf(inp1,"%d",&nd[a].num);fscanf(inp1,"%d",&nd[a].b);
fscanf(inp1,"%d",&nd[a].p);fscanf(inp1,"%f",&nd[a].elev);
fscanf(inp1,"%f",&nd[a].x);fscanf(inp1,"%f",&nd[a].y);
fscanf(inp1,"%f",&nd[a].con);
}

for (j=1;j<=nds;j++) {nd[j].pos=0;}
out=fopen("pnodes.dat","w");intsect=fopen("intsect.in","w");
shortp=fopen("shortp.in","w");fgrid = fopen("fgrid.dat","r");
fscanf(fgrid,"%f",&incr);fprintf(out,"%10.4f\n",incr);

nodes=0;
for (k=1;k<=nds;k++) {
 if ((nd[k].b==fws)&&(nd[k].p==1)) {
  nodes++;nd[k].pos=nodes;
  fprintf(out,"%d %10.4f  %10.4f
%10.4f\n",nodes,nd[k].x,nd[k].y,nd[k].elev);

ndbnd[nodes].x=nd[k].x;ndbnd[nodes].y=nd[k].y;ndbnd[nodes].z=nd[k].elev;
 }
}
for (k=1;k<=nds;k++) {
 if (nd[k].b==tws) {
  nodes++;nd[k].pos=nodes;
  fprintf(out,"%d %10.4f  %10.4f
%10.4f\n",nodes,nd[k].x,nd[k].y,nd[k].elev);

ndbnd[nodes].x=nd[k].x;ndbnd[nodes].y=nd[k].y;ndbnd[nodes].z=nd[k].elev;
 }
}
k=nodes;
fprintf(shortp,"%d %f\n",k,diam);
for (k=1;k<=1000000;k++) {
 fscanf(fgrid,"%f",&x);check=x;
 if (check==-9999)
   break;
 fscanf(fgrid,"%f",&y);fscanf(fgrid,"%f",&z);
 nodes++;
```

```
 fprintf(out,"%d %10.4f %10.4f %10.4f\n",nodes,x,y,z);
 ndbnd[nodes].x=x;ndbnd[nodes].y=y;ndbnd[nodes].z=z;
}
inp = fopen("bound.dat","r");
for (i=1;i<=100000;i++) {
 fscanf(inp,"%d",&id);check=id;
 if (id==0)
   break;
 fscanf(inp,"%f",&x);fscanf(inp,"%f",&y);fscanf(inp,"%f",&z);
 nodes++;fprintf(out,"%d %10.4f %10.4f %10.4f\n",nodes,x,y,z);
 ndbnd[nodes].x=x;ndbnd[nodes].y=y;ndbnd[nodes].z=z;
}

fprintf(out,"%d %d %d\n",0,0,0);fclose(out);fclose(inp);

fclose(intsect);fclose(shortp);
system("intgen.exe");system("shortp.exe");
shrt=fopen("shortp.out","r");
fscanf(shrt,"%f",&length);fscanf(shrt,"%d",&nodesalong);
/**** manipulation of shortp output *****/
for (i=1;i<=nodesalong;i++) {
 fscanf(shrt,"%d",&val);
 for (j=1;j<=nodes;j++) {
  if (nd[j].pos==val)
   path[i]=j;
  if (nd[j].pos==1)
   path[1]=j;
 }
 x1[i]=ndbnd[val].x;y1[i]=ndbnd[val].y; z1[i]=ndbnd[val].z;
}
inp1=fopen("fpath.out","w");
fprintf(inp1,"%d %10.2f\n",nodesalong,length);
for (i=1;i<=nodesalong;i++) {
 fprintf(inp1,"%d %10.2f %10.2f %10.2f\n",path[i],x1[i],y1[i],z1[i]);
}
/**** end of manipulation  *******/
fclose(shrt);
}
```

GENSHTP4

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
struct card1 {int
num,used,fn,tn,d,del,wshed,number,wshed2,trav,outnd;float
deficit,outel,len,el1,el2,x1,y1,x2,y2;} link[3500],plink[3500];
struct card2 {int id,fnnum,used,out,linked,wshed;float con,x,y,elev;}
node[1500];
struct card3 {int outnd,num,used; float outel,xc,yc;
 struct card4 {int id; float x,y,elev;} node[1000];
} ws[200];
/*GLOBAL VARIABLES FOR MAX DEPTH CHECKING */
struct card5 {int num,fn,tn;float slope,len,x1,x2,y1,y2,el1,el2,d1,d2;}
lnk[1000];
int d[1000][1000],lnks,nds;
/*END GLOBAL VARIABLES FOR MAX DEPTH CHECKING */

int
numnodes,start,terminus,wshed,wl,wl1,wl2,links,nodes,count=0,as[1000],as
1[1000],asnum;
int outnd,prev,e,q,nd[1500],unit;
float range,outel,dist,alb,mincov,minslp=.0024;
main()
{
float maxdepth(int ax,int bx,int cx,int dx);
void up1(int z, int legs);void up(int z, int legs);
void sort(void);void nodelink(void);
void flip (int i);void deldbl(int wshed, int c);void centroid(int a);
void awshed2 (int wshed);int merger (int a, float b);
FILE *inp1,*inp2,*inp3,*inp4,*out,*out1,*out2,*conn,*nodes2,*units;
int i,j,k,l,r,s,p,qq,outlet,num,outnum,merge1,merge2;
int missing;
float high,low,check,dm,d1,d2,deficit;
inp1=fopen("nodes.dat","r");inp2=fopen("genshtp.dat","r");
inp3=fopen("genshtp.aux","r");
inp4=fopen("gsdpm3.aux","r");
out=fopen("gennum.dat","w");
out1=fopen("genshtp.arc","w");out2=fopen("genshtp1.arc","w");
units=fopen("unit.dat","r");
fscanf(inp3,"%d",&outnd);fscanf(inp3,"%f",&alb);
fscanf(inp4,"%f",&mincov);
fscanf(units,"%d",&unit);
alb = alb + 1;
check=1;i=0;
while (check!=0) {i++;
fscanf(inp1,"%d",&node[i].id);fscanf(inp1,"%f",&node[i].x);
fscanf(inp1,"%f",&node[i].y);fscanf(inp1,"%f",&node[i].elev);fscanf(inp1
,"%f",&node[i].con);
node[i].out=0;node[i].used=0;check=node[i].x;
node[i].linked=0;
}
nodes=i-1;i=0;check=1;
while (check!=0) {i++;
fscanf(inp2,"%d",&plink[i].fn);fscanf(inp2,"%d",&plink[i].tn);
fscanf(inp2,"%f",&plink[i].len);
check=plink[i].len;plink[i].used=0;link[i].wshed=0;link[i].wshed2=0;
link[i].del=0;link[i].deficit=0;link[i].trav=0,link[i].d=0;
}
```

```
links=i-1;wshed=0;
sort(); nodelink();
for (i=1;i<=nodes;i++) {if(node[i].id==outnd){node[i].out=1;outnum=i;}}

for (i=1;i<=nodes;i++) {
       /* Look for node with least elev.*/
 dm=10000000;num=0;
 for (j=1;j<=nodes;j++) {
  if ((node[j].elev<dm)&&(node[j].linked==0)) {
   dm=node[j].elev;outlet=node[j].id;num=j;}
 }

     /* End Look for node with least elev. */
 if (num>0) {wshed++;
  node[num].out=1;
   if (i==1)
    outlet=outnd;
   for (j=1;j<=links;j++)  {
    for (l=1;l<=nodes;l++)
     nd[l]=0;
    up1 (outlet,links);
   }
   wl=0;
   ws[wshed].num=0;
   for (j=1;j<=links;j++) {
    if (link[j].wshed==wshed) {ws[wshed].num++;
     link[j].outnd=outlet;
     link[j].outel=node[num].elev;
     ws[wshed].outnd=outlet;
     ws[wshed].outel=node[num].elev;
     ws[wshed].node[ws[wshed].num].id=link[j].fn;
     ws[wshed].node[ws[wshed].num].x=link[j].x1;
     ws[wshed].node[ws[wshed].num].y=link[j].y1;
     ws[wshed].node[ws[wshed].num].elev=link[j].el1;
     if (link[j].number>wl) {
      wl=link[j].number;qq=j;
     }
    }
   }
   if(wl>0) {ws[wshed].num++;
    ws[wshed].node[ws[wshed].num].id=link[qq].tn;
    ws[wshed].node[ws[wshed].num].x=link[qq].x2;
    ws[wshed].node[ws[wshed].num].y=link[qq].y2;
    ws[wshed].node[ws[wshed].num].elev=link[qq].el2;
   }
  /* centroid(wshed) */; ws[wshed].used=0;

 }

    /* Check for nodes with links attached */
 if(num>0) {
  for (s=1;s<=links;s++)
   if (link[s].wshed>0)
    for (r=1;r<=nodes;r++)
     if(node[r].linked==0)
      if ((link[s].fn==node[r].id)||(link[s].tn==node[r].id)) {
       node[r].linked=1;
      } /* End of Check for nodes with links attached */
 }
}
```

