DOCUMENTATION OF FINITE-ELEMENT MESH GENERATION PROGRAMS USING A GEOGRAPHIC INFORMATION SYSTEM

By Robert A. Lowther and Eve L. Kuniansky

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 92-4155

A contribution of the Regional Aquifer-System Analysis Program

Austin, Texas

1992

U.S. DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, Jr., Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
U.S. Geological Survey
8011 Cameron Rd., Bldg. 1
Austin, TX 78753

Copies of this report can be
purchased from:

U.S. Geological Survey
Books & Open-File Reports Section
Federal Center
P.O. Box 25425
Denver, CO 80225

CONTENTS

CONTENTS--Continued

ILLUSTRATIONS

ILLUSTRATIONS-Continued

CONVERSION FACTORS

Multiply	By	To obtain
foot (ft)	0.304	meter
mile (mi)	1.6093	kilometer

DOCUMENTATION OF FINITE-ELEMENT MESH GENERATION PROGRAMS USING A GEOGRAPHIC INFORMATION SYSTEM

By

Robert A. Lowther and Eve L. Kuniansky

ABSTRACT

Finite-element mesh generation for models of ground-water and surface-water systems is a tedious process due to the irregular nature of the geologic, hydrologic, and geomorphic features. In developing a mesh for the Edwards-Trinity aquifer-system project, a vector-based geographic information system is used to generate a variably sized triangular element finite-element mesh from mappable features. Important digitally mapped features are automatically linked to nodes in the finite-element model, ensuring an efficient, virtually error-free alternative to the tedious process of mesh design and data-input preparation by other methods. The computer programs have been developed in a macro language and may be useful for other processes. Some of the programs use commercially developed software commands, while others use Fortran programs developed specifically for the finite-element model. This report documents the programs developed for mesh generation and provides, as an example, the mesh developed for the Edwards-Trinity Regional Aquifer-System Analysis project. The programs are stand-alone and provide the necessary information on node coordinates and element connection data for three-nodal triangular elements for many finite-element model applications.

INTRODUCTION

The irregular nature of geologic, hydrologic, and geomorphic features causes finite-element mesh generation to be a tedious process for ground-water and surface-water models. The finite-element method was chosen for simulation of the Edwards-Trinity regional aquifer system due to anisotropy, which varied in direction throughout the area of the system. Programs developed for mesh generation in structural design were initially used to design the mesh of the system. The development of the original mesh took about four months to complete. This mesh was then linked to a geographic information system (GIS) in order to facilitate data preparation.

As the Edwards-Trinity Regional Aquifer-System Analysis (RASA) project evolved, the decision was made to develop a more detailed model of the most active (relative to ground-water flow) part of the aquifer system. This required the design of another mesh. Rather than spend four months manually designing a mesh, the GIS software developed by Environmental Systems Research Institute, Inc. (ESRI), ARC/INFO[1] was used to design computer programs specifically for mesh creation. The computer programs have been developed in ARC/INFO (version 5.0) macro language (AML) and have been tested on a Prime minicomputer and on Data General Aviion workstations. These AML programs, also referred to as macros, call Fortran programs or commercially developed ARC/INFO commands. The AML programs allow the modeler to efficiently generate different mesh designs.

This report documents the AML programs developed for mesh generation and the development of other model input data for simulating streams in a ground-water system. The subregional model mesh for the Edwards-Trinity aquifer system is used as an example of the process. The process for developing a mesh has been divided into several steps that use different macros. Some of these may be useful for processing geographic data other than mesh generation.

[1]Use of firm/trade names in this report is for identification purposes only and does not constitute endorsement by the U.S. Geological Survey.

In order to use the information presented in this document, the reader needs working knowledge of the finite-element method and ARC/INFO. ARC/INFO commands are presented in italicized all capitals and are referenced in the ESRI manuals (ESRI, 1987). The basic hardware and software requirements are: minicomputer or workstation, plotting device, color high-resolution graphics terminal, Fortran77 compiler, and ARC/INFO software with the TIN package.

If the features important to the model are already digitally available, a mesh can be designed in several days using the mesh generation AML programs. These macros were designed as stand-alone modules within the GIS software and are used to set up the basic node coordinate and element connection data necessary for many different finite-element model programs.

The assistance of Leonard L. Orzol, USGS, Portland, Oregon in porting all of the mesh generation programs to the Unix workstations is greatly appreciated by the authors. We also appreciate the assistance of Jonathon C. Scott, USGS, Oklahoma City, Oklahoma for providing an example program for reading the binary files within ARC/INFO.

FINITE-ELEMENT MESH DESIGN CONSIDERATIONS

The finite-element method allows two-dimensional space to be subdivided (discretized) into a mesh with variably sized elements. Certain generalizations can be made in designing a mesh. These considerations vary with the partial differential equation or equations being solved.

For ground-water flow problems, the mesh should have the smallest elements where the gradient of the potentiometric surface of the aquifer to be simulated is greatest. The equations are being solved for potentiometric head at the vertices of each element, which represents a plane whose orientation in space is defined by the nodal values of head. Thus, more elements are required to approximate surfaces having a variable slope than to approximate surfaces having a relatively uniform slope. Small elements may be required to define irregular aquifer geometry or complex geologic structures, such as faults, that affect ground-water movement.

For surface-water problems, such as two-dimensional flow over a flood plain where both velocity and water-surface elevation are unknown, smaller elements are necessary where there are sharp gradients in velocity or depth of bed. This usually occurs in and around the stream channel.

For all finite-element problems, the size and shape of the elements affect the solution of the equations. Theoretically, as the size of the elements approaches zero, the solution approaches the true solution of the equation. However, as the element size decreases, the number of equations solved approaches infinity and roundoff error increases. The size and number of elements for each model is problem dependent. One must create a mesh with enough finite-elements for approximating the unkown variable without creating an unmanageable number of simultaneaous equations to solve (If you have access to a super-computer you could try for 30,000 nodes, regardless of over-discretization.).

The shape of elements affects the solution because the finite-element method uses interpolation functions, also called shape functions or basis functions, that are derived from the coordinates of the nodes for each element. For the solution of the ground-water flow equation by the Galerkin finite-element method, triangular elements with angles between 22.5° and 90° result in coefficient matrices with positive diagonal terms and negative off-diagonal terms (Strang and Fix, 1973, p. 138-139). Strang and Fix (1973, p. 78) also state that the sum of two adjacent angles between elements should not be greater than 180°. This allows one obtuse angle without causing the final matrix to be less well conditioned for the equations' solution. The most numerically stable triangular-element mesh is one of regularly spaced equilateral triangles (Strang and Fix, 1973).

When a direct solver is used for solving the set of equations developed by the finite-element approximation of the problem, node numbering of the elements becomes important. The mesh generation programs incorporate an algorithm developed by Collins (1973) for renumbering the nodes to minimize the bandwidth of the matrix solved by the finite-element programs.

If a large number of triangular elements are connected at one vertex (patch of elements), the bandwidth of the matrix to be solved is increased. This will create a matrix that is sparse (many zeros between numbers in the rows and columns), which leads to increased roundoff errors when using a direct solver. This patch may also have thin needle-like elements, which are undesirable because the resulting triangles would have angles less than 22.5° or greater than 90°. Patches of more than eight or nine triangular elements should be avoided.

The authors' experience with surface-water and ground-water finite-element mesh designs has been that these considerations are rarely met for all elements in a mesh. If an attempt is made to minimize the number of poorly shaped elements and patches of elements, while incorporating the important geomorphic and geologic features, numerical stability will not be a problem.

OVERVIEW OF THE MESH DESIGN PROCESS WITH EXAMPLE

The basic mesh design process consists of three major parts: determination of relevant input features, generalization of those features, and generation of a triangular mesh based on those features. Part one consists of two steps. Part two consists of seven steps. Part three consists of one step. After the basic mesh design has been completed with these 10 steps, an optional eleventh step may be useful for some meshes: the assigning of altitudes to certain feature-based points in one of the output coverages. The details of each of these 11 steps follow.

Part one is to determine which features are required for the design of the mesh. These features must exist as ARC/INFO coverages. The mesh designer must also know the features that define the model boundary and the features that require a finer mesh because of an anticipated change in potentiometric surface.

Part two is the generalization of the features, necessary because feature coverages will tend to have more detail than is feasible to include in the model. This is accomplished by reducing the number of line vertices with *SPLINE* and by reducing the number of points in the model with *SNAP*. All feature coverages should have a unique user-ID for each point, and *IDEDIT* should be used to update the coverages whenever a change is made to their user-ID's.

Part three is to generate a mesh, beginning with a point cover based on the input features. Arcs are reduced to their vertex points. All feature-based points, both those that were originally line vertices and those that were in a point cover, are *SNAP*ped to reduce the total number of points in the model. For areas in which few points are generated from these features, regularly spaced grids of points will be used. Because the elements are triangles whose vertices are the aforementioned points, this ensures a more uniform element size across the entire model.

The remainder of the process is to generate three GIS data layers (coverages) related to the points from the previous steps and two ascii files, FILEECD and FILENCD. These three coverages can be used for creation of other needed input data to the numerical model that is used. All coverages will have the same user-supplied root name, and have the extensions .ELMS, .ELPT, and .NOD. The .ELMS cover is a polygon cover, composed of triangles that have, as vertices, the points generated in the previous steps and whose user-ID's are the element numbers of the mesh. The .ELPT cover is a point cover made up of the centroids of the triangles in the .ELMS cover and having user-ID's as the element numbers of the mesh. The .NOD cover is a point cover whose points are the vertices of the triangles in the .ELMS cover. The user-ID's of the points in the .NOD cover are the model node numbers. Attributes can then be assigned to the .ELPT points, whose user-ID's correspond to the polygon user-ID's in the .ELMS cover of triangular elements. FILEECD has the element connection data and FILENCD has the node-coordinate data for the finite-element model. The three major parts in the process are contained in the 11 detailed steps that follow. For quick reference, however, a summary of the necessary commands for each step is included in the supplementary data section "Mesh generation procedure quick-reference guide."

To facilitate understanding of the mesh generation process, it is described for the subregional model of the Edwards-Trinity aquifer system (fig. 1). The area containing all features, and, consequently the mesh, is called "the study area." This particular mesh uses 10 feature coverages incorporating point, line, and polygon data.

The feature coverages are: faults, FAUCL, a line cover; surface drainage divides, BASUB, a line cover; subsurface limits of geologic formations, SUBGLIM, a line cover; irrigation pumpage, IRRPUM, a point cover; municipal and industrial pumpage, MUINPUM, a point cover; major springs, SPRINGS, a point cover; streams, STR, a line cover; geographic subregions representing the Edwards and Trinity aquifers, TREDAQ, a polygon cover; outcrop of high permeability rocks, OCRCL, a polygon cover; and the Devil's River trend, an ancient reef structure , DEVTR, a polygon cover. They are shown in figures 2-11.

Feature covers of the Edwards-Trinity aquifer system that are used for mesh generation.