Appendix D:

```
/* Print */
qq=0;
for (i=1;i<=wshed;i++) {
qq++;
for (j=1;j<=links;j++)
 if ((link[j].del==0)&&(link[j].wshed==i))
 {
  fprintf(out1,"%d \n",qq);
  fprintf(out1,"%f %f\n",link[j].x1,link[j].y1);
  fprintf(out1,"%f %f\n",link[j].x2,link[j].y2);
  fprintf(out1,"END \n");
 }
}
fprintf(out1,"END \n");fclose(out1);
/* End Print */

deldbl(wshed,0);
awshed2(wshed);

/* new linking */
hey:;
i=1;
while (i<wshed) {
for (qq=1;qq<=wshed;qq++) {asnum=0;merge2=0;merge1=0;
 dm=1000000;wl=0;
 if(ws[i].used!=1)
 for (k=i+1;k<=wshed;k++) {asnum=0;
  if(ws[k].used!=1) {
    for (j=1;j<=links;j++) {
     if (link[j].wshed==i) {
      deficit=link[j].deficit;outel=link[j].outel;outnd=link[j].outnd;
     }
     if ((link[j].wshed2==i)&&(link[j].wshed==k)&&(link[j].del==0)) {
      asnum++;as[asnum]=j;
     }
    }
  }
  wl=0;
  if (asnum>0) {dm=999999;
   for (j=1;j<=asnum;j++) {
    link[as[j]].d=1;
    if (link[as[j]].el1-link[as[j]].outel<dm) {
     dm=link[as[j]].el1-link[as[j]].outel;wl=as[j];d2=dm;
    }
   }
   link[wl].d=0;
    /*Check if the allowable is exceeded by deficit */
   dm=maxdepth(i,k,link[wl].outnd,outnd);
    d1=deficit;
   if (dm<=alb) {
    link[wl].del=0;missing=merger(wl,d1);merge1=1;
    deldbl(wshed,0);
    awshed2(wshed);
    for (l=1;l<=links;l++) {
     link[l].trav=1;
     if ((link[l].wshed==i)&&(link[l].del==0))
      link[l].trav=0;
    }
```

```
    e=0;
    for (l=1;l<=links;l++)
     up(outnd,links);
    goto hey;
   }
  }
 }
}
i++;
}
deldbl(wshed,1);
/* end new linking */

/*Renumber Link wsheds */
s=0;i=0;
for (i=1;i<=wshed;i++)
 if (ws[i].used==0) {s++;
  for (j=1;j<=links;j++)
   if (link[j].wshed==i)
    link[j].wshed=s;
 }
/* End Renumber Link wsheds */

nodes2=fopen("nodes2.dat","w");
fprintf(nodes2,"%d\n",nodes);
for (i=1;i<=nodes;i++) {
 node[i].out=0;
 for (j=1;j<=links;j++) {
  if (link[j].del==0) {
   if ((node[i].id==link[j].fn)||(node[i].id==link[j].tn)) {
    node[i].wshed=link[j].wshed;
    if ((node[i].id==link[j].tn)&&(node[i].id==link[j].outnd))
     node[i].out=1;
    break;
   }
  }
 }
 fprintf(nodes2,"%d %d %d %f %f %f
%f\n",node[i].id,node[i].wshed,node[i].out,node[i].elev,node[i].x,node[i
].y,node[i].con);
}
fclose(nodes2);

printf("%d s \n",s);

for (j=1;j<=s;j++) {
 for (l=1;l<=nodes;l++) {
  if((node[l].wshed==j)&&(node[l].out==1)) {
   outnd=node[l].id;break;
  }
 }
 for (l=1;l<=links;l++) {
  link[l].trav=1;
  if ((link[l].wshed==j)&&(link[l].del==0))
   link[l].trav=0;
 }
printf("%d outnd \n",outnd);

 e=0;
 for (l=1;l<=links;l++)
```

```
  up(outnd,links);
}


s=0;
 /* Print data */
conn=fopen("conn.out","w");
for (i=1;i<=links;i++)
 if (link[i].del==0)
  s++;
fprintf(conn,"%d\n",s);
for (i=1;i<=links;i++)
 if (link[i].del==0)
  fprintf(conn,"%d %d %d %d
%d\n",i,link[i].fn,link[i].tn,link[i].number,link[i].wshed);
fclose(conn);

s=0;
for (j=1;j<=wshed;j++) {wl=0;
 for (i=1;i<=links;i++) {
  if ((link[i].wshed==j)&&(link[i].del==0)) {wl++;outnd=link[i].outnd;}
 }
 if (wl>0) {
  fprintf(out,"9999\n");fprintf(out,"%d\n%d\n",wl,outnd);
  for (i=1;i<=links;i++)
   if ((link[i].wshed==j)&&(link[i].del==0)) {s++;
    fprintf(out,"%d %d %d %14.4f
%14.4f\n",link[i].fn,link[i].tn,link[i].wshed,link[i].el1,link[i].el2);
    fprintf(out2,"%d \n",s);
    fprintf(out2,"%f %f\n",link[i].x1,link[i].y1);
    fprintf(out2,"%f %f\n",link[i].x2,link[i].y2);
    fprintf(out2,"END\n");
   }
 }
}
fprintf(out2,"END\n");fprintf(out,"-9999\n");
fclose(out2);fclose(out);fclose(inp1);fclose(inp2);
}


void flip(int i)
{
int fn,tn;float el1,el2,x1,x2,y1,y2;
tn=link[i].fn;fn=link[i].tn;link[i].fn=fn;link[i].tn=tn;
x1=link[i].x2;
y1=link[i].y2;
x2=link[i].x1;
y2=link[i].y1;
link[i].x1=x1;
link[i].y1=y1;
link[i].x2=x2;
link[i].y2=y2;
el2=link[i].el1;el1=link[i].el2;link[i].el1=el1;link[i].el2=el2;
}

void deldbl(int wshed,int c) {
/*delete all links except 1 the are in the same
  subwatershed and drain a common node */

int wc,wd,i,j,k,l,cl[100],c1;float dm;
```

```
if (c==0)
for (i=1;i<=wshed;i++) {
 for (k=1;k<=nodes;k++) {
  c1=0;
  for (j=1;j<=links;j++) {
   if ((node[k].id==link[j].fn)&&(link[j].wshed==i)&&(link[j].del==0)) {
    c1++;cl[c1]=j;
   }
  }
  if(c1>1) {dm=999999;
   for (j=1;j<=c1;j++) {
    link[cl[j]].del=1;
     if (link[cl[j]].len<dm) {
      dm=link[cl[j]].len;wc=cl[j];
     }
   }
   link[wc].del=0;
  }
 }
}
if (c==1) {
 for (k=1;k<=nodes;k++) {
  c1=0;
  for (j=1;j<=links;j++) {
   if ((node[k].id==link[j].fn)&&(link[j].del==0)) {
    c1++;cl[c1]=j;
   }
  }
  if(c1>1) {wd=999999;
   for (j=1;j<=c1;j++) {
    link[cl[j]].del=1;
    if (link[cl[j]].wshed<wd) {
     wd=link[cl[j]].wshed;wc=cl[j];
    }
   }
   link[wc].del=0;
  }
 }
}

/* End delete all links except 1 the are in the same
  subwatershed and drain a common node */
}

void awshed2 (int wshed) {
int c[100],tp,i,j,k,n1,n2;float dm;
/* Assign wshed2 */
for (i=1;i<=nodes;i++) {
 node[i].fnnum=0;
 for (j=1;j<=links;j++)
  if ((link[j].fn==node[i].id)&&(link[j].del==0))
   node[i].fnnum++;
}
for (i=1;i<=nodes;i++) {
 if (node[i].fnnum>1) {k=0;
  for (j=1;j<=links;j++) {
   if ((link[j].fn==node[i].id)&&(link[j].del==0)) {k++;
    c[k]=link[j].wshed;
   }
  }
```

```
   n1=100000;
   for (j=1;j<=k;j++)
    if (c[j]<n1)
     n1=c[j];
   n2=100000;
   for (j=1;j<=k;j++)
    if ((c[j]<n2)&&(c[j]!=n1))
     n2=c[j];
   for (j=1;j<=links;j++) {
    if ((link[j].fn==node[i].id)&&(link[j].del==0)) {
     if (link[j].wshed!=n1)
      link[j].wshed2=n1;
     if (link[j].wshed==n1)
      link[j].wshed2=n2;
     if (link[j].wshed==1)
      link[j].wshed2=0;
    }
   }
  }
}

 /* End Assign wshed2 */
}

int merger (int a, float b) {
void up (int z, int legs);
int r,i,j,k,l,missing;
ws[link[a].wshed].used=1;
missing=link[a].wshed;
/*********************/
for (l=1;l<=links;l++) {
 link[l].trav=1;
 if ((link[l].wshed==link[a].wshed)&&(link[l].del==0))
  link[l].trav=0;
}
k=link[a].fn;
for (l=1;l<=links;l++)
 up(k,links);
/*****************/
for (r=1;r<=links;r++) {
 if ((link[r].wshed==link[a].wshed)&&(r!=a)) {
  link[r].wshed=link[a].wshed2;link[r].outnd=outnd;
  link[r].outel=outel;
  link[r].deficit=b; link[r].wshed2=0;
 }
 if(link[r].wshed2==link[a].wshed)
  link[r].wshed2=link[a].wshed2;
}
link[a].wshed=link[a].wshed2;
link[a].outnd=outnd;link[a].outel=outel;
link[a].deficit=b;link[a].wshed2=0;
return missing;
}


void up(int z, int legs){
void flip(int i);int f=0,i=1;
while (i<=legs) {
 if ((link[i].fn==z)&&(link[i].trav==0)&&(link[i].tn!=prev)) {
   flip(i); break;} i++;
```

```
}i=1;
while (i<=legs) {
  if ((link[i].tn==z)&&(link[i].trav==0)) {
  prev=z;z=link[i].fn;f=1;break;
 }i++;
}i=1;
if (f == 0)
while (i<=legs) {
  if ((link[i].fn == z)&&(link[i].tn==prev))   {
   link[i].trav = 1; e++;link[i].number=e;break;
 } i++;
}
if (f == 1) {up(z,legs);}
}


void up1 (int z, int legs){
void flip(int i);int f=0,i=1;nd[z]++;
while (i<=legs) {
 if ((link[i].fn==z)&&(link[i].trav==0)&&(link[i].tn!=prev))
  if (link[i].el2-link[i].el1>=0)
    {
  flip(i); break;} i++;
}i=1;
while (i<=legs) {
 if (nd[link[i].fn]<2)
 if ((link[i].tn==z)&&(link[i].trav==0))
    if (link[i].el1-link[i].el2>=0) {
  prev=z;z=link[i].fn;f=1;break;
 }i++;
}i=1;
if (f == 0)
while (i<=legs) {
 if ((link[i].fn == z)&&(link[i].tn==prev)){
  link[i].trav = 1; e++;link[i].number=e;
  link[i].wshed=wshed;break;
 } i++;
}
if (f == 1) {up1(z,legs);}
}

void sort() {
int i,j,p; float dm;
/*sorting links so the the shortest occurs first */
for (i=1;i<=links;i++) {
 dm=10000000;
 for (j=1;j<=links;j++)
  if (plink[j].used==0)
   if (plink[j].len<dm) {
     dm=plink[j].len;p=j;
   }
link[i].fn=plink[p].fn;link[i].tn=plink[p].tn;
link[i].len=plink[p].len;plink[p].used=1;
}
/* End sort */
}

void nodelink() {
/* Assigns attributes to links based on their fn and tn */
int i,j;
```

```
for (i=1;i<=links;i++) {
 for (j=1;j<=nodes;j++) {
  if (link[i].fn == node[j].id) {
    link[i].el1 = node[j].elev;link[i].x1 = node[j].x;
    link[i].y1 = node[j].y;
  }
  if (link[i].tn == node[j].id) {
    link[i].el2 = node[j].elev;link[i].x2 = node[j].x;
    link[i].y2 = node[j].y;
  }
 }
}
/* End nodelink */
}

void centroid(int a) {
int i;
float numb;
ws[a].xc=0;
ws[a].yc=0; numb=0;
for (i=1;i<=ws[a].num;i++) {
ws[a].xc=ws[a].xc+ws[a].node[i].x;
ws[a].yc=ws[a].yc+ws[a].node[i].y;
numb=numb+1;
}
ws[a].xc = ws[a].xc/numb;ws[a].yc = ws[a].yc/numb;
/* printf(" %d %f %f\n",a, ws[a].xc, ws[a].yc); */

}

float maxdepth(int ax, int bx, int start, int end)
{
int chk(int p,int num);
int a,b,i,j,k,l;
float dx,dy,dz,slope,depth=0;
for (i=1;i<=nodes;i++)
 for (j=1;j<=nodes;j++)
  d[i][j]=999;
for (i=1;i<=links;i++) {
 if ((link[i].wshed==ax)&&(link[i].del==0)) {
  d[link[i].fn][link[i].tn]=0;
  d[link[i].tn][link[i].fn]=0;
 }
 if((link[i].wshed==bx)&&(link[i].del==0)&&(link[i].d==0)) {
  d[link[i].fn][link[i].tn]=0;
  d[link[i].tn][link[i].fn]=0;
 }
}
/*start=16;end=3;*/
a=start;
while(1) {
 b=a;b=chk(b,nodes);
 if(b>0) {
  d[a][b]++;a=b;
 }
 if(b==end)
  break;
}
k=0;
for(i=1;i<=nodes;i++)
```

```
 for(j=1;j<=nodes;j++)
  if((d[i][j]==1)&&(d[j][i]==0)) {k++;lnk[k].fn=i;lnk[k].tn=j;}

b=0;i=0;a=start;
while (b==0) {
 for(j=1;j<=k;j++)
  if(lnk[j].fn==a) {i++;
   lnk[j].num=i;a=lnk[j].tn;
   if (lnk[j].tn==end)
    b=end;
   break;
  }
}
lnks=k;
for (i=1;i<=lnks;i++){
for (j=1;j<=nodes;j++) {
 if (lnk[i].fn == j) {
   lnk[i].el1 = node[j].elev;lnk[i].x1 = node[j].x;
   lnk[i].y1 = node[j].y;
 }
 if (lnk[i].tn == j) {
   lnk[i].el2 = node[j].elev;lnk[i].x2 = node[j].x;
   lnk[i].y2 = node[j].y;
 }
}
if (unit==1) {
lnk[i].el1=lnk[i].el1/.3048;
lnk[i].el2=lnk[i].el2/.3048;
lnk[i].x1=lnk[i].x1/.3048;
lnk[i].x2=lnk[i].x2/.3048;
lnk[i].y1=lnk[i].y1/.3048;
lnk[i].y2=lnk[i].y2/.3048;
}
}
for(i=1;i<=k;i++)
 for(j=1;j<=k;j++)
  if(lnk[j].num==i) {
   if (i==1)
    lnk[j].d1=mincov ;
   dx=lnk[j].x1-lnk[j].x2;dy=lnk[j].y1-lnk[j].y2;
   dz=lnk[j].el1-lnk[j].el2;
   lnk[j].len=sqrt(pow(dx,2)+pow(dy,2));
   slope = minslp;
   lnk[j].d2=lnk[j].d1-(lnk[j].el1-lnk[j].el2)+lnk[j].len*slope;
   if (lnk[j].d2<mincov )
    lnk[j].d2=mincov ;
   if (i<k)
    for (l=1;l<=k;l++)
     if(lnk[l].num==i+1)
      lnk[l].d1=lnk[j].d2;
   if (lnk[i].d2>depth)
    depth=lnk[i].d2;
  }
return depth;
}

int chk(int p,int num)
{int j,k=0;
for (j=1;j<=num;j++) {
 if((d[p][j]==0)&&(d[j][p]!=1)) {k=j;break;}
```

```
}
if (k==0)
 for (j=1;j<=num;j++) {if(d[p][j]==0) {k=j;break;}}
return k;
}
```