Feature	Cover Name	Feature Class
Faults	FAUCL	lines
surface drainage divides	BASUB	lines
boundaries of geologic formations	SUBGLIM	lines
Irrigation pumpage	IRRPUM	points
Municipal and industrial pumpage	MUINPUM	points
Major springs	SPRINGS	points
Streams	STR	lines
Subregions of the Edwards and Trinity aquifers	TREDAQ	polygons
Outcrop area	OCRCL	polygons
Boundary of the Devils River Trend (an ancient reef)	DEVTR	polygon

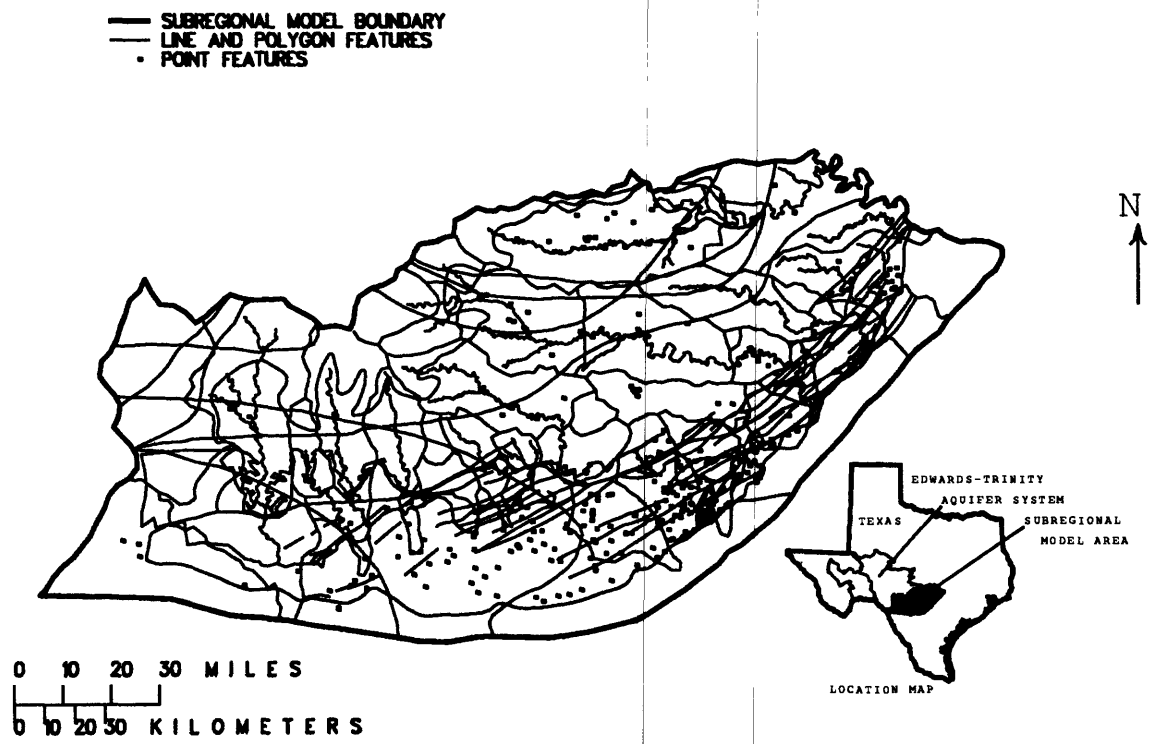


Figure 1.--The subregional model area, including significant line, polygon, and point features.

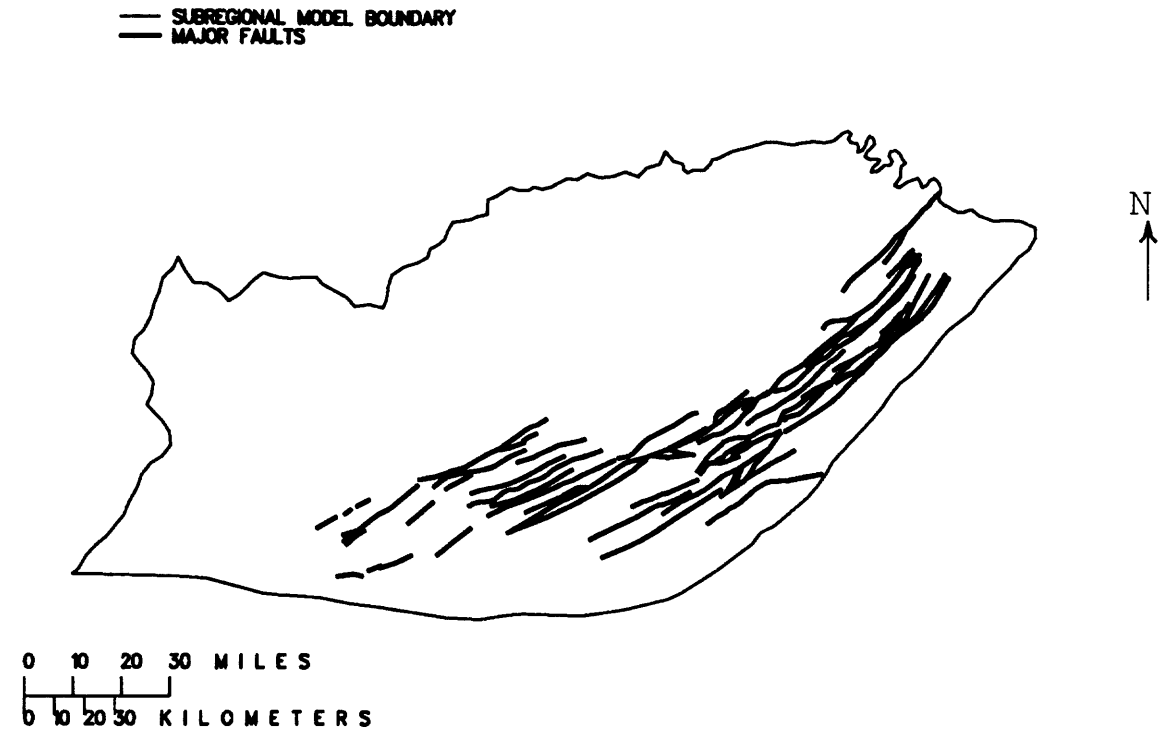


Figure 2.--Location of major faults in the study area (FAUCL).

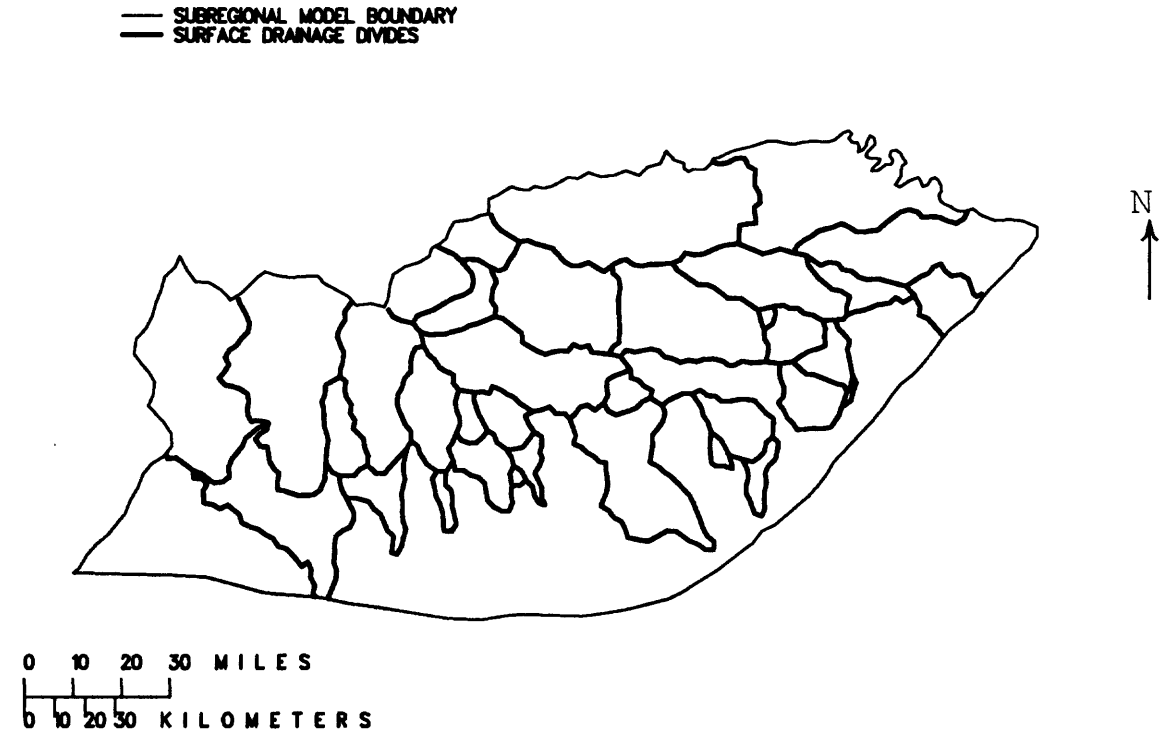


Figure 3.--Location of surface drainage divides in the study area (BASUB).

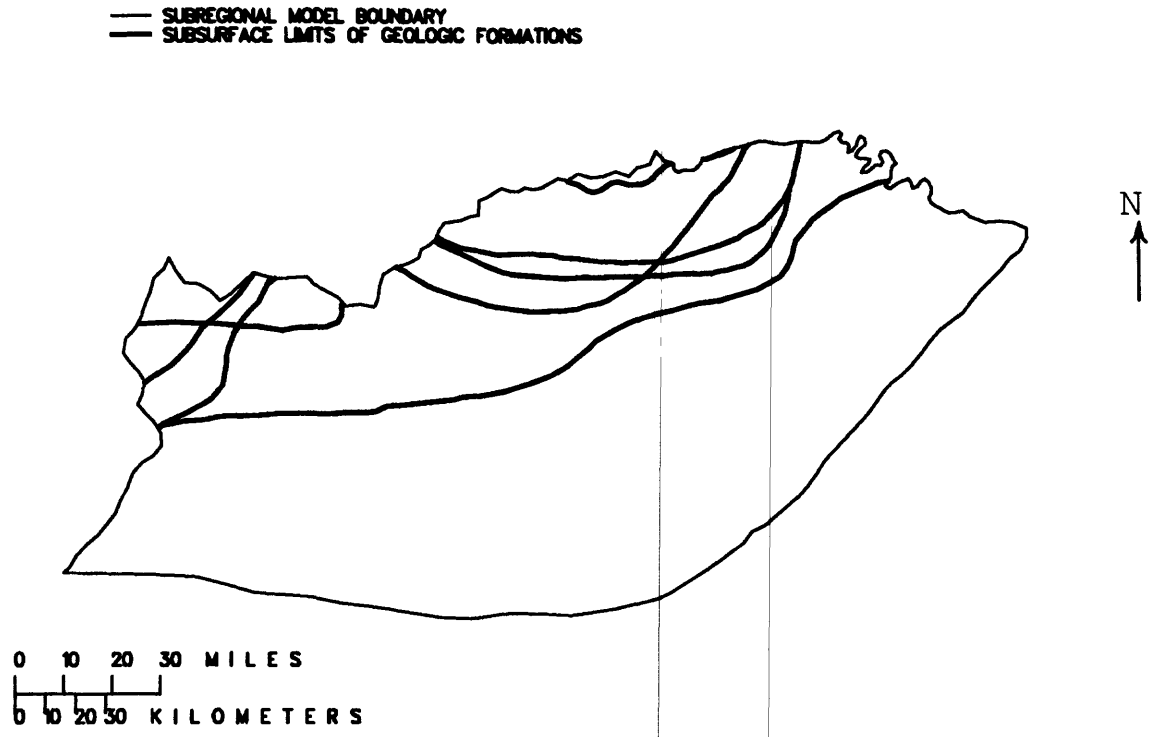


Figure 4.--The subsurface limits of geologic formations in the study area (SUBGLIM).

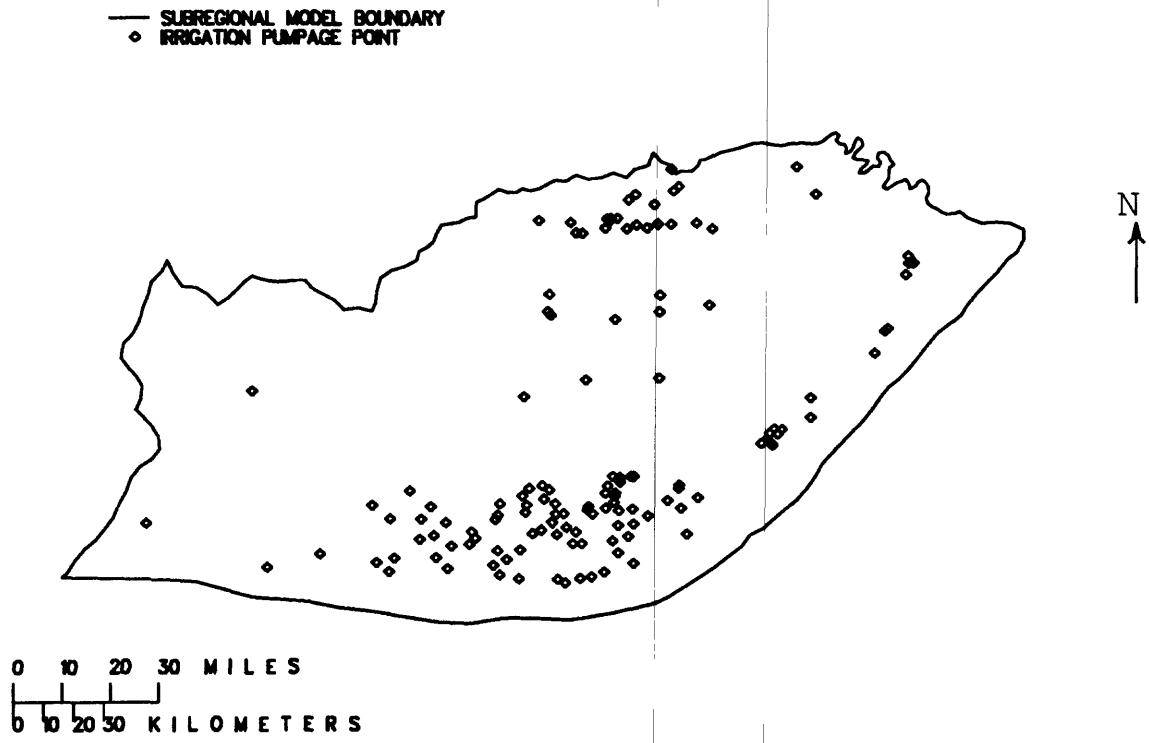


Figure 5.--Location of irrigation pumpage in the study area (IRRPUM).