INTGEN

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
struct card1 {
int fn,tn,id,del,id1,id2;float x,y,z,x1,y1,z1,x2,y2,z2;
} plot[8000],link[2],nodes[8000];
main()
{
int intsect(float x1,float y1,float x2,float y2,float x3,float y3,float
x4,float y4);
FILE *inp2,*inp,*out1,*grid;
int i,j,k,m,a,b,gc,junk=0;
float check,dx,dy,dm,y1,y2,y3,y4,x1,x2,x3,x4,incr;
inp = fopen("pnodes.dat","r");out1=fopen("intsect.out","w");
inp2=fopen("poly.dat","r");
grid=fopen("persgrid.out","w");
gc=0;
b=0;check=1;
printf("begin intgen\n");
while (check!=0) {b++;
fscanf(inp2,"%f",&plot[b].x1);fscanf(inp2,"%f",&plot[b].y1);
fscanf(inp2,"%f",&plot[b].x2);fscanf(inp2,"%f",&plot[b].y2);
check=plot[b].x1;
}
fscanf(inp,"%f",&incr);
check=1;i=0;
while (check != 0) {
i++;fscanf(inp,"%d",&nodes[i].id);check=nodes[i].id;
fscanf(inp,"%f",&nodes[i].x);fscanf(inp,"%f",&nodes[i].y);
fscanf(inp,"%f",&nodes[i].z);
}
m=i-1;fprintf(out1,"%d \n",m);a=1;
for (j=1;j<m;j++) {
printf("%d %d \n",m, j);
for (k=j+1;k<=m;k++) {
 link[a].id1=nodes[j].id;
 link[a].id2=nodes[k].id;
 link[a].x1=nodes[j].x;
 link[a].y1=nodes[j].y;
 link[a].z1=nodes[j].z;
 link[a].del=0;
 link[a].x2=nodes[k].x;
 link[a].y2=nodes[k].y;
 link[a].z2=nodes[k].z;

 dx=link[a].x2-link[a].x1;dy=link[a].y2-link[a].y1;
 dm=sqrt(dx*dx+dy*dy);
 dm=0.9800*dm;
 if (dm>sqrt(pow(incr,2)+pow(incr,2)))
  link[a].del=1;

 for (i=1;i<b;i++) {
  if (link[a].del==0) {
   x1=plot[i].x1;x2=plot[i].x2;y1=plot[i].y1;y2=plot[i].y2;
   x3=link[a].x1;x4=link[a].x2;y3=link[a].y1;y4=link[a].y2;
   link[a].del=intsect(x1,y1,x2,y2,x3,y3,x4,y4);
  }
 }
```

```
 if (link[a].del==0) {
  gc++;
  fprintf(grid,"%d \n%8.4f %8.4f \n%8.4f %8.4f \n%s
\n",gc,link[a].x1,link[a].y1,link[a].x2,link[a].y2,"END");
  fprintf(out1,"%d %d %8.4f %8.4f %8.4f %8.4f %8.4f
%8.4f\n",link[a].id1,link[a].id2,link[a].x1,link[a].y1,link[a].z1,link[a
].x2,link[a].y2,link[a].z2);
 }
}
}
fprintf(grid,"%s\n","END");
fprintf(out1,"%d %d %d\n",0,0,0);
fclose(inp);fclose(out1);
printf("end intgen\n");

}
int intsect(float x1,float y1,float x2,float y2,float x3,float y3,float
x4,float y4)
{int box1(float x1,float x2,float x3,float x4);
int box2(float y1,float y2,float y3,float y4);
float same (float x1,float y1,float x2,float y2,float x3,float y3,float
x4,float y4);
int p=0;float p1,p2;
if ((x1==x2)&&(x2==x3)&&(x3==x4)) {p=box1(x1,x2,x3,x4);goto hey;}
if ((y1==y2)&&(y2==y3)&&(y3==y4)) {p=box2(y1,y2,y3,y4);goto hey;}
p1=same(x1,y1,x2,y2,x3,y3,x4,y4);p2=same(x3,y3,x4,y4,x1,y1,x2,y2);
if ((p1<=0)&&(p2<=0)){p=1;}
hey:;return p;
}

float same(float x1,float y1,float x2,float y2,float x3,float y3,float
x4,float y4)
{int p=0;float dx,dy,dx1,dy1,dx2,dy2,temp,a1,a2;
dx=x4-x3;dy=y4-y3;dx1=x1-x3;dy1=y1-y3;dx2=x2-x4;dy2=y2-y4;
temp=(dx*dy1-dy*dx1)*(dx*dy2-dy*dx2);
return temp;
}

int box1(float x1,float x2,float x3,float x4)
{ int p=0;float temp;
if (x2<x1) { temp=x2;x2=x1;x1=temp;}
if (x4<x3) { temp=x4;x4=x3;x3=temp;}
if (((x1>=x3)&&(x2<=x3))||((x1>=x4)&&(x2<=x4)))
 p=1;
return p;
}
int box2(float y1,float y2,float y3,float y4)
{ int p=0;float temp;
if (y2<y1) { temp=y2;y2=y1;y1=temp;}
if (y4<y3) { temp=y4;y4=y3;y3=temp;}
if (((y1>=y3)&&(y2<=y3))||((y1>=y4)&&(y2<=y4)))
 p=1;
return p;
}
```

SHORTP

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
struct card {
int fn,tn,id,del,id1,id2;
float x1,y1,x2,y2;
} plot[1600],link[2];
struct card2 {
int num[1600],nodes;
} shortest;
int endnode,units;
float dist,diam;
main()
{
void shortpath(int nodes);
FILE *inp1,*inp2,*out,*out1,*unit;
int i,j,k,a,b,nodes;
inp1=fopen("shortp.in","r");
unit=fopen("unit.dat","r");fscanf(unit,"%d",&units);fclose(unit);
fscanf(inp1,"%d",&nodes);
fscanf(inp1,"%f",&diam);
dist=1000000;
shortpath(nodes);
out=fopen("shortp.out","w");
fprintf(out,"%10.4f  %d\n",dist,shortest.nodes);
for (i=1;i<=shortest.nodes;i++) {
 fprintf(out,"%d \n",shortest.num[i]);
}
fclose(out);
fclose(inp1);
}

void shortpath(int nodes)
{
/***************Function Shortest Path*****************/
FILE *inp;
int m,x,t6,n,i,j,t0,t5,t8,p,s,t,q[1400], o[1400], r[1400];
float x1,x2,y1,y2,z1,z2,len,t1,t3,t2,t7, l[1400],d[1400][1400];
float dx,dy,dz,eqlen,e,f,g,v;
e=.00085;g=32.2;v=4;
f=pow((2*log10(diam/e)+1.14),-2);
inp = fopen("intsect.out","r");
t5 = 1400;t5=t5-1;
for (i=1;i<=t5;i++) {
 for (j=1;j<=t5;j++) {
  d[i][j] = 999999;
  d[j][i] = 999999;
 }o[i]=0;d[i][i]=0;
}
/* shortest path algorithm (Dijkstra) */
fscanf(inp,"%d",&n);
while (i!=0) {
fscanf(inp,"%d",&i);fscanf(inp,"%d",&j);
fscanf(inp,"%f",&x1);fscanf(inp,"%f",&y1);fscanf(inp,"%f",&z1);
fscanf(inp,"%f",&x2);fscanf(inp,"%f",&y2);fscanf(inp,"%f",&z2);

dx=x2-x1;dy=y2-y1;dz=z2-z1;
if (units==1) {
```

```
dz=dz/.3048;dy=dy/.3048;dx=dx/.3048;
}
len=sqrt(pow(dx,2)+pow(dy,2)+pow(dz,2));
d[i][j]=len;
d[j][i]=len;
eqlen=2*diam*g*sqrt(pow(dz,2))/(f*pow(v,2));
if (dz>0)
 d[i][j]=d[i][j]+eqlen;
if (dz<0)
 d[j][i]=d[j][i]+eqlen;
}
fclose(inp);

for (m=2;m<=nodes;m++) {
 s=1;t=m;
 for (i=1;i<=n;i++) {          /* Step 0 */
  l[i]=999999;q[i]=0;
 }q[s]=1;l[s]=0;p=s;

 while (q[t]==0) {             /* Step 1 */
  for (i=1;i<=n;i++)
   if (q[i]!=1) {
    t1=l[p]+d[p][i];
    if (l[i]>=t1)
     l[i]=t1;
   }p=0;t2=999999;
  for (i=1;i<=n;i++)
   if ((q[i]!=1)&&(t2>=l[i])) {
    p=i;t2=l[i];
   }q[p]=1;
 }
 for (j=1;j<=n;j++)    /*  Step 3 */
  if (q[j]!=0)
   for (i=1;i<=n;i++)
    if (i!=j) {
     t3 = l[i]+ d[i][j];
     if (l[j]>=t3) {
      r[j]=i;
      if (d[i][j]<999998)
       break;
      goto end;
     }
    }t8=t5;o[t8]=t;i=r[t];          /*step 4 */
 for (t1=1;t1<=1000000;t1++) {
  t8--;o[t8]=i;
  if (i==s)
   break;
  i=r[i];
 }
 /*
 printf("\n path found is");
 */
 for (i=t8;i<=t5;i++)
 /*  printf("- %d",o[i]);

 printf("\n this has length %f",l[t]);
 */
 if (l[t]<dist) {
  x=0;
  for (i=t8;i<=t5;i++) {
```

```
    x++;
    shortest.num[x]=o[i];shortest.nodes=x;
   }
   dist=l[t];endnode=o[t5];
  }
 end:;
}
/**********************End of Shortest Path **************/
}
```

COMPILE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{

 system("cc tree.c -o tree.exe -lm");
 system("cc intgen1.c -o intgen1.exe -lm");
 system("cc intgen.c -o intgen.exe -lm");
 system("cc trace.c -o trace.exe -lm");
 system("cc join.c -o join.exe -lm");
 system("cc num.c -o num.exe -lm");
 system("cc conn.c -o conn.exe -lm");
 system("cc draw.c -o draw.exe -lm");
 system("cc forcemod.c -o forcemod.exe -lm");
 system("cc forcem.c -o forcem.exe -lm");
 system("cc fpath.c -o fpath.exe -lm");
 system ("cc design.c -o design.exe -lm");
 system("cc wetwell.c -o wetwell.exe -lm");
 system("cc costs.c -o costs.exe -lm");
 system("f77 pumpsel.for -o pumpsel.exe");
 system("f77 gsdpm3.for -o gsdpm3.exe");

}
```

BOUND

```
&s time1 [date -time]
/*&goto skip
/******calculating generalization
mape %.topo%
&s xmin [extract 1 [show mape]]
&s ymin [extract 2 [show mape]]
&s xmax [extract 3 [show mape]]
&s ymax [extract 4 [show mape]]

&s dx %xmax% - %xmin%
&s dy %ymax% - %ymin%
&s d2 %dx%
&if %dy% < %dx% &then
 &s d2 %dy%

&s offset %.n% * 2
&s offset %d2% / %offset%

&s xmin %xmin% + %offset%
&s ymin %ymin% + %offset%
&s xmax %xmax% - %offset%
&s ymax %ymax% - %offset%

&s dx %xmax% - %xmin%
&s dy %ymax% - %ymin%
&s d2 %dy%
&if %dy% < %dx% &then
 &s d2 %dx%

&s incr %d2% / %.n%

/***end calculating gen
/*&label skip

/*&messages &off &all
/*&s .bldg bldg2
/*&s .buff 1
/*&s .tinfile tin
graphics on
&s .cov %.bldg%

q no
build %.cov% arc
build %.cov% node
arcedit
&if [exists %.cov%1 -cover] &then
  kill %.cov%1 yes
&if [exists %.cov%2 -cover] &then
  kill %.cov%2 yes
&if [exists %.cov%3 -cover] &then
  kill %.cov%3 yes
&if [exists temp2 -cover] &then
  kill temp2 yes
ec %.cov%
ef arc
sel all
coo keyb
&sv numarc [show number select]
```

```
&do a = 1 &to %numarc%
 &s bigx 0
 &s vert%a% [show arc %a% npnts]
 &s verta [value vert%a%]
 &do vtnum = 1 &to %verta%
   &s x [extract 1 [show arc %a% vertex %vtnum%]]
   &s y [extract 2 [show arc %a% vertex %vtnum%]]
   &if %x% gt %bigx% &then
    &do
    &s bigx %x%
    &s buff2 %.buff% * 1.2
    &s y%a% %y%
    &s x%a% %x% + %buff2%
    &s platy%a% %y%
    &s platx%a% %x% + %.buff%
    &end
  &type %x% %y%
 &end
 &s num%a% %a%
 &if %a% = 1 &then
  &do
   sel $recno = 1
   copy parallel
   2, %platx1%, %platy1%
   put %.cov%3
   copy parallel
   2, %x1%, %y1%
   put %.cov%1
  &end
 &if %a% > 1 &then
  &do
   sel $recno = %a%
   copy parallel
   2, [value platx%a%], [value platy%a%]
   put %.cov%3
   y
   copy parallel
   2, [value x%a%], [value y%a%]
   put %.cov%1
   y
  &end
&end
save temp2
save
kill temp2 yes
/*DENSIFY ARCS
  &sys arc densifyarc %.cov%1 # %incr% arc
  &sys arc build %.cov%1 arc
/* END DENSIFY
ec %.cov%1
ef arc
sel all
&sv numarc [show number select]
&s clstat [close -all]
&s bound [open bound.dat bnd -w]
&s open1 [open poly.dat opens1 -w]
&s k 0
&s p 0
&do a = 1 &to %numarc%
 &s vert [show arc %a%   npnts]
```

```
 &s vert %vert% - 1
 &do vtnum = 1 &to %vert%
   &s k %k% + 1
   &s x%k% [extract 1 [show arc %a%   vertex %vtnum%]]
   &s y%k% [extract 2 [show arc %a%   vertex %vtnum%]]
   &s wrt [write %bound% [quote %k% [value x%k%] [value y%k%]]]
 &end
&end
&s wrt [write %bound% [quote END]]
ec %.cov%3
ef arc
sel all
&sv numarc [show number select]
&s a 0
&do d = 1 &to %numarc%
 &s vert [show arc %d% npnts]
/* &s vert %vert% - 1
/*write all sides of polygon except 1
 &s vert %vert% - 2
 &do b = 1 &to %vert%
  &s a %a% + 1
  &s x1%a% [extract 1 [show arc %d% vertex %b%]]
  &s y1%a% [extract 2 [show arc %d% vertex %b%]]
  &s c %b% + 1
  &s x2%a% [extract 1 [show arc %d% vertex %c%]]
  &s y2%a% [extract 2 [show arc %d% vertex %c%]]
  &s wstat [write %open1% [quote [value x1%a%] [value y1%a%] [value
x2%a%] [value y2%a%]]]
 &end
&end
&s wstat [write %open1% [quote 0 0 0 0]]
q
generate temp2
input bound.dat
points
q
build temp2 point
arcedit
/* COULD BE DONE TOGETHER WITH EARLIER STEPS
ec %.cov%3
ef arc
sel all
&sv numarc [show number select]
&if %numarc% > 0 &then
&do
 &sv a := 1
 &do &while %a% le %numarc%
  &s arc [show select %a%]
  &s f 1
  &s vert%a% [show arc %arc% npnts]
  &s vtnum := 1
  &do &while %vtnum% le [value vert%a%]
    &s poly%a%node%f%x [extract 1 [show arc %arc% vertex %vtnum%]]
    &s poly%a%node%f%y [extract 2 [show arc %arc% vertex %vtnum%]]
    &s f %f% + 1
    &s vtnum %vtnum% + 1
  &end
  &s a %a% + 1
 &end
 coo keyboard
```

```
 ec temp2
 ef point
 &s k 1
 &do a = 1 &to %numarc%
  asel polygon within
  &do f = 1 &to [value vert%a%]
   &sv x [value poly%a%node%f%x]
   &sv y [value poly%a%node%f%y]
   %x%,%y%
  &end
~
 &end
 &s numsel [show number select]
 &if %numsel% gt 0 &then
  delete
 save
&end

kill %.cov%1 yes
&sys arc tinspot %.tinfile% temp2
ec temp2
ef point
sel all
&sv numpt [show number select]
&s clstat [close -all]
&s bound [open bound.dat bnd -w]
&s k 0
&do a = 1 &to %numpt%
 &s x [extract 1 [show label %a% coordinate]]
 &s y [extract 2 [show label %a% coordinate]]
 &s z [show label %a% item spot]
 &s wrt [write %bound% [quote %a% %x% %y% %z%]]
&end
&s wrt [write %bound% [quote 0 0 0 ]]
kill temp2 yes
&s clst [close -all]
&messages &on
graphics on
&s time2 [date -time]
&type [quote %time1%]
&type [quote %time2%]