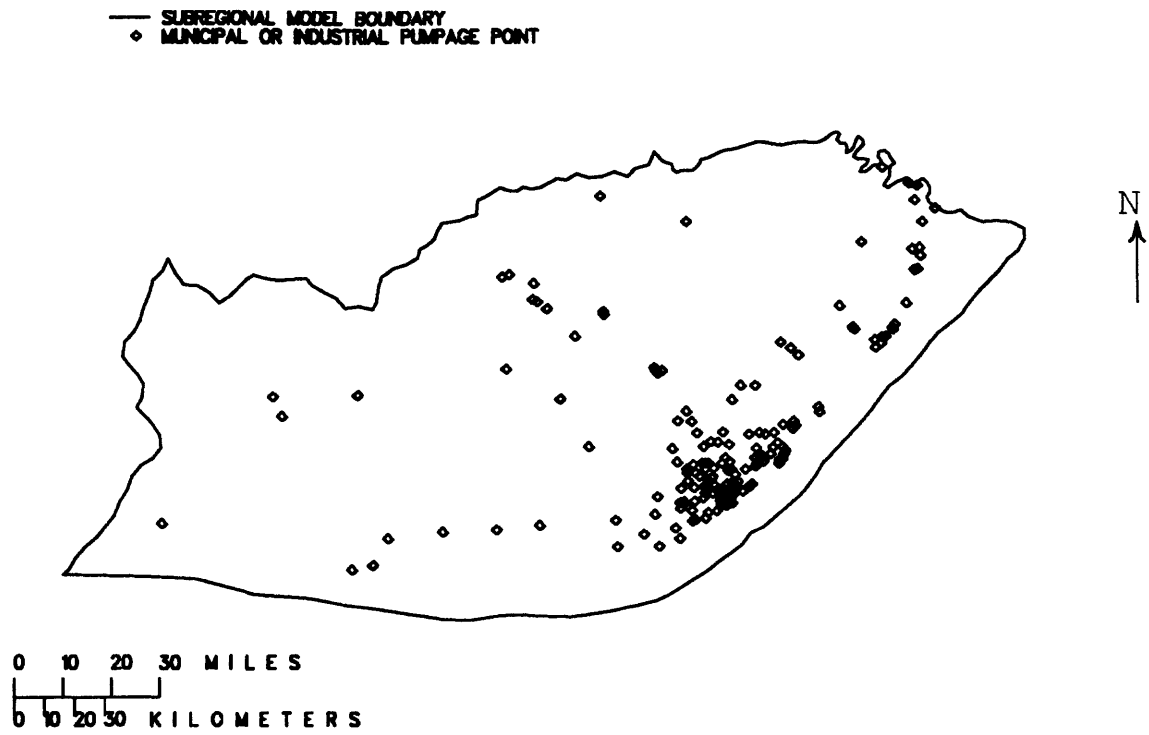


Figure 6.-- Location of municipal or industrial pumpage in the study area (MUINPUM).

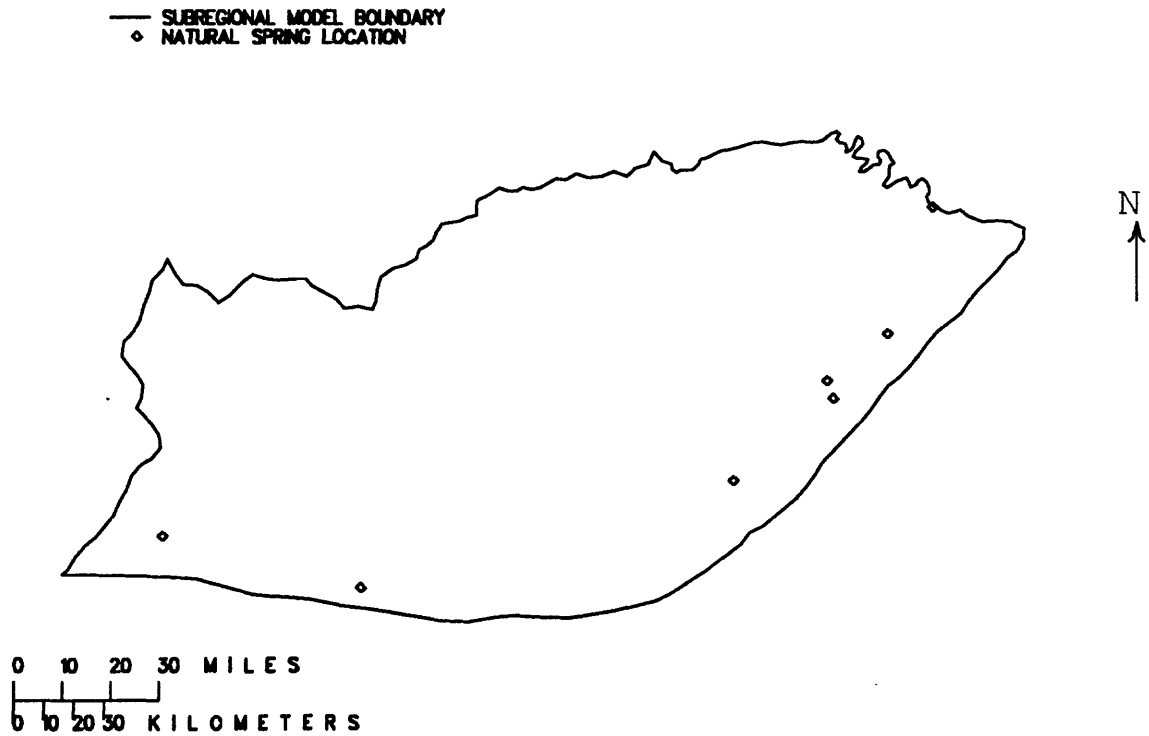


Figure 7.--Location of major springs in the study area (SPRINGS).

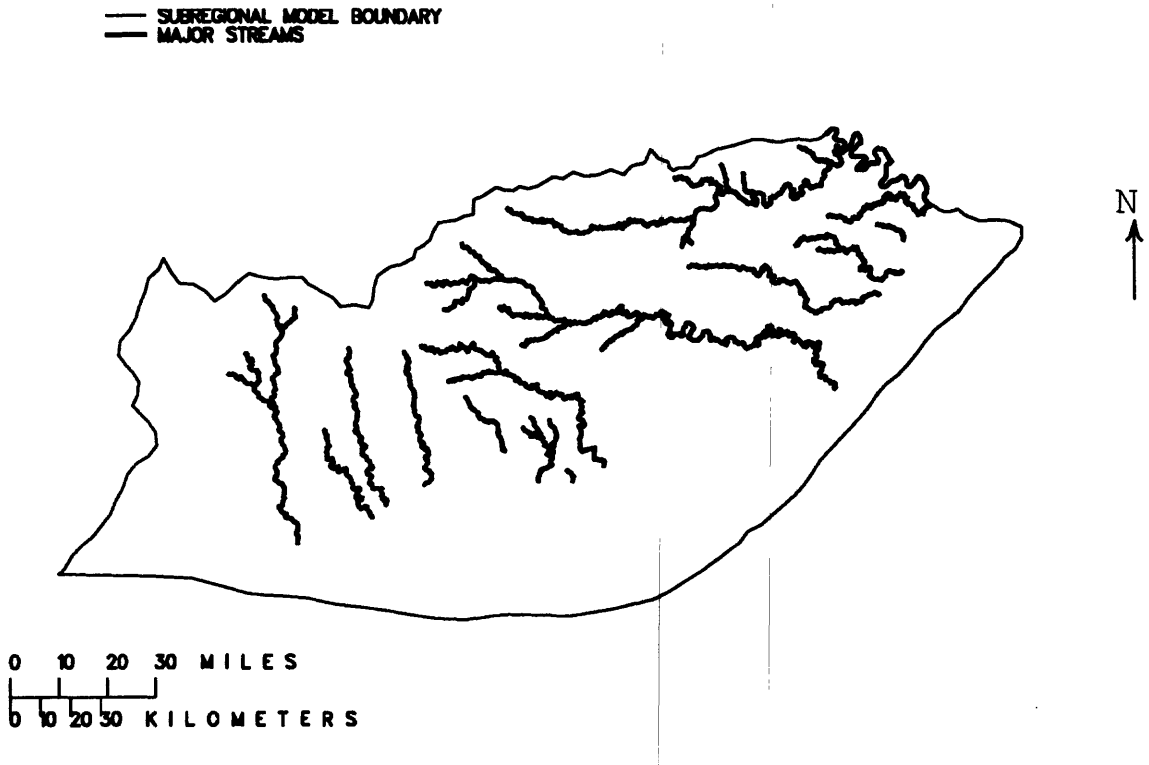


Figure 8.--Location of major streams in the study area (STR).

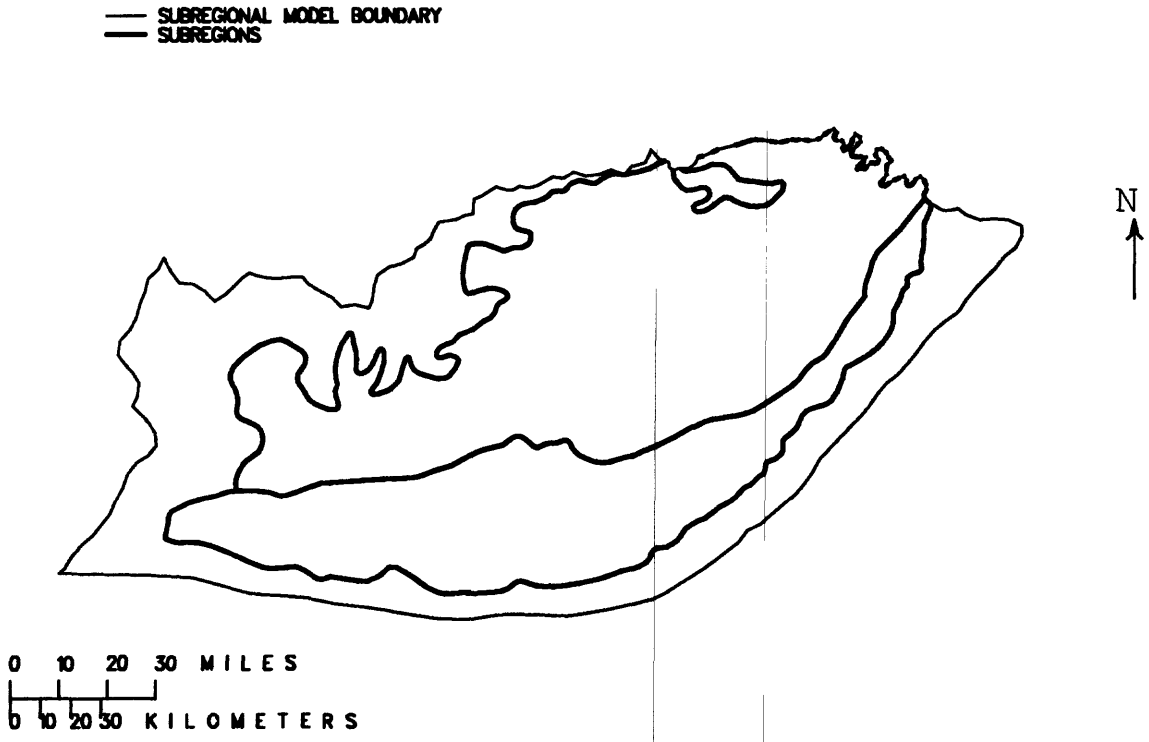


Figure 9.--Location of subregions in the study area (TREDAQ).