&return
```

FGRID

```
&messages &off &all
&s n %.n%
ec %.cover%
mape %.topo%
&s xmin [extract 1 [show mape]]
&s ymin [extract 2 [show mape]]
&s xmax [extract 3 [show mape]]
&s ymax [extract 4 [show mape]]
&s clstat [close -all]
&s open1 [open fgrid.dat opens1 -w]
&s wstat [write %open1% [quote %n%]]
&s wstat [write %open1% [quote %xmin% %ymin% %xmax% %ymax%]]
&s cls [close -all]
&sys fgrid.exe
q
&if [exists fgrid -cover] &then
  kill fgrid all
generate fgrid
input fgrid.out
points
q
build fgrid point
arcedit

ec %.bldg%3
ef arc
sel all
&sv numarc [show number select]
&if %numarc% > 0 &then
&do
 &sv a := 1
 &do &while %a% le %numarc%
  &s arc [show select %a%]
  &s f 1
  &s vert%a% [show arc %arc% npnts]
  &s vtnum := 1
  &do &while %vtnum% le [value vert%a%]
    &s poly%a%node%f%x [extract 1 [show arc %arc% vertex %vtnum%]]
    &s poly%a%node%f%y [extract 2 [show arc %arc% vertex %vtnum%]]
    &s f %f% + 1
    &s vtnum %vtnum% + 1
  &end
  &s a %a% + 1
 &end
 coo keyboard
 ec fgrid
 ef point
 &s a 1
 &s k 1
 &do &while %a% le %numarc%
  asel polygon within
  &s f 1
  &do &while %f% le [value vert%a%]
   &sv x [value poly%a%node%f%x]
   &sv y [value poly%a%node%f%y]
   %x%,%y%
   &s f %f% + 1
  &end
```

```
~
 &s numsel [show number select]
 &s a %a% + 1
 &end
 &s numsel [show number select]
 &if %numsel% gt 0 &then
 delete
 save
&end

&sys arc tinspot %.tinfile% fgrid
&s cls [close -all]
&s open1 [open fgrid.ou opens1 -r]
&s incr [read %open1% rd]
ec fgrid
ef point
sel spot = -9999
&if [show number select] gt 0 &then
 delete
save
&s cls [close -all]
&s open1 [open fgrid.dat opens1 -w]
&s wstat [write %open1% [unquote %incr%]]
ef point
sel all
&s numpt [show number select]
&do a = 1 &to %numpt%
 &s x [extract 1 [show label %a% coordinate]]
 &s y [extract 2 [show label %a% coordinate]]
 &s z [show label %a% item spot]
 &s wstat [write %open1% [quote %x% %y% %z%]]
&end
&s stop -9999
&s wstat [write %open1% [quote %stop%]]
&s cls [close -all]
&messages &on
&return
```

GRADISP

```
&s .backcov1 %.force%
&s .backcov2 %.outcov%
&s .backcov3 %.bldg%
&s .backcov4 %.lots%
&s .backcov5 %.street%
&s .backcov6 %.topo%
&s .backcov7 %.cover%
&do a = 2 &to 6
 &s .bec%a% .FALSE.
&end

&s be2 arc
&s be3 node
&s be4 arc arrows
&s be5 point
&s be6 all
&menu editc.menu
&menu backc.menu
&if %.maincov% = 0 &then
 &s mc %.outcov%
&if %.maincov% = 1 &then
 &s mc %.force%
&if %.maincov% = 2 &then
 &s mc %.bldg%
&if %.maincov% = 3 &then
 &s mc %.lots%
&if %.maincov% = 4 &then
 &s mc %.street%
&if %.maincov% = 5 &then
 &s mc %.topo%

de all off
ec %mc%
&if %.dem1% = .TRUE. &then
 de arc
&if %.dem2% = .TRUE. &then
 de node
&if %.dem3% = .TRUE. &then
 de arc arrows
&if %.dem4% = .TRUE. &then
 de arc label
&if %.dem5% = .TRUE. &then
 de point
&if %.dem6% = .TRUE. &then
 de all


&if %.bcov1% = .TRUE. &then
&do
 bc %.backcov1% %.becol1%
 &do a = 2 &to 6
  &if [value .bea%a%] = .TRUE. &then
   be %.backcov1% [value be%a%]
 &end
&end

&if %.bcov2% = .TRUE. &then
&do
```

```
 bc %.backcov2% %.becol2%
 &do a = 2 &to 6
  &if [value .beb%a%] = .TRUE. &then
   be %.backcov2% [value be%a%]
 &end
&end

&if %.bcov3% = .TRUE. &then
&do
 bc %.backcov3% %.becol3%
 &do a = 2 &to 6
  &if [value .bec%a%] = .TRUE. &then
   be %.backcov3% [value be%a%]
 &end
&end

&if %.bcov4% = .TRUE. &then
&do
 bc %.backcov4% %.becol4%
 &do a = 2 &to 6
  &if [value .bed%a%] = .TRUE. &then
   be %.backcov4% [value be%a%]
 &end
&end

&if %.bcov5% = .TRUE. &then
&do
 bc %.backcov5% %.becol5%
 &do a = 2 &to 6
  &if [value .bee%a%] = .TRUE. &then
   be %.backcov5% [value be%a%]
 &end
&end

&if %.bcov6% = .TRUE. &then
&do
 bc %.backcov6% %.becol6%
 &do a = 2 &to 6
  &if [value .bef%a%] = .TRUE. &then
   be %.backcov6% [value be%a%]
 &end
&end

&if %.bcov7% = .TRUE. &then
&do
 bc %.backcov7% %.becol7%
 &do a = 2 &to 6
  &if [value .beg%a%] = .TRUE. &then
   be %.backcov7% [value be%a%]
 &end
&end

draw
```

PLAT

```
&args .coverp
&sys arc build %.coverp% line
ec %.coverp%
&s clstat [close -all]
&s open1 [open poly.dat opens1 -w]
ef arc
sel all
&sv numarc [show number select]
&s a 0
&do d = 1 &to %numarc%
 &s vert [show arc %d% npnts]
 &if %vert% <> 5 &then
  &do
   /* &s vert %vert% - 1
   /*write all sides of polygon except 1
   &s vert %vert% - 2
   &do b = 1 &to %vert%
    &s a %a% + 1
    &s x1%a% [extract 1 [show arc %d% vertex %b%]]
    &s y1%a% [extract 2 [show arc %d% vertex %b%]]
    &s c %b% + 1
    &s x2%a% [extract 1 [show arc %d% vertex %c%]]
    &s y2%a% [extract 2 [show arc %d% vertex %c%]]
   &end
  &end
 &if %vert% = 5 &then
  &do
   &s a %a% + 1
   &s x1%a% [extract 1 [show arc %d% vertex 1]]
   &s y1%a% [extract 2 [show arc %d% vertex 1]]
   &s x2%a% [extract 1 [show arc %d% vertex 3]]
   &s y2%a% [extract 2 [show arc %d% vertex 3]]
   &s a %a% + 1
   &s x1%a% [extract 1 [show arc %d% vertex 2]]
   &s y1%a% [extract 2 [show arc %d% vertex 2]]
   &s x2%a% [extract 1 [show arc %d% vertex 4]]
   &s y2%a% [extract 2 [show arc %d% vertex 4]]
  &end
&end
&do d = 1 &to %a%
 &s wstat [write %open1% [quote [value x1%d%] [value y1%d%] [value
x2%d%] [value y2%d%]]]
&end
 &s wstat [write %open1% [quote 0 0 0 0]]
&s cls [close -all]
```

POLY

```
&s .coverp pstreet
&sys arc build %.coverp% line
ec %.coverp%
&s clstat [close -all]
&s open1 [open poly.dat opens1 -w]
ef arc
sel all
&sv numarc [show number select]
&s a 0
&do d = 1 &to %numarc%
 &s vert [show arc %d% npnts]
 &s vert %vert% - 1
 &do b = 1 &to %vert%
  &s a %a% + 1
  &s x1%a% [extract 1 [show arc %d% vertex %b%]]
  &s y1%a% [extract 2 [show arc %d% vertex %b%]]
  &s c %b% + 1
  &s x2%a% [extract 1 [show arc %d% vertex %c%]]
  &s y2%a% [extract 2 [show arc %d% vertex %c%]]
 &end
&end
&s wstat [write %open1% %a%]
&do d = 1 &to %a%
 &s wstat [write %open1% [quote [value x1%d%] [value y1%d%] [value
x2%d%] [value y2%d%]]]
/* &type  [value x1%d%] [value y1%d%] [value x2%d%] [value y2%d%]
&end
```

PROFILE

```
 clear
/*mape wsan1
pageunits inches
pagesize 11 8.5
units map
surface tin tin linear
/*textsize 20 pt
/*axis vertical nodraw
/*axis horizontal nodraw
surfaceprofile '2 1 10.73 8' wsan1 prwsan1 5
surfaceprofile '2 1 10.73 8' force prforc1 5
/*axis vertical nodraw
/*axis horizontal nodraw
/*axisruler Elevation
/*axis horizontal
/*axistext X_Distan
/*axis vertical
axishatch 10 2070
```

PROFILE2

```
edit prforc1 info
sel all
&s num [show number select]
&type %num%
&s cls [close -all]
&s opn1 [open prforc1.dat opens1 -w]
&s opn2 [open prwsan1.dat opens2 -w]
&s wstat [write %opn1% %num%]
&do a = 1 &to %num%
 &s x [show info %a% item px]
 &s y [show info %a% item py]
 &s z [show info %a% item spot]
 &s d [show info %a% item distance]
 &s wstat [write %opn1% [quote %x%  %y%  %z% %d%]]
&end
edit prwsan1 info
sel all
&s num [show number select]
&type %num%
&s wstat [write %opn2% %num%]
&do a = 1 &to %num%
 &s x [show info %a% item px]
 &s y [show info %a% item py]
 &s z [show info %a% item spot]
 &s d [show info %a% item distance]
 &s wstat [write %opn2% [quote %x%  %y%  %z%  %d%]]
&end
&s cls [close -all]
```

```
START

reset no
&terminal 9999
&menu button.menu &form

&if [exists tin -tin] &then
 &s .tinfile tin

&if %.tin% = 0 &then
 &do
  &if [exists tin -tin] &then
   &sys arc kill tin all
  &sys arc arctin %.topo% tin line contour
 &end

&if %.cunit% = 1 &then
 &s .buff %.buff% * .3048

&if %.cbuf% = 0 &then
 &r bound

&if %.cgrid% = 0 &then
 &r fgrid

&if %.gsdpm3% = 0 &then
 &do
  &menu gsdpm3.menu
  &s cls [close -all]
  &s opn [open gsdpm3.aux opns -w]
  &s wrt [write %opn% %.mincov%]
  &s wrt [write %opn% %.ppc%]
  &s wrt [write %opn% %.fpp%]
  &s wrt [write %opn% %.meth%]
  &s wrt [write %opn% %.gpf%]
  &s wrt [write %opn% %.manns%]
  &s wrt [write %opn% %.stand%]
  &s wrt [write %opn% %.meansidx%]
  &s wrt [write %opn% %.year%]
  &s wrt [write %opn% %.infilt%]
  &s wrt [write %opn% %.matrl%]
  &s wrt [write %opn% %.cityidx%]
  &s wrt [write %opn% %.ctime%]
  &s wrt [write %opn% %.len%]
  &s cls [close -all]
 &end
&if [exists %.outcov% -cover] &then
  kill %.outcov% yes
&if [exists force -cover] &then
  kill %.force% yes
 &s cls [close -all]
&s op [open unit.dat opens1 -w]
&s wstat [write %op% [quote  %.cunit%]]
&s cls [close -all]
&r testgen1
```

TEMP

```
q
&if [exists temp -cover] &then
kill temp all
generate temp
input genshtp.arc
lines
q
&if [exists test2 -cover] &then
kill test2 all
generate test2
input genshtp1.arc
lines
q
build test2 line
build temp line
display 9999
arcedit
ec test2
de arc
de arc arrows
bc wman 3

be point
draw
&return
```

TESTGEN1

```
&s bldg %.bldg%
&s maxlen 150
&if %.cunit% = 2 &then
&s maxlen %maxlen% * 3.37
&s tin %.tinfile%
&s point %.cover%
/*&messages &off &all
graphics off
&if [exists man -cover] &then
  kill man yes
&if [exists mantin -tin] &then
  &sys arc kill mantin
&if [exists mantina -cover] &then
  kill mantina yes
&sys arc copy %point% man
&sys arc tinspot %tin% man contour
&sys arc arctin man mantin point contour
&sys arc tinarc mantin mantina line percent
&sys arc build mantina node
&sys arc build mantina arc

/* BEGIN DELETION OF LINKS THAT ARE PASSING THROUGH BUILDING LOTS
&s cover mantina
graphics off
&if [exists %cover%1 -cover] &then
  kill %cover%1 yes
&sys arc copy %cover% %cover%1
ec %bldg%
ef arc
sel all
&sv numarc [show number select]
&if %numarc% > 0 &then
&do
 &sv a := 1
 &do &while %a% le %numarc%
  &s arc [show select %a%]
  &s f 1
  &s vert%a% [show arc %arc% npnts]
  &s vtnum := 1
  &do &while %vtnum% le [value vert%a%]
    &s poly%a%node%f%x [extract 1 [show arc %arc% vertex %vtnum%]]
    &s poly%a%node%f%y [extract 2 [show arc %arc% vertex %vtnum%]]
    &s f %f% + 1
    &s vtnum %vtnum% + 1
  &end
  &s a %a% + 1
 &end
 coo keyboard
 intersectarcs off
 ec %cover%1
 ef arc
 &s a 1
 &s k 1
 &do &while %a% le %numarc%
  asel polygon passthru
  &s f 1
```

```
   &do &while %f% le [value vert%a%]
    &sv x [value poly%a%node%f%x]
    &sv y [value poly%a%node%f%y]
    %x%,%y%
    &s f %f% + 1
   &end
~
 &s numsel [show number select]
 &s a %a% + 1
 &end
 asel length > %maxlen%
 &s numsel [show number select]
 &if %numsel% gt 0 &then
 delete
 save
&end
graphics on
q
&if [exists testposlnk -cover] &then
  kill testposlnk all
rename %cover%1 testposlnk
arcedit
&r testgen2a
q
renode testposlnk
arcedit
graphics on
&messages &on
&r testgen2
```

TESTGEN2

```
&s clstat [close -all]
&s open1 [open genshtp.dat opens1 -w]
&s end 0 0 0 0
&sys arc pointnode man testposlnk 0.1
ec testposlnk
ef arc
sel all

&sv numarc [show number select]
&sv a := 1
&do &while %a% le %numarc%
 &s arc%a%fn [show arc %a% fnode#]
 &s arc%a%tn [show arc %a% tnode#]
 &s arc%a%len [show arc %a% item length]
 &s wstat [write %open1% [quote  [value arc%a%fn] [value arc%a%tn]
[value arc%a%len]]]
 &s a %a% + 1
&end
 &s wstat [write %open1% [quote %end%]]
&s clstat [close -all]

&s open2 [open nodes.dat opens1 -w]
&s opn [open genshtp.aux opens1 -w]
ef node
sel all
&sv numnode [show number select]
&sv a := 1
&do &while %a% le %numnode%
 &s node%a%id [show node %a% item testposlnk#]
 &s node%a%x [extract 1 [show node %a% coordinate]]
 &s node%a%y [extract 2 [show node %a% coordinate]]
 &s node%a%elev [show node %a% item contour]
 &s node%a%con [show node %a% item con]
 &s node%a%out [show node %a% item outlet]
 &if [show node %a% item outlet] = 1 &then
  &s wstat [write %opn% [quote  [value node%a%id] %.alb%]]
 &s wstat [write %open2% [quote  [value node%a%id] [value node%a%x]
[value node%a%y] [value node%a%elev] [value node%a%con]]]
 &s a %a% + 1
&end
&s wstat [write %open2% [quote %end%]]
&s clstat [close -all]
&sys genshtp.exe
&sys forcearr.exe
&sys forcemod.exe
&r temp
&sys arc kill man all
&sys arc kill mantin all
&sys arc kill mantina all
&return
```

TESTGEN2A

```
/* for all the links put x1,y1,x2,y2
&if [exists junkme2 -cover] &then
 kill junkme2 yes
q
copy testposlnk junkme2
build junkme2 node
build junkme2 arc
build junkme2 poly
createlabels junkme2
build junkme2 poly
arcedit
ec junkme2
ef node
sel all
&s numnd [show number select]
&do a = 1 &to %numnd%
 &s x%a% [extract 1 [show node %a% coordinate]]
 &s y%a% [extract 2 [show node %a% coordinate]]
&end
ef arc
sel all
&s numarc [show number select]
&do a = 1 &to %numarc%
 &s fn [show arc %a% item fnode#]
 &s tn [show arc %a% item tnode#]
 &s link%a%fn %fn%
 &s link%a%tn %tn%
 &s link%a%x1 [value x%fn%]
 &s link%a%y1 [value y%fn%]
 &s link%a%x2 [value x%tn%]
 &s link%a%y2 [value y%tn%]
 &s link%a%len [show arc %a% item length]
&end


ef label
sel all
&s numlab [show number select]
&s m 0
ef arc
&do a = 1 &to %numlab%
 sel rpoly# = %a% or lpoly# = %a%
 &if [show number select] = 3 &then
  &do
   &s r [show select 1]
   &s s [show select 2]
   &s t [show select 3]
   &s p 0
/*find max link */
   &if [value link%r%len] > %p% &then
    &do
     &s p [value link%r%len]
     &s max %r%
    &end
   &if [value link%s%len] > %p% &then
    &do
     &s p [value link%s%len]
     &s max %s%
```

```
     &end
    &if [value link%t%len] > %p% &then
     &do
      &s p [value link%t%len]
      &s max %t%
     &end
/* end find max link */
  &do c = 1 &to 2
   &s e %c% + 1
   &do d = %e% &to 3
    &s j [show select %c%]
    &s k [show select %d%]

    &if %j% <> %max% &then
     &if     %k% <> %max% &then
     &do
      &if [value link%j%fn] = [value link%k%fn] &then
       &do
        &s l1 [radang   [invangle [value link%j%x1] [value link%j%y1]
[value link%j%x2] [value link%j%y2]]]
        &s l2 [radang   [invangle [value link%k%x1] [value link%k%y1]
[value link%k%x2] [value link%k%y2]]]
       &end
      &if [value link%j%tn] = [value link%k%fn] &then
       &do
        &s l1 [radang   [invangle [value link%j%x2] [value link%j%y2]
[value link%j%x1] [value link%j%y1]]]
        &s l2 [radang   [invangle [value link%k%x1] [value link%k%y1]
[value link%k%x2] [value link%k%y2]]]
       &end
      &if [value link%j%fn] = [value link%k%tn] &then
       &do
        &s l1 [radang   [invangle [value link%j%x1] [value link%j%y1]
[value link%j%x2] [value link%j%y2]]]
        &s l2 [radang   [invangle [value link%k%x2] [value link%k%y2]
[value link%k%x1] [value link%k%y1]]]
       &end
      &if [value link%j%tn] = [value link%k%tn] &then
       &do
        &s l1 [radang   [invangle [value link%j%x2] [value link%j%y2]
[value link%j%x1] [value link%j%y1]]]
        &s l2 [radang   [invangle [value link%k%x2] [value link%k%y2]
[value link%k%x1] [value link%k%y1]]]
       &end
      &s m1 [max %l1% %l2%]
      &s m2 [min %l1% %l2%]
      &s l3 %m1% - %m2%
      &if %l3% > 180 &then
       &s l3 360 - %l3%
      &if %l3% > 155 &then
       &do
        &s m %m% + 1
        &s del%m% %max%
       &end
     &end
   &end
  &end
  &end
&end
&do a = 1 &to %m%
```

```
sel $recno = [value del%a%]
delete
&end
save
kill testposlnk yes
&sys arc copy junkme2 testposlnk
kill junkme2 yes

&return
```

TEXTDISP.AML

```
&s cls [close -all]
/*&messages &off
/*&s opn [open forcemod.out opns -r]
&s opn [open subwat1.out  opns -r]
&s rds 0
&s a 0
&do &while %rds% = 0
 &s a %a% + 1
 &s line%a% [substr [read %opn% rds] 1 128]
 &if [type [value line%a%]] = -1 &then
  &s line%a% [quote [value line%a%]]
&end
reset no
&if [exists junk -info] &then
 kill junk info yes
create junk info
itm
128
128
c
~
edit junk info
&do b = 1 &to %a%
 add
 moveitem [value line%b%] itm
&end
save
```

WPLOT

```
mape wtopo10
clear
pagesize 8.5 10.9
pageunits inches
units page
textfont times
textquality tightkern
lineset template.lin
mapposition cen cen
mapshift none
maplimits  0.5 0.5 8 10
mape wtopo2 bldg
lineset plotter.lin
linesymbol 1
arcs bldg
linesymbol 5
arcs bldg1
textsize 5 pt
linesymbol 1
arctext wtopo10 cont line # blank
linesymbol 25
arcs wsan1
linesymbol 41
arcs force50
linesymbol 45
arcs force
linesymbol 1
units map
spot 10927284 3626071 20 outline
spot 10926578 3624270 20 outline
units page
textsize 10 pt
textfont 'times bold'
move 3, 2.37
text 'Destination'
move 4.47, 6.22
text 'Source'
move 6, 3.3
textsize 16 pt
text 'LEGEND'
textfont times
textsize 14 pt
keyarea 5.3 3.1 8 1
keybox .5 .3
keyseparation .15 .2
keyline wplot.key
```

WYATTDXF