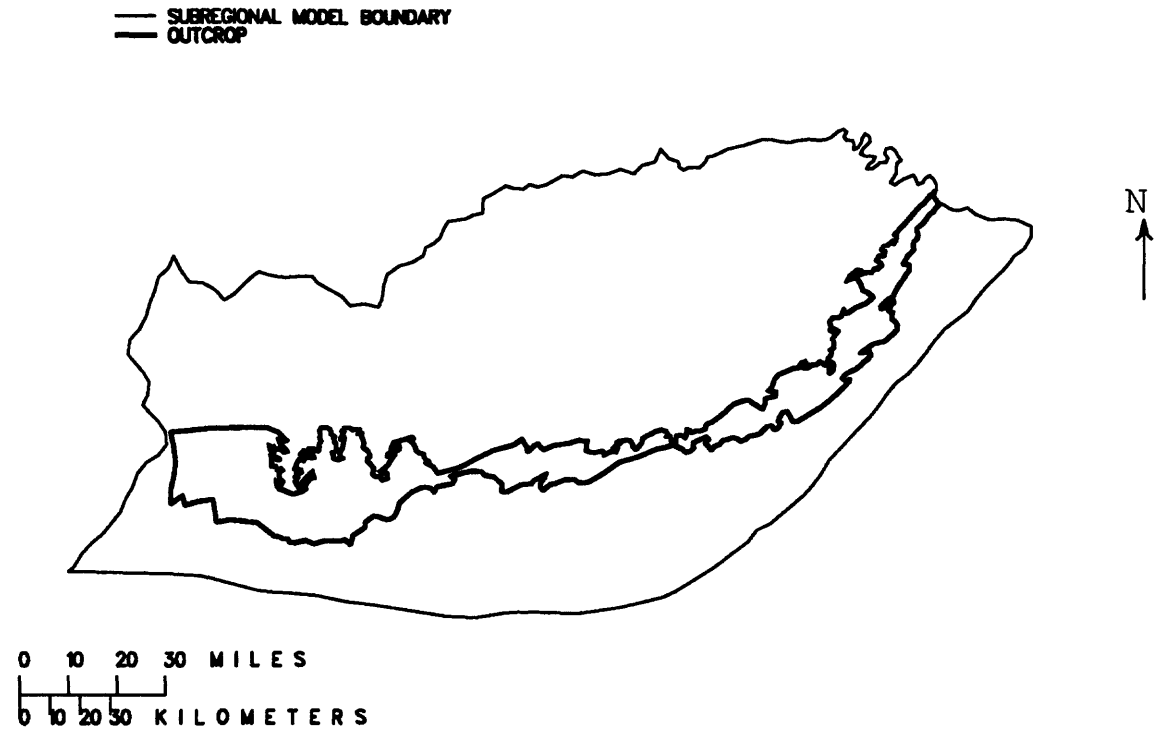


Figure 10.--Location of geologic outcrop in the study area (OCRCL).

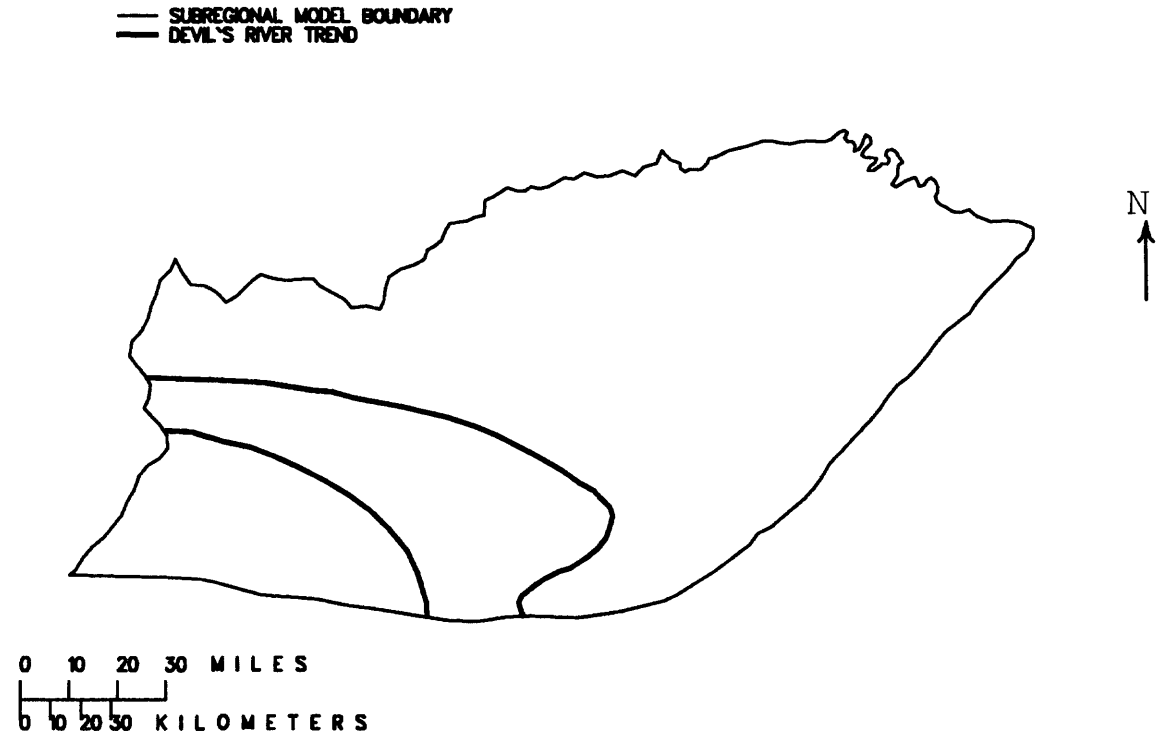


Figure 11.--Location of the Devil's River Trend in the study area (DEVTR).

For model the of the Edwards-Trinity aquifer system, the boundary was made up of several different data layers for hydrogeologic reasons. The northwestern boundary was based on surface drainage divides and was simulated as a no-flow boundary (part of BASUB). The aquifer system is a water-table system in the northwestern area. Potentiometric-surface maps of the system indicate that ground-water flow parallels surface drainage. The southern part of the system is confined by a wedge of post-Cretaceous sediments composed of clays with smectite minerals. This part of the Edwards-Trinity system dips steeply toward the coast. The permeability of the Edwards-Trinity aquifer system diminishes rapidly along the southern edge of the aquifer. The low permeability area is coincident with a freshwater/saline-water transition zone, an interface of freshwater (less than 500 parts per million (ppm) dissolved solids) to the northwest and moderately saline-water (1,000 to 10,000 ppm dissolved solids) to the southeast. The southern boundary of the model is simulated as a no-flow boundary several miles within the low permeability zone and is found in a cover called BADH2O. The northeastern boundary of the model is the Colorado River, which is simulated as a head-dependent source/sink (part of STR).

Part One. Select Important Features

Step 1.1 is to decide which features are important to the mesh. Any set of features that is not in an ARC/INFO cover should be created and built (*BUILD*) as the appropriate cover type (point, polygon, or line). Features of each type should be in separate coverages.

Step 1.2 is to make basic decisions about the mesh. A model boundary must be decided upon. This boundary is simply the outer edge of all features of interest for the model, and will form the outline of the mesh. A model boundary may consist of parts of several coverages. Also, any features whose level of detail requires a more finely spaced mesh than the rest must be identified. One such common cover is the stream cover, whose arcs often have many crenulations.

Part Two. Generalize Features

Step 2.1 is to define the subarea whose greater detail necessitates a finer-spaced mesh, as decided upon in step two. An area is specified by creating an ARC polygon cover with only one polygon that surrounds the area of interest. This cover is referred to as a "polygon outline." The Polygon Attribute Table (PAT) of the cover should contain a special integer item, the "identifying item." This item should be set to a value of "1" for the polygon area inside of the arc defining the desired area. It should be set to "0" for all other areas (specifically, the polygon listed in the PAT, which has a negative AREA, and is defined to be all space outside of any other polygons in the PAT). All polygon outlines will have to be defined, or built (*BUILD*), as polygon coverages (that is, have PAT's); and some will also have to be built (*BUILD*) as line coverages (that is, have Arc Attribute Tables, or AAT's). Fortunately, these two definitions are simultaneously possible in the ARC/INFO system.

An area of detail can either be defined as all space within a certain distance from a specified point or line geographic feature or as a polygon included in the geographic model features. If the former is the case, then that area must be defined in ARC/INFO. In order to define such an area in ARC, we *BUFFER* the given feature. This is done in two stages:

First, if the cover forming the basis for the subarea is a line cover, then this cover must be *SPLINE*ed at an appropriate distance. If the coverages are point coverages, then this stage is unnecessary. We do not want to directly *SPLINE* coverages at this point, as this would alter them and thereby distort data that we will later need. We therefore make temporary copies of the coverages, designated <cover name>.TEMP, and *SPLINE* these coverages instead. SPLEEN.AML is used at this point to *SPLINE* the coverages.

A good rule of thumb is to *SPLINE* at either the smallest desired distance between two points in the mesh, or 1/10th of the *BUFFER* distance, whichever is larger. *SPLINE*ing is necessary in order to prevent anomalies from arising when the cover(s) are *BUFFER*ed. Attempting to *BUFFER* a convoluted line at any relatively great distance will produce such anomalies and prevent proper operation of the *BUFFER* command.

The second stage of step 2.1 is to create a polygon outline of the area requiring a finer-spaced mesh. If more than one cover contains features forming the basis for a region, the temporary copies of the relevant coverages should be combined into one cover. Any arcs or points in these basic feature coverages that are not part of the detailed-area definition should be removed, in *ARCEDIT*, from the temporary copies before they are combined. Once all of, and only, the relevant features are assembled into one cover, we can proceed to step 2.2.

We used three different levels of detail for the Edwards-Trinity aquifer-system model. Two polygon outlines, BUFB and BUFL, were created to divide the three areas. We create these coverages by *BUFFER*ing a cover combining desired features from the stream cover copy, STR.TEMP, and the fault line cover copy, FAUCL.TEMP, at two different *BUFFER* distances. The desired features have been extracted from separate coverages and combined into DTAILAREA (fig. 12). For our model, these features require more detail because they cause sharper variations in aquifer head.

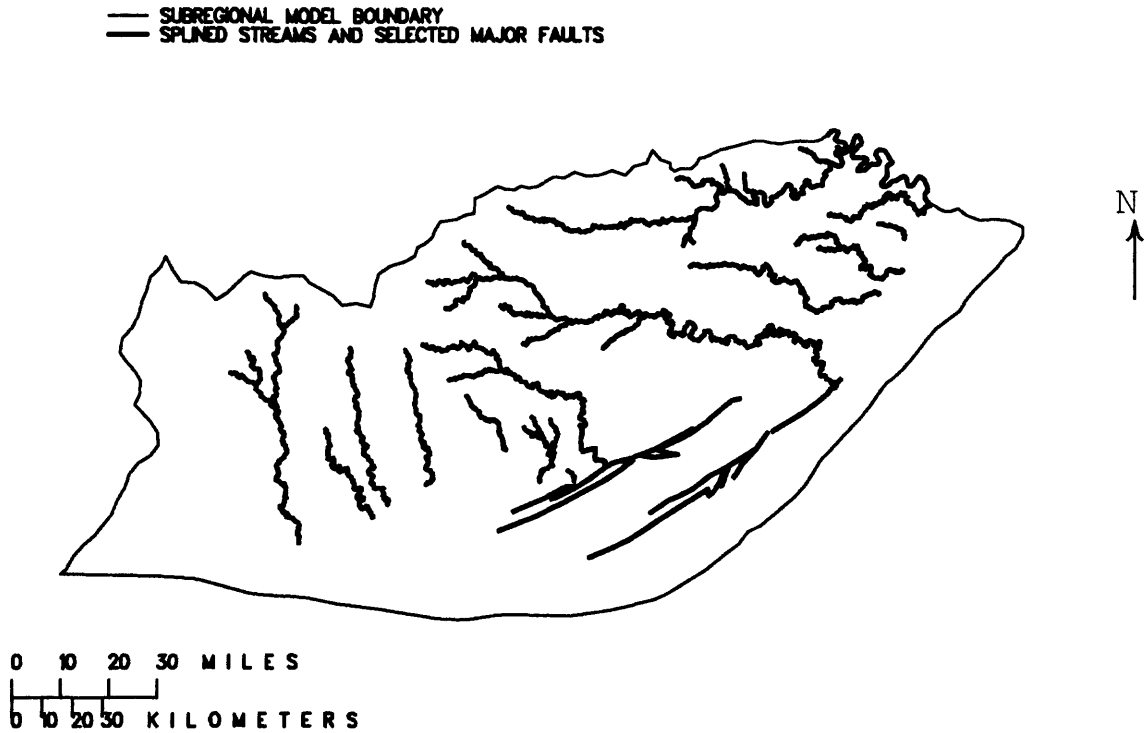


Figure 12.--Location of feature lines forming the basis for areas of greater detail in the model.

We use BUFFNSHINE.AML to *BUFFER* DTAILAREA at 60,000 and at 20,000 ft. to create the coverages BUFB and BUFL respectively (figs. 13 and 14). BUFFNSHINE.AML also adds an identifying item to the PAT's of the coverages it creates. For our example, the items are INBUFB and INBUFL. The *BUFFER* distances of 20,000 and 60,000 ft. were chosen based upon the average size of the triangles to be generated within these areas of detail. The relative sizes of triangles are subjective decisions based on the scale and detail of the modeling effort.

If the detail area was already in the feature coverages, then *COPY* and *ARCEDIT* should be used to create a cover whose arc outline is the desired area. Unwanted interior arcs are removed, and an identifying PAT item for the interior area is defined with MAKOUTLIN.AML. As with BUFFNSHINE.AML, the user must supply an identifying item name for MAKOUTLIN.AML. The output will be a polygon outline.

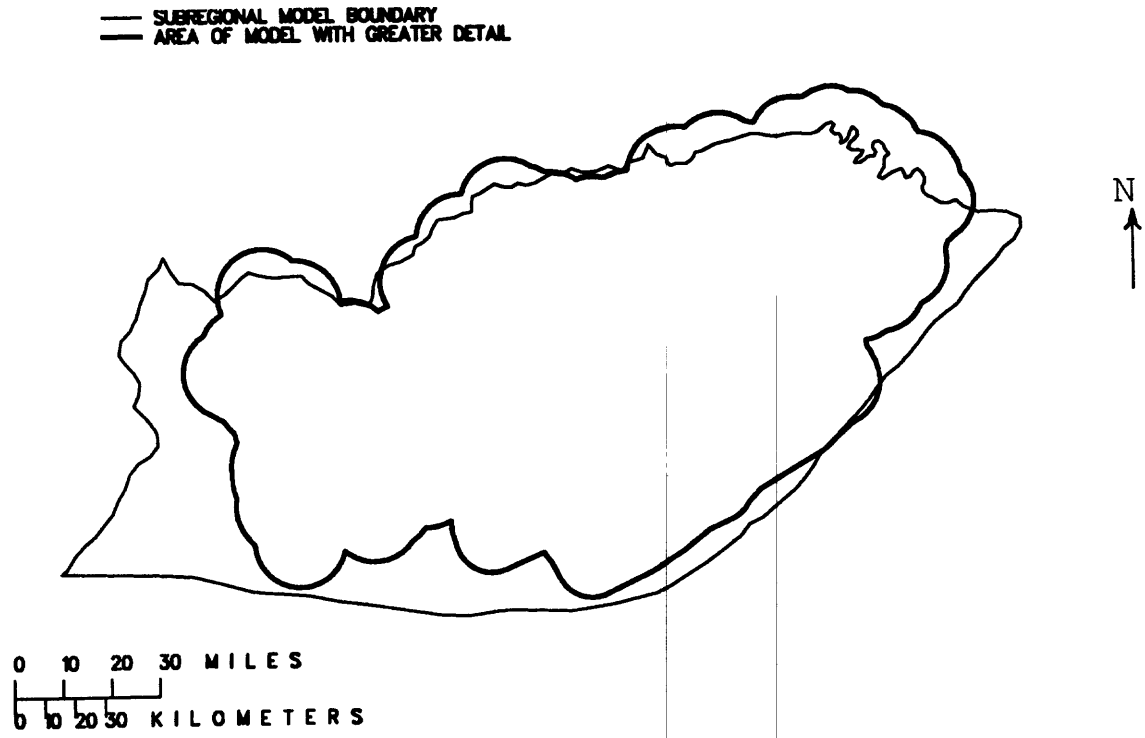


Figure 13.--Location of the area with greater detail in the model.

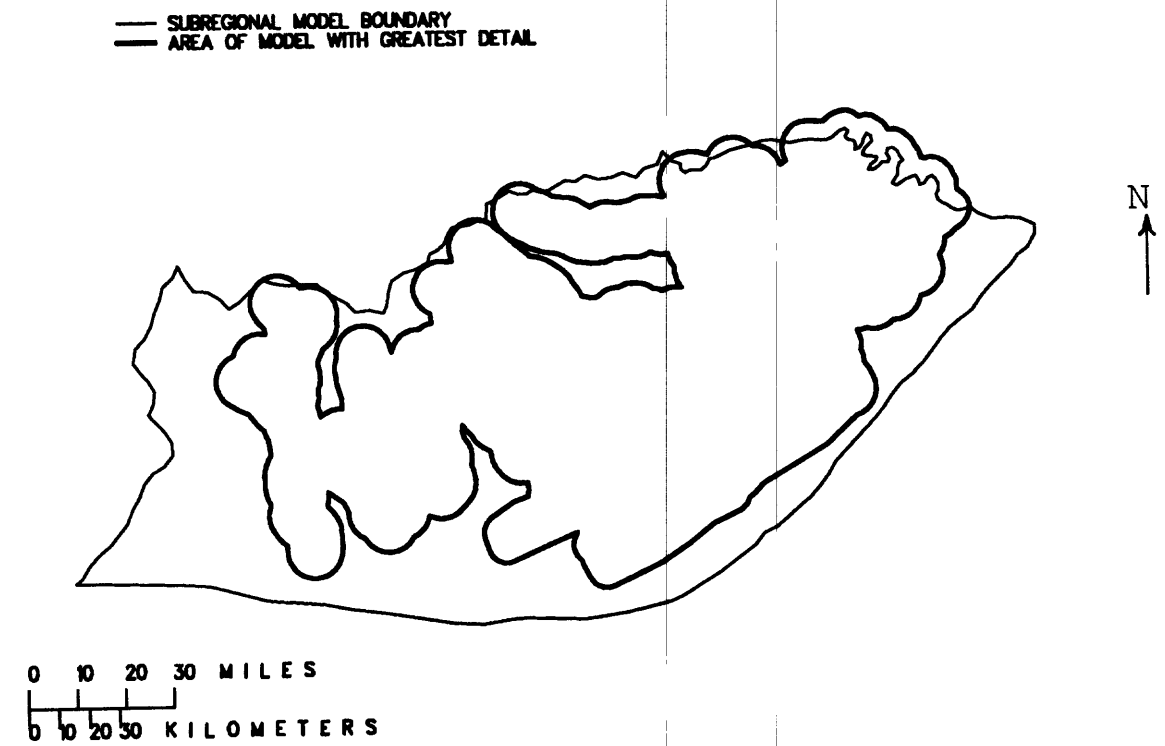


Figure 14.--Location of the area with greatest detail in the model.

For the example mesh, the commands used for step 2.1 are:

```
COPY STR STR.TEMP
COPY FAUCL FAUCL.TEMP
&R SPLEEN 4207 8000 0 0 STR.TEMP FAUCL.TEMP
(Use ARCEDIT to remove unwanted arcs from the FAUCL.TEMP cover)
APPEND DTAILAREA (with the following entries)
 STR.TEMP
 FAUCL.TEMP
 END
BUILD DTAILAREA LINE
&R BUFFNSHINE DTAILAREA LINE 60000 INBUFB
&R BUFFNSHINE DTAILAREA LINE 20000 INBUFL
KILL STR.TEMP ALL
KILL FAUCL.TEMP ALL
```

Step 2.2 in the mesh design process is to implement the model boundary decision made in step 1.2 and define the study area. The model boundary is simply a *SPLINE*ed polygon outline of the total study area. It is used to *CLIP* other coverages and thereby eliminate any features outside of the study area.

To create the model boundary, use *ARCEDIT* on a temporary copy of each cover whose features define the boundary, and combine these copies into one cover of the model boundary. If several feature coverages duplicate the same model boundary, use the most detailed feature. This most commonly occurs where a stream coverage has been digitized with enough detail for accurate length and a political boundary duplicates the stream. The political boundary should be deleted and the stream used as the boundary for the model. Features that approximate, but do not duplicate, parts of the model boundary must be eliminated from that feature cover.

Stream coverages must be treated specially if the stream length is important for other aspects of the modeling effort. A special AML for computing the actual stream length associated with a node is included in this documentation. It will be described in detail in the section REALENGTH.AML. Keep in mind that *SPLINE*ed and non-*SPLINE*ed copies of the stream cover must contain the same stream segments. If the non-*SPLINE*ed cover contains streams that run along the model boundary, so must the *SPLINE*ed cover. In order for the *SPLINE*ed cover to contain streams along the model boundary, it must duplicate that boundary, not approximate it.

The relevant parts of the three model boundary-defining coverages from the Edwards-Trinity subregional model example are shown in figure 15. These are parts of *SPLINE*ed copies of the Colorado River, the surface drainage divides, and the updip limit of the freshwater/saline-water transition zone. Features that lay outside of the study area or were already in another of the temporary copy coverages werw removed. (In the Edwards-Trinity example, a segment of the Trinity aquifer boundary that is approximately coincident with a segment of a model boundary-defining drainage divide had to be removed.)

After deleting unwanted features, *APPEND* the coverages into one master cover. For the example, the three relevant coverages (fig. 15) are combined into MODBASIS. Use MAKOUTLIN.AML to create a polygon outline, the model boundary, with an identifying item from this master cover. The example model boundary, MODBND, has the identifying item "INMOD." The model boundary looks exactly like the master cover, but is a polygon as opposed to a line cover and has an identifying item in its PAT. Also *BUILD* the model boundary as a line cover, as it will need to be both a line cover and a polygon cover for KITSINK.AML to run properly.

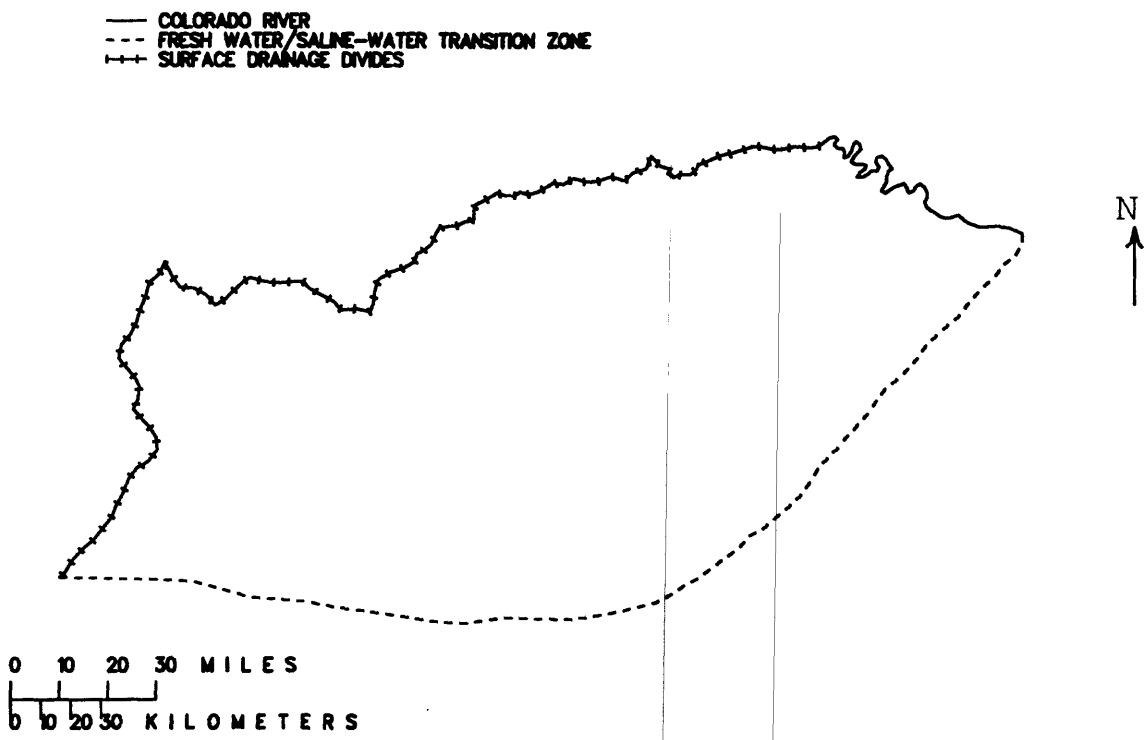


Figure 15.--Location of the model boundary.