```
dxfarc wyatt.dxf wright
p-rightofway
END
Y
dxfarc wyatt.dxf wsan
p-sansewer
END
Y
dxfarc wyatt.dxf wbldg
x-bldg
END
Y
dxfarc wyatt.dxf wbound
x-boundary
END
Y
dxfarc wyatt.dxf wtopo2
x-topo2
END
Y
/*dxfarc wyatt.dxf wtopo10
/*x-topo10
/*END
/*Y
```

```
MENUS

BACKC

7 Out coverage menu

                GRAPHIC DISPLAY MENU

    Enter Required information:
       BACKGROUND LAYERS:

 %p1 Force Main Layer
    Default Backenvironment: %a2 ARC %a3 NODE %a4 ARROWS %a5 POINT %a6
ALL
    Color:   %p2


 %q1 Sewer Network Layer
    Default Backenvironment: %b2 ARC %b3 NODE %b4 ARROWS %b5 POINT %b6
ALL
    Color:   %q2


 %r1 Buildings Layer
    Default Backenvironment: %c2 ARC %c3 NODE %c6 ALL
    Color:   %r2


 %s1 Property Lots Layer
    Default Backenvironment: %d2 ARC %d3 NODE %d6 ALL

   Color:   %s2


 %t1 Streets Layer
    Default Backenvironment: %e2 ARC %e3 NODE %e6 ALL
    Color:   %t2


 %u1 Contours Layer
    Default Backenvironment: %f2 ARC %f6 ALL
    Color:   %u2


 %v1 Manhole Layer
    Default Backenvironment: %g5 POINT
    Color:   %v2


                            %ok         %cancel


%p1 checkbox .bcov1 init .FALSE.
%q1 checkbox .bcov2 init .FALSE.
%r1 checkbox .bcov3 init .FALSE.
%s1 checkbox .bcov4 init .FALSE.
%t1 checkbox .bcov5 init .FALSE.
%u1 checkbox .bcov6 init .FALSE.
%v1 checkbox .bcov7 init .FALSE.
```

```
/* Colors
%p2 choice .becol1 pairs initial 2 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%q2 choice .becol2 pairs initial 3 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%r2 choice .becol3 pairs initial 4 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%s2 choice .becol4 pairs initial 5 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%t2 choice .becol5 pairs initial 6 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%u2 choice .becol6 pairs initial 7 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7
%v2 choice .becol7 pairs initial 7 RED 2 GREEN 3 BLUE 4 CYAN 5 MAGENTA 6
YELLOW 7

%a2 checkbox .bea2 init .TRUE.
%a3 checkbox .bea3 init .TRUE.
%a4 checkbox .bea4 init .TRUE.
%a5 checkbox .bea5 init .FALSE.
%a6 checkbox .bea6 init .FALSE.

%b2 checkbox .beb2 init .TRUE.
%b3 checkbox .beb3 init .FALSE.
%b4 checkbox .beb4 init .TRUE.
%b5 checkbox .beb5 init .FALSE.
%b6 checkbox .beb6 init .FALSE.

%c2 checkbox .bec2 init .TRUE.
%c3 checkbox .bec3 init .FALSE.
%c6 checkbox .bec6 init .FALSE.

%d2 checkbox .bed2 init .TRUE.
%d3 checkbox .bed3 init .FALSE.
%d6 checkbox .bed6 init .FALSE.


%e2 checkbox .bee2 init .TRUE.
%e3 checkbox .bee3 init .FALSE.
%e6 checkbox .bee6 init .FALSE.

%f2 checkbox .bef2 init .TRUE.
%f6 checkbox .bef6 init .FALSE.

%g5 checkbox .beg5 init .TRUE.


%ok button OK &return
%cancel button cancel 'CANCEL' &return
```

```
BUTTON

7 Out coverage menu
            MAIN MENU

              Enter Required information:

 Building Coverage/Layer       %1

 Manhole Coverage/Layer        %2

 Topographic Coverage/Layer    %3

 Streets Coverage/Layer        %st

 Zoning Coverage/Layer         %zo

 Property Lots Coverage/Layer %lot

 Output Coverage/Layer         %4

 Force Main Coverage/Layer     %5


 Number of grid <10X10 - 30X30>
 %6

 Coverage Units In:
 %7

 Max. Allowable Depth Beyond Which a Pump Must be Used(Feet):
 %8

 Buffer Width (Feet):
 %9

 Do You Want To Create a New Buffer File?:                    %10

 Do You Want To Create a New Grid File?:                      %11

 Do You Want To Change GSDPM3 or Wet Well Design Parameters?:%12

 Do You Want To Create a New TIN Layer?:                      %13


                              %ok          %cancel

%1 input .bldg 40 init bldg2  cover * -all -other 'Select a Building
coverage'
%2 input .cover 40 init wman  cover * -all -other 'Select a Manhole
coverage'
%3 input .topo 40 init wtopo2  cover *  -all -other 'Select a Topo
coverage'
%st input .street 40 init wpave  cover *  -all -other 'Select a Street
coverage'
%zo input .zone 40 init testzone  cover *  -all -other 'Select a Zoning
coverage'
%lot input .lots 40 init wplot  cover *  -all -other 'Select a Property
Lots coverage'
```

```
%4 input .outcov 40 init temp  character
%5 input .force 40 init force  character
%6 slider .n 30 initial 50 integer 10  100
%7 choice .cunit pairs init 2 Meters 1 Feet 2
%8 slider .alb 50 initial 15 integer 0  40
%9 slider .buff 50 initial 10 integer 0  200
%10 choice .cbuf pairs init 1 NO 1 YES 0
%11 choice .cgrid pairs init 1 NO 1 YES 0
%12 choice .gsdpm3 pairs init 1 NO 1 YES 0
%13 choice .tin pairs init 1 NO 1 YES 0
%ok button OK &return
%cancel button cancel 'CANCEL' &return
```

EDITC

```
7 Out coverage menu

               GRAPHIC DISPLAY MENU

    Enter Required information:


 Main Coverage/Layer       %1

 Draw Environment:
 %2 ARC %3 NODE %4 ARROWS %5 LABEL %6 POINT %7 ALL



                            %ok          %cancel


%1 choice .maincov pairs Network 0 ForceMain 1 Buildings 2 Property_Lots
3 Streets 4 Contour 5
%2 checkbox .dem1 init .TRUE.
%3 checkbox .dem2 init .TRUE.
%4 checkbox .dem3 init .FALSE.
%5 checkbox .dem4 init .FALSE.
%6 checkbox .dem5 init .FALSE.
%7 checkbox .dem6 init .FALSE.


%ok button OK &return
%cancel button cancel 'CANCEL' &return
```

GSDPM3

```
7 Out coverage menu

            Enter Required information:


            GSDPM3 Design Parameters:


 Minimum Depth of Cover              %1
 Number of Connections Per Person    %2
 Flow Per Person (gal/day)           %3
 Matching of Inverts                 %4
 Global Peak Factor                  %5
 Manning's n                         %6
 Standard Used                       %7
 Means Cost Index                    %8
 Year of Design                      %9
 Type of Pipe Material:              %10
 Max. Allowable infiltration         %11
 City Cost Index                     %12

            WET WELL Parameters:


 Cycle Time (Minutes)
  %13

 Wet Well Width (Feet):
  %14

          %ok       %cancel

%1 input .mincov 5 init 3 real
%2 input .ppc 5 init 2.7 real
%3 input .fpp 5 init 100 real
%4 choice .meth pairs init 1 'Match Crowns' 1 '0.8 Depth' 0
%5 input .gpf 5 init 2.5 real
%6 input .manns 5 init 0.013 real
%7 choice .stand pairs init 0 'Ten States' 0 'West Virginia' 1
%8 input .meansidx 5 init 221.6 real
%9 input .year 4 init 1995 real
%10 choice .matrl pairs init 1 PVC 1 CLAY 2 CONCRETE 3
%11 input .infilt 5 init 500 real
%12 input .cityidx 5 init 78 real
%13 slider .len 50 init 10 integer 0 20
%14 slider .ctime 50 init 20 integer 0 30
%ok button OK &return
%cancel button cancel 'CANCEL' &return
```

PUMPSEL.FOR

```
      REAL Q,G,H,KQMAX(100),
     *KHMAX(100),NSMAX(100),N(100),NS(100),PKQ(100),PKH(100),PNS(100),
     *PTH(100),PA(100),PB(100),PC(100),PEA(100),PEB(100),PEC(100),
     *PSPEED(100),PNSCALC(100),NS1,NSPRIME,KQ,KH,DIAM,EF,POWER,DELTA,
     *SPEED,DIAMTEMP,EFTEMP,POWERTEMP,SPEEDTEMP,DIFF,IMPDIAM(100),HA,
     *QA,QB,HAV,DM,PW,EFC,SP

      INTEGER Y,Z,I,J,K,PTYPE(100),IVAL(100),IV,W,NUMPUMPS,PUMPNUMTEMP,
     *PNUM
      CHARACTER*12 OUTPUT
      OPEN (UNIT = 5, FILE = 'pumpsel.in',status = 'old')
      READ(5,3) OUTPUT
      READ(5,*) Q
      READ(5,*) H
C     READ HEAD IN FT CONVERT TO METERS
C     READ FLOW IN GPM, CONVERT TO CU M/S
      H=H*.3048
      Q=Q*0.0631/1000
    3 format(A12)
    4 format(1X,A12)
      OPEN(UNIT=6, FILE=OUTPUT,status='UNKNOWN')
      DIFF=10
      DIFF=10
      PTH(1)=0
      G = 9.81
      IMPDIAM(1)=.314
      CALL ROOT(NUMPUMPS)

      OPEN (UNIT = 10, FILE = 'coeffs.dat',status = 'UNKNOWN')
      I = 0
      DO WHILE ((PA(I).NE.99).AND.(PB(I).NE.99).AND.(PC(I).NE.99))
      I = I + 1
      READ (10,*) PA(I), PB(I), PC(I)
      IF ((PA(I).EQ.99).AND.(PB(I).EQ.99).AND.(PC(I).EQ.99)) THEN
        GOTO 5
      END IF
      READ (10,*) PEA(I), PEB(I), PEC(I)
C      REM CALCULATE NS, KQ, KH VALUES AT MAXIMUM EFFICIENCY
      KQMAX(I) = -PEB(I) / (2 * PEC(I))
      KHMAX(I) = PA(I) + PB(I) * KQMAX(I) + PC(I) * KQMAX(I) ** 2
      NSMAX(I) = KQMAX(I) ** .5 / KHMAX(I) ** .75
C      REM PRINT USING "##.###  ##.###  ##.###"; KQMAX(I); KHMAX(I);
NSMAX(I)
      PNS(I) = NSMAX(I)
      PKQ(I) = KQMAX(I)
      PKH(I) = KHMAX(I)
   5  END DO
      CLOSE (10)