For the example mesh, the commands used are:

```
COPY STR STR.TEMP
COPY BASUB BASUB.TEMP
COPY BADH2O BADH2O.TEMP
(Use ARCEDIT to remove unwanted parts of all three coverages)
APPEND MODBASIS (with the following entries)
 STR.TEMP
 BASUB.TEMP
 BADH2O.TEMP
 END
BUILD MODBASIS LINE
&R MAKOUTLIN MODBASIS MODBND INMOD
BUILD MODBASIS LINE
KILL STR.TEMP ALL
KILL BASUB.TEMP ALL
KILL BADH2O.TEMP ALL
```

Step 2.3 is to *SPLINE* all of the line coverages to be used for the mesh. This is done to generalize the features so that the final mesh will have a manageable number of points. Before doing this, however, an archive copy must be made of any stream cover in the model. This is because stream-length segments will be assigned to the points representing the stream in process step 3.1. For this to occur correctly, a non-*SPLINE*ed copy of the stream cover must exist. Therefore, at this point in the process, create an archival copy of the stream cover. In the example, we *COPY*ed STR into STR.LEN. After this, *SPLINE*ing all of the coverages (including the regular copy of the stream cover, STR). This is done with SPLEEN.AML.

SPLEEN.AML will allow *SPLINE*ing at different distances based on the areas defined in step 2.1, up to two specialized areas or a total of three *SPLINE* distances (two detailed area distances plus the general *SPLINE* distance). For the example mesh, we used a *SPLINE* distance of 8,000 ft. inside the subarea BUFL, a distance of 14,000 ft. inside the subarea BUFB, and a general distance of 22,000 ft. (The *BUFFER* distance of 20,000 ft. for BUFL was chosen to allow at least two 8,000-ft. equilateral triangles on each side of a stream or relevant fault; the *BUFFER* distance of 60,000 ft. for BUFB was chosen to allow two 14,000-ft. equilateral triangles between BUFL and BUFB. 60,000-20,000 = 40,000 and 40,000/14,000 > 2)

For the example mesh, the commands required for this step are:

```
COPY STR STR.LEN
&R SPLEEN 4207 22000 14000 8000 MODBND FAUCL BASUB SUBGLIM STR TREDAQ
 OCRC DEVTR (with the following entries)
  BUFB
  INBUFB
  BUFL
  INBUFL
```

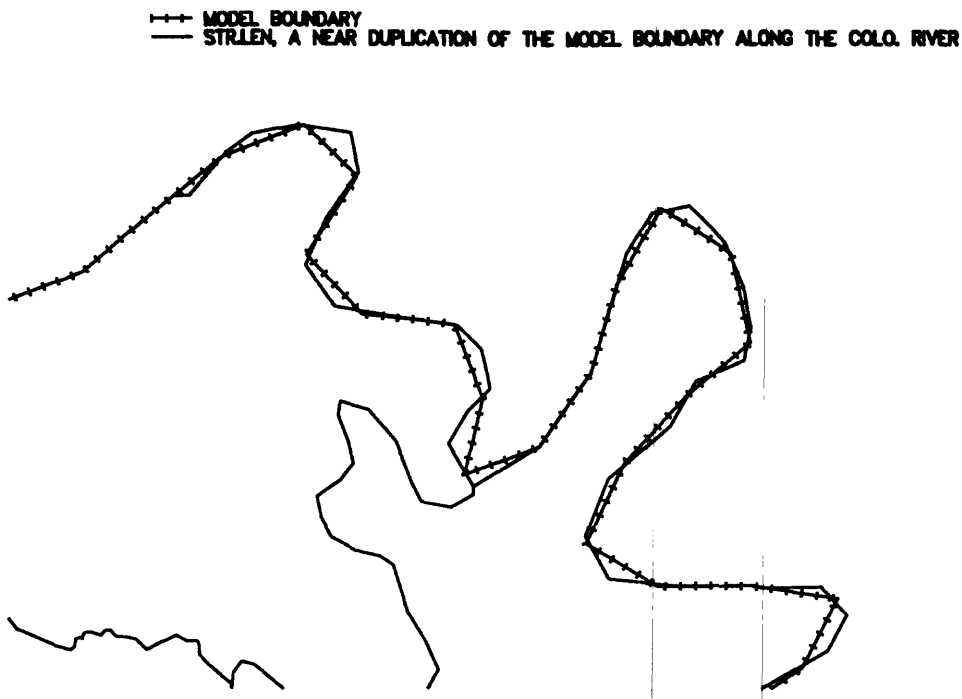
Step 2.4 involves the creation of archival copies of the *SPLINE*ed coverages for step 3.1 of the process to help identify the origin of each point in the final mesh cover. Use the *COPY* command and the naming convention "(first three letters of file name).A" to make a copy of each of the *SPLINE*ed input coverages (including the now-*SPLINE*ed stream cover). Be sure to follow the naming convention carefully, as KITSINK.AML will look for files based on this convention.

For the example mesh, the commands required for this step are:

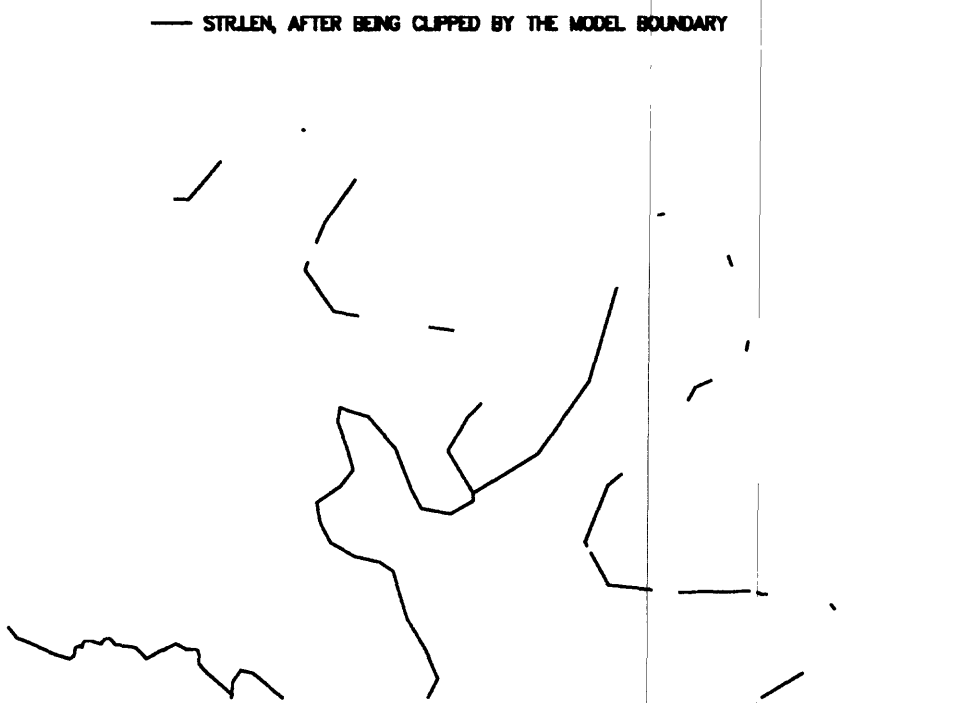
```
COPY FAUCL FAU.A
COPY BASUB BAS.A
COPY SUBGLIM SUB.A
COPY STR STR.A
COPY TREDAQ TRE.A
COPY OCRCL OCR.A
COPY DEVTR DEV.A
```

Step 2.5 is the final *CLIP*ping of the coverages of interest. Use CLIPIT.AML on these coverages, as it will allow you to use the model boundary to *CLIP* all of the coverages of one type in one operation. The output coverages from CLIPIT.AML will have the same names as the input coverages, but with ".CL" appended to the names to indicate that they have been *CLIP*ped. For the example, MODBND was used to *CLIP* all 10 coverages. Also, be sure to *CLIP* the non-*SPLINE*ed stream cover. This cover must contain only the streams and stream segments used in the model to generate feature points. If the non-*SPLINE*ed copy differs in extent from the regular cover, then the stream-length algorithm will not operate correctly.

Be sure to remove any features that nearly duplicate a part of the model boundary. This would not include a river segment which defines, and so is the same as, the boundary. Figure 16 shows an example of the complications arising from near duplication of feature lines. Part of the model boundary is nearly duplicated by STR.LEN (fig. 16a). If *CLIP*ping occurs without the removal of the duplicate line segment, then the result is a series of short, meaningless line segments (fig. 16b). For example, trying to *CLIP* the non-*SPLINE*ed cover with *CLIP* may create these line segments. If this occurs, then "*CLIP*ping" the non-*SPLINE*ed cover may have to be done interactively, in *ARCEDIT*, in order to ensure that any stream that follows the model boundary, and is included in the normal stream cover, is included as a continuous line in STR.LEN.



a. Before *CLIP*ping



b. After *CLIP*ping

Figure 16.--Illustration of detrimental effects of CLIPping a cover with features that approximate, but do not duplicate, the edge of the CLIPping cover.

For the example mesh, the commands required for this step are:

```
&R CLIPIT 4207 MODBND INMOD 0 LINE FAUCL IN BASUB IN SUBGLIM IN STR IN
 TREDAQ IN OCRCL IN DEVTR IN
&R CLIPIT 4207 MODBND INMOD 0 POINT IRRPUM IN MUINPUM IN SPRINGS IN
```

(Use ARCEDIT to remove line segments from STR.LEN so that it includes all of, and only the, streams in STR.CL.)

The part of STREAMS used to define part of the model boundary should be included in both STR.CL and STR.LEN.

Step 2.6 consists of interactively editing each of the input model coverages using *ARCEDIT*. This editing is similar to that done to the temporary copies of the model boundary-defining coverages in step 1.4. The *CLIP*ping operation may leave small, unwanted parts of feature lines. These line segments are created when a feature line approximates, but does not duplicate one of the *CLIP*ping boundaries (as noted above). Again, be sure that the normal stream cover and the non-*SPLINE*d copy represent the same set of streams and stream segments.

Step 2.7 is the creation of the regularly spaced grid(s) of points that will fill the gaps left in the feature point cover. The irregular nature of feature-generated points in the model means that, in some areas, the points are much further apart than in others. In order to avoid creating huge triangles in these areas when the line mesh is created, we must fill these gaps with points. For uniformity, these points should be regularly spaced. We therefore create a regularly spaced grid of points. We create as many grids as we desire different point spacings (usually smaller spacings in the areas of the model requiring greater detail and hence having denser feature points). These grids are separated by the polygon outlines defined in step 2.1. Use TRIANGLE.AML as many times as necessary to create the desired grid(s). The model boundary cover should be used as the "background cover" for these operations, and the points created should adequately cover the entire outline (fig. 17). In the example, there are three different grid sizes separated by BUFL and BUFB. The outer grid has a spacing of 22,000 ft. between points, with spacings of 14,000 and 8,000 ft. for the grids within the big and little buffers, respectively (fig. 18). The grid spacing used in detailed areas can be the same as the *SPLINE* distances used in the same areas. This helps to ensure more uniform triangles.