C      REM NOMINAL PUMP SPEED DATA
      N(1) = 1750
      N(2) = 1650
      N(3) = 1550
      N(4) = 1450
      N(5) = 1350
      N(6) = 1250
      N(7) = 1150
```

```
      N(8) = 1050
      N(9) = 950
      N(10) = 850
      N(11) = 750
      N(12) = 650
      N(13) = 550
      W = NUMPUMPS

      DELTA = .001
   10 K = 0
      DO 30 I = 1,13
        NS(I) = N(I) * Q ** .5 / (G * H) ** .75
        NS(I) = NS(I) / 60
        DO 20 J = 1,W
          IF (ABS(PNS(J) - NS(I)).LT. DELTA) THEN
            K = K + 1
            PTYPE(K) = J
            IVAL(K) = I
            PSPEED(K) = N(I)
            PNSCALC(K) = NS(I)
          END IF

   20   CONTINUE
   30 CONTINUE
      IF (K .LT. 34 .AND. DELTA .LT. .1) THEN
        DELTA = DELTA + DELTA / 10.0
        GOTO 10
      END IF

C      REM FOR EACH PUMP, USE BISECTION METHOD TO MATCH THE NS WITH THE
OPERATING
C      REM NS
      QB = 0
   35 DO 65 M = 1,K
        IV=IVAL(M)
        NS1= NS(IVAL(M))
        KQ = PKQ(PTYPE(M)) / 2
   40   KH = PA(PTYPE(M)) + KQ * PB(PTYPE(M)) + KQ ** 2 * PC(PTYPE(M))
        IF (KH.LT.0) GOTO 65

        NSPRIME = SQRT(KQ)/(KH ** .75)
        IF ((NS1- NSPRIME).GT.0.0001) THEN
C          REM 1.5 IS FOR BISECTION METHOD
          KQ = KQ * 1.5
          GOTO 40
        END IF

        IF ((NS1- NSPRIME).LT.-0.0001) THEN
C          REM .5 IS FOR BISECTION METHOD
          KQ = KQ * .5
          GOTO 40
        END IF
        Z = PTYPE(M)
        EF = PEA(Z) + KQ * PEB(Z) + (KQ ** 2) * PEC(Z)
        KH = PA(Z) + KQ * PB(Z) + (KQ ** 2) * PC(Z)
        Y=IVAL(M)
        DIAM = (Q * 60 / (N(Y) * KQ)) ** (1.0 / 3.0)
        POWER = G * Q * H / (EF / 100)
        SPEED = (NS1* (G * H) ** .75 / Q ** .5) * 60
        QA = KQ*(SPEED/60)*IMPDIAM(1)**3
```

```
        PUMPNUMTEMP = 0
        IF (ABS(DIAM - IMPDIAM(1)) .LT. 0.1*IMPDIAM(1)) THEN
          SPEEDTEMP= SPEED
          DIAMTEMP= DIAM
          POWERTEMP = POWER
          EFTEMP = EF
          HA = KH*(SPEED/60)**2*IMPDIAM(1)**2/G
          IF (HA .GT. H) THEN
            PUMPNUMTEMP = PTYPE(IV)
          END IF
        END IF
   58   FORMAT (1X, A, I5, A, F6.0 ,A)
   59   FORMAT (1X, A, F7.4, A)
   60   FORMAT (1X, A, F6.3, A, F6.3, A)
   61  IF ((PUMPNUMTEMP .GT. 0) .and. ((ABS(QA-Q)) .LT. 0.4*Q)) THEN
        IF (QA .GT. QB) THEN
         PNUM = PUMPNUMTEMP
         SP = SPEEDTEMP
         DM = DIAMTEMP
         PW = POWERTEMP
         EFC = EFTEMP
         QB = QA
         HAV = HA
        END IF
        PUMPNUMTEMP = 0
       END IF
   65 CONTINUE
C       WRITE(*,58) 'PUMP #',PNUM,'  SPEED = ', SP,' rev/min'
C       WRITE(*,60) 'DIAMETER = ',DM,' m   or ',DM*3.281, ' ft'
C       WRITE(*,60) 'POWER =     ',PW,' kW  or ',PW*1.34,' hp'
C       WRITE(*,59) 'EFFC. =     ',EFC, ' %'
C       WRITE(*,59) 'FLOW =    ',QB, ' m^3/s '
C       WRITE(*,60) 'Required head = ',H,' m or ',H*3.281,' ft'
C       WRITE(*,60) 'Available head = ',HAV,' m or ',HAV*3.281,' ft'


      WRITE(6,68) PNUM
      WRITE(6,68) SP
      WRITE(6,67) DM*3.281
      WRITE(6,67) PW*1.34
      WRITE(6,67) EFC
      WRITE(6,67) H*3.281
      WRITE(6,67) HAV*3.281
   67 FORMAT(' ',F16.5)
   68 FORMAT(' ',F6.0)
   69 END


C     ********************************************************
C     * SUBROUTINE ROOT                                      *
C     ********************************************************
      SUBROUTINE ROOT(NUMPUMPS)

      REAL F3(16),F2(16),SUM3(16),EF(16),SUM2(16),SUM1(16,16),
     *E(3,3),G(3,3),F(3,3),M(16,3),T(3,16),
     *KQ(16), KH(16),DET,DETINV,P1,Q1,R1
      INTEGER L,W,I,J,K
      OPEN (UNIT = 8, FILE = 'pumps.dat',status='old')
      OPEN (UNIT = 9, FILE = 'coeffs.dat',status='UNKNOWN')
      DO WHILE (KQ(W).NE.9999)
```

```
      W = 0
   70 W = W + 1
      READ (8,*) KQ(W), KH(W), EF(W)
C       REM CHECKS FOR END OF INDIVIDUAL PUMP DATA
      IF ((KQ(W) .EQ. 0) .AND. (KH(W) .EQ. 0)) THEN
        GOTO 220
      END IF

C       REM CHECK END OF FILE
      IF (KQ(W) .NE. 9999) THEN
        GOTO 70
      END IF
      W = W - 1
      NUMPUMPS = W
C       REM FIRST ROUND(L=1) CALCULATES PUMP CHARACTERISTIC FITTING EQN
C       REM SECOND ROUND (L=2) CALC. EFF. FITTING EQN
      DO 200 L = 1,2
        IF (L .EQ. 2) THEN
          DO 75 I = 1,W
            KH(I) = EF(I)
   75       CONTINUE
        END IF
        DO 80 J = 1,W
          M(J, 1) = 1
          M(J, 2) = KQ(J)
          M(J, 3) = KQ(J) ** 2
          T(1, J) = 1
          T(2, J) = KQ(J)
          T(3, J) = KQ(J) ** 2
   80     CONTINUE

        SUM = 0
        DO 110 K = 1,3
          DO 100 J = 1,3
            DO 90  I = 1,W
              SUM = SUM + T(J, I) * M(I, K)
   90       CONTINUE
            SUM1(J, K) = SUM
            F(J, K) = SUM
            SUM = 0
  100       CONTINUE
  110     CONTINUE
C         REM XTRAN * Y
        SUM = 0
        DO 130 J = 1,3
          DO 120 I = 1,W
            SUM = SUM + T(J, I) * KH(I)
  120       CONTINUE
          SUM2(J) = SUM
          F2(J) = SUM
          SUM = 0
  130     CONTINUE

        E(1,1) = F(2, 2) * F(3, 3) - F(3, 2) * F(2, 3)
        E(1,2) = -(F(2, 1) * F(3, 3) - F(3, 1) * F(2, 3))
        E(1,3) = F(2, 1) * F(3, 2) - F(2, 2) * F(3, 1)
        E(2,1) = -(F(1, 2) * F(3, 3) - F(1, 3) * F(3, 2))
        E(2,2) = F(1, 1) * F(3, 3) - F(1, 3) * F(3, 1)
        E(2,3) = -(F(1, 1) * F(3, 2) - F(1, 2) * F(3, 1))
```

```
          E(3,1) = F(1, 2) * F(2, 3) - F(1, 3) * F(2, 2)
          E(3,2) = -(F(1, 1) * F(2, 3) - F(1, 3) * F(2, 1))
          E(3,3) = F(1, 1) * F(2, 2) - F(2, 1) * F(1, 2)

          DO 150 I = 1,3
            DO 140 J = 1,3
              G(I, J) = E(J, I)
  140       CONTINUE
  150     CONTINUE
          P1 = F(1, 1) * (F(2, 2) * F(3, 3) - F(3, 2) * F(2, 3))
          Q1 = F(1, 2) * (F(2, 1) * F(3, 3) - F(3, 1) * F(2, 3))
          R1 = F(1, 3) * (F(2, 1) * F(3, 2) - F(2, 2) * F(3, 1))

          DET = P1 - Q1 + R1
          DETINV = 1 / DET

          DO 170 I = 1,3
            DO 160 J = 1,3
              G(I, J) = G(I, J) * DETINV
  160       CONTINUE
  170     CONTINUE

          SUM = 0
          DO 190 J = 1 , 3
            DO 180 I = 1 , 3
              SUM = SUM + G(J, I) * F2(I)
  180       CONTINUE
            SUM3(J) = SUM
            F3(J) = SUM
            SUM = 0
  190     CONTINUE
          A = F3(1)
          B = F3(2)
          C = F3(3)
          WRITE (9,*)  A, B, C
C          REM QUADRATIC OF FORM: A + BX + CX**2
  200   CONTINUE
      END DO
  220 CLOSE (8)
      WRITE (9,*) 99,99,99
      CLOSE (9)
      RETURN
      END
```

Vita

Newland Agbenowosi was born in Leklebi, Ghana on May 16, 1972 to Patrick Agbenowosi and Patience Agbenowosi. After completing his primary education in Datus Preparatory School, Tema, 1984, he enrolled in Presbyterian Boys' Secondary and Sixth Form Science College (PRESEC). Upon completion, he went on to attend Central State University in 1990 where he received his Bachelor of Science Degree in Water Resources Management in 1993. Newland then entered Virginia Polytechnic Institute and State University for his master of science degree in Civil Engineering concentrating in Hydrosystems Engineering.