If more than one triangular grid is necessary (as in our example), then the grids must be combined to create a master grid of varying point density. Referencing the differing grid border polygon outlines created in step 2.1, use CLIPIT.AML to remove the unwanted parts (either within or without the polygon outlines) of each grid. In figure 18, the intermediately spaced point grid is used between BUFL and BUFB as a transition between the sparse and the dense grids. Be sure to *CLIP* each grid with the model boundary, to ensure that no points fall outside of it. *APPEND* these grids into a master cover to complete the grid. The example master grid is shown in figure 18.

For the example mesh, the commands required for this step are:

```
&R TRIANGLE 4207 MODBND GRDBIG 22000
&R TRIANGLE 4207 MODBND GRDMED 14000
&R TRIANGLE 4207 MODBND GRDSML 8000
&R CLIPIT 4207 MODBND INMOD 0 POINT GRDBIG IN GRDMED IN GRDSML IN
&R CLIPIT 4207 BUFB INBUFB 0 POINT GRDBIG.CL OUT GRDMED.CL IN
&R CLIPIT 4207 BUFL INBUFL 0 POINT GRDMED.CL.CL OUT GRDSML.CL IN
APPEND MSTGRD (with the following entries)
 GRDBIG.CL.CL
 GRDMED.CL.CL.CL
 GRDSML.CL.CL
 END
BUILD MSTGRD POINT
```

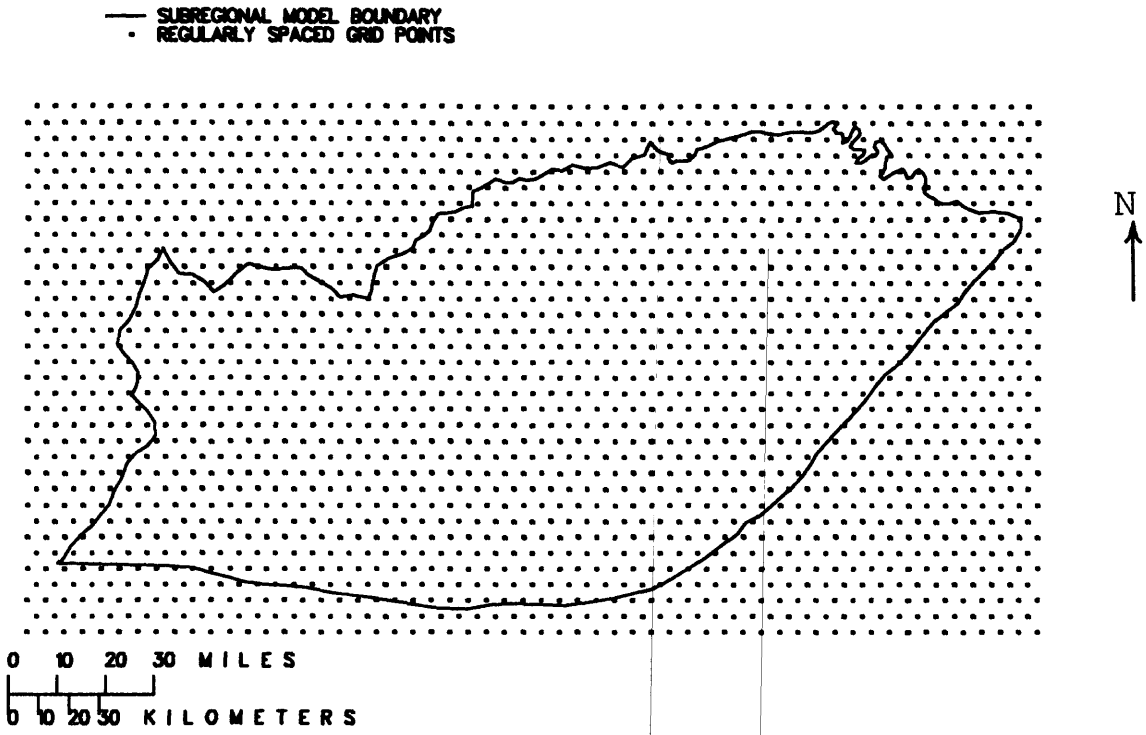


Figure 17.--Sample output from TRIANGLE.AML, an equilateral triangular grid.

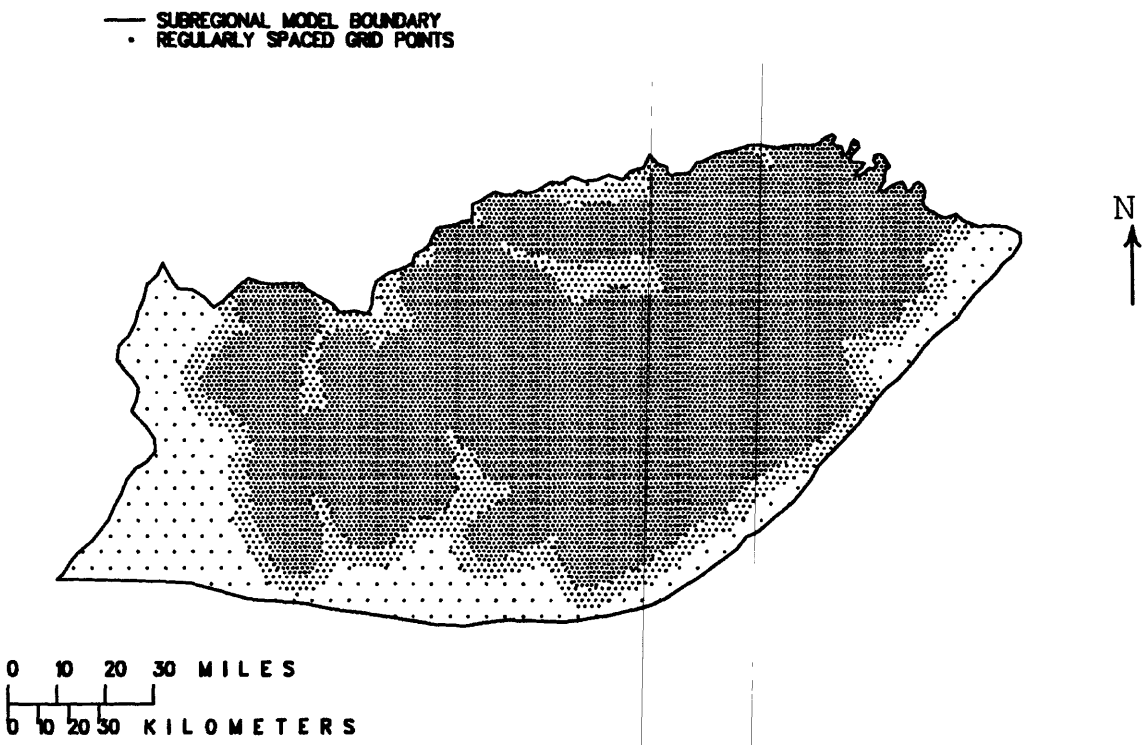


Figure 18.--The composite regularly spaced grid for the model.

Part Three. Generate a Mesh

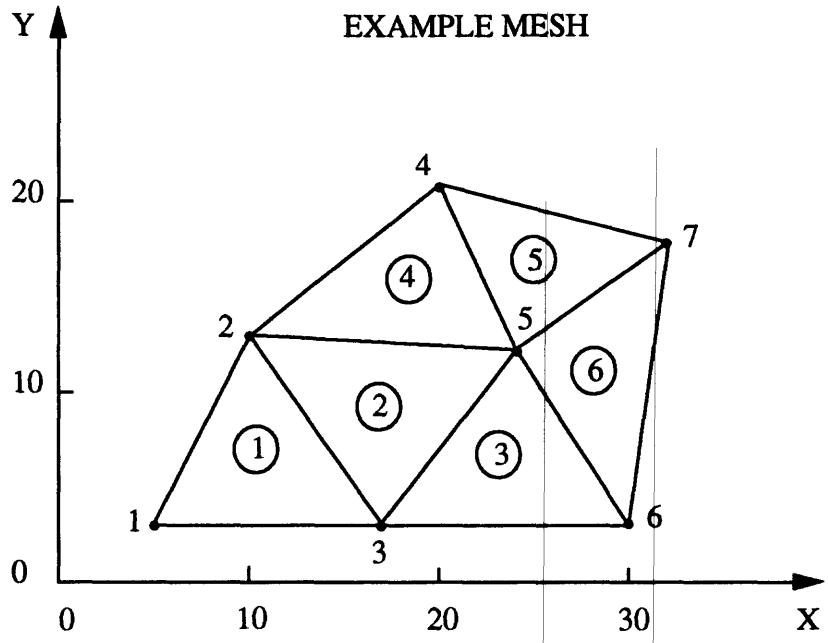
The tenth step is to run KITSINK.AML. KITSINK.AML completes the mesh generation process from this point. It creates the node coordinate data file, FILENCD, and the element connection data file, FILEECD, needed for modeling. Samples of node coordinate data and element connection data are shown in figure 19. KITSINK.AML creates a point cover containing all of the element label points, a point cover containing all of the element node points, and a polygon cover containing all of the triangular elements based on the feature and grid points. For the example, these coverages are called EXAMP.ELPT, EXAMP.NOD, and EXAMP.ELMS. Figures 20a and 20b show EXAMP.ELMS and EXAMP.ELPT, respectively. KITSINK.AML performs an optimization routine for node numbering, and therefore needs a user-imposed maximum number of optimization runs. A typical maximum would be 10 attempts.

KITSINK.AML performs a large number of operations, and requires a correspondingly long time (anywhere from five minutes to 10 hours). For most of this time, it may be left unattended, but one particular process requires user input. After its first operation, converting each input cover to a point cover and combining these into a master feature point cover, KITSINK.AML does several *SNAP* operations on this master cover. Because of difficulties accessing internal ARC/INFO command information, however, these operations require the user to answer a question, and hence the user must be present until the question can be answered "y" instead of "n." We hope to solve this problem in the future, but for now, a more detailed description of the problem is included in the description of SNAPPY.AML. Fortunately, this process is early in the KITSINK.AML procedure and therefore leaves the user free for most of KITSINK.AML's extensive run time.

For the example mesh, the commands required for this step are:

&R KITSINK 4207 MSTFEAT 500000 8000 MSTGRD ALLPTS MODBND INMOD 8000
MSTPTS MSTPOL 10 (with the following entries)
FAUCL.CL
LINE
FAULT
BASUB.CL
LINE
DRAINDIV
SUBGLIM.CL
LINE
SBGEOLIM
IRRPUM.CL
POINT
IRRIG
MUINPUM.CL
POINT
MUNI_IND
SPRINGS.CL
POINT
SPRING
STR.CL
LINE
STREAM
TREDAQ.CL
LINE
SUBREGN
OCRCL.CL
LINE
OUTCROP
DEVTR.CL

LINE
DVLRVR
STR.LEN



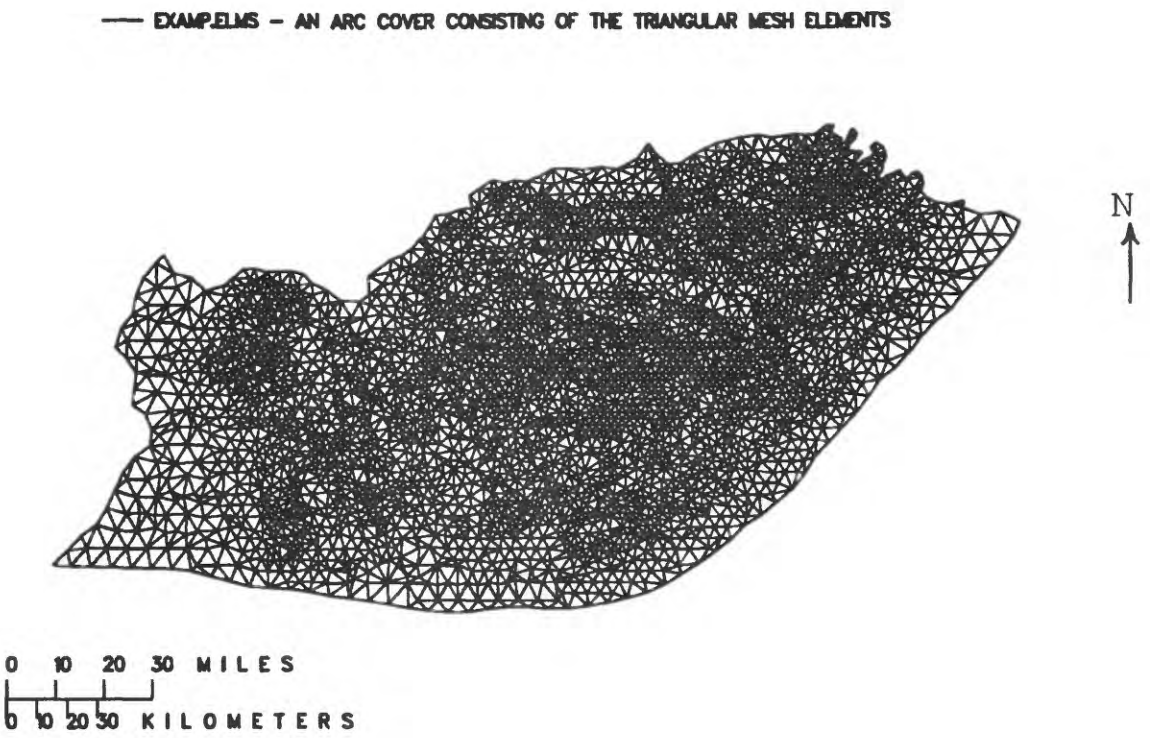
a. node coordinate data

node number	x coordinate	y coordinate
1	5	3
2	10	13
3	17	3
4	20	21
5	24	12
6	30	3
7	32	18

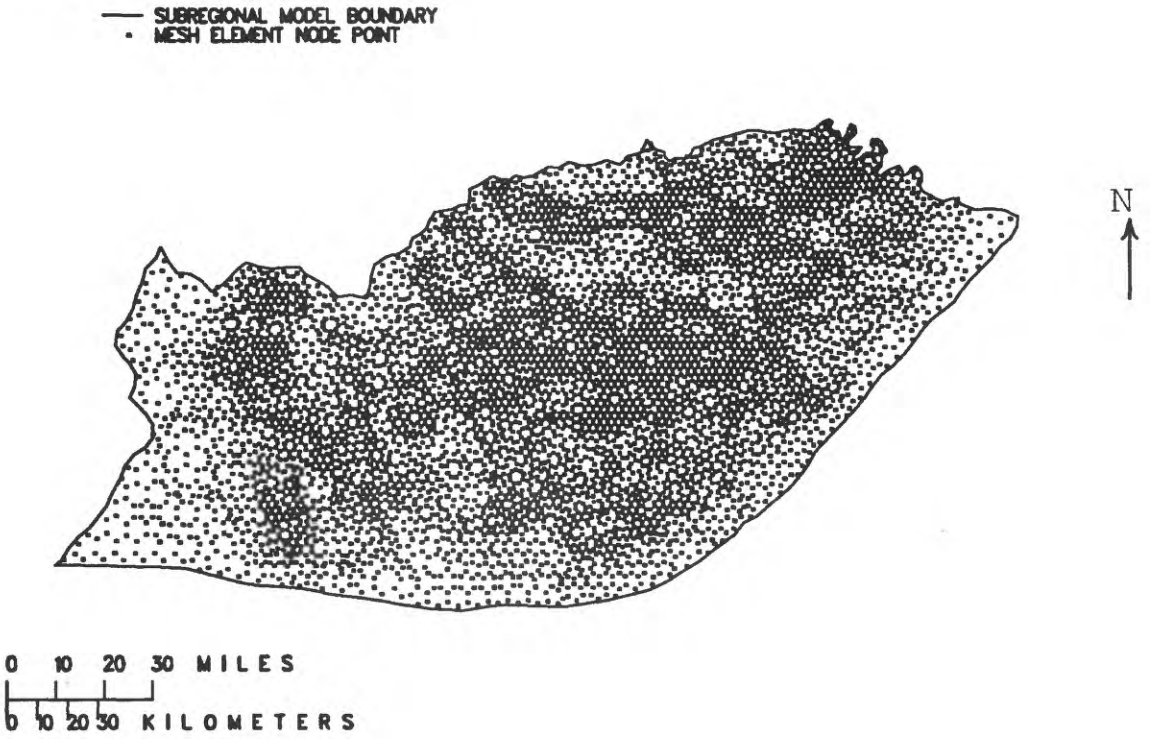
b. element connection data

element number	node connections i	j	k
1	1	3	2
2	2	3	5
3	3	6	5
4	4	2	5
5	4	5	7
6	7	5	6

Figure 19.--Example finite-element mesh, node coordinate data, and element connection data (from Kuniansky,1990, fig.1).



a. Finite-element mesh



b. Element centroids

Figure 20.--Output finite-element mesh and centroids of each element.

After KITSINK.AML, an optional eleventh step may be run on the mesh. This AML program assigns altitudes to certain feature-based points in the ".NOD" output cover. The altitudes are based upon the intersections of the selected feature with topographic contour lines. For example, altitudes can be assigned to all points in the ".NOD" file that represent stream vertices based upon the intersections of the streams with topographic contour lines.

This AML program is ELEVATE.AML. ELEVATE.AML requires as input a point cover that contains points coincident with the lines of a particular line cover. The points represent the intersections of the lines in the line cover with the equal-altitude lines of a topographic map. Some item in the PAT of this point cover should equal the topographic altitudes at the points of intersection. An example cover, TOPO.ELEV, is shown in figure 21. These points and elevations were digitized from a 1:24,000 topographic map. ELEVATE.AML also requires a point cover, some of whose points need this altitude information. The points needing altitude values (usually stream points) must have an identifying item set equal to one. ELEVATE.AML will interpolate between the given altitudes to produce altitudes for each point for which the identifying item is equal to one. The points with items equal to one should approximate the lines that were originally used to create the topographic intersections cover to ensure functioning of the AML program.

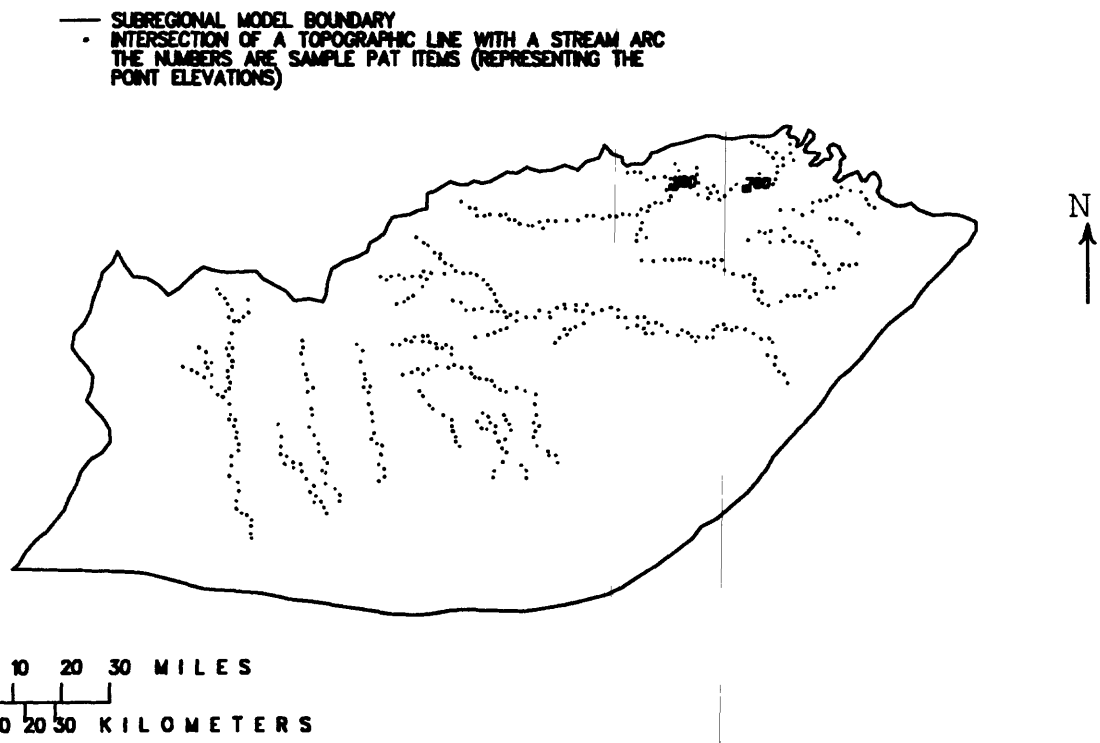


Figure 21.--Intersections of topographic elevation lines and the major streams in the study area.

For the example mesh, the command for this process is:

&R ELEVATE 4207 MSTPOL.NOD STREAM ELEV TOPO.ELEV ALT

IMPLEMENTATION OF MESH GENERATION (AML) PROGRAMS

The mesh generation programs are a series of ARC commands and ARC Macro Language routines, or macros, some of which call Fortran77 programs, organized into a procedural format. The AML programs are designed to be useful for other processes in addition to finite-element mesh generation. For example, ARCPOTIN.AML simplifies the use of any *TIN* command. The mesh generation process requires the user to run many macros separately, making decisions at each of the 10 steps. For these reasons, the macros are each described separately, as stand-alone modules. Some of the macros are used repeatedly; others are used only once. They are presented alphabetically in the section entitled "AML Programs" for ease of reference.

The macros used in the mesh generation procedure described in the section entitled "Overview of the mesh design process with example" are explained within that section. Because each module is composed of, or at least called by, an AML program, the macros are presented as main subsections in the section entitled "AML Programs." Any Fortran77 programs called by a macro are listed in the order in which they are called, as subsections of these main subsections. The exception to this format is MAKEADDL.AML. This macro is presented as a subsection of the REALENGTH.AML subsection because it is a rudimentary macro designed to create the INFO program ADD.LENGTHS. As such, it has no use outside of REALENGTH.AML.

The Fortran programs on the enclosed diskette have not been compiled. They must be compiled for use on whatever kind of computer is to be used. All programs were originally written and tested on a PRIME minicomputer. All code was written in American National Standards Institute, 1978, FORTRAN 77 where possible. Despite these efforts, however, individual differences in compilers may make some of these programs non-compilable. If this happens, the necessary changes should be minor.

When linking the programs, three libraries should be included. They are the standard FORTRAN library, VAPPLB if on a Prime minicomputer (or the appropriate system-specific library), and ARCLIB (for ARC interface commands). More information about compiling is included in the Supplemental Data section. One example of use of the ARCLIB library is the set of calls to AENTER, LUNINI, MINIT, and MESINI found at the beginning of some of the FORTRAN programs. These routines are necessary to interface with ARC/INFO and are found in ARCLIB. After making any necessary minor changes to system-specific commands and linking with the aforementioned libraries, the programs should run on any system.

AML programs must be run from within the ARC/INFO system, specifically at the "Arc" prompt and with the "&run" command. In order to reduce screen clutter, each AML program begins with the command "&echo &off." This command prevents the AML program commands from being printed to the screen. The computer system's responses to these commands will, however, be displayed. They are allowed to display so that the user can monitor the progress of the macro, which is particularly useful during macros that require extensive operating time.

These responses may be suppressed, on the Prime, by the insertion of the following command immediately after "&echo &off" in each AML program: "&sys como -ntty." One word of caution about the use of this command, however: Each macro has a rudimentary input error-checking routine. These routines display an error message explaining the correct usage of each macro. Other run-time error messages will cause a macro to abnormally abort. The macro will display a standard error message, but the source of the error will be displayed in the system error message that precedes the macro message. Use of the above command may cause suppression of this system error message, making errors more difficult to locate and correct.

One final note: If an AML program is run without any arguments, then it will respond with a "USAGE" statement, which tells the user which arguments need to be included. Arguments that are specified as "created" will be created by the macro. Arguments specified as "existing" should already exist before the macro is called.

AML PROGRAMS

ARCPOTIN.AML

Description

At least two of the ARC commands used by the mesh generation program, specifically *ARCPOINT* and *ARCTIN*, are more precisely *TIN* commands. *TIN* commands require that the input cover attribute table include the item "SPOT." The addition, and subsequent removal, of this item constitutes the primary purpose of the macro ARCPOTIN.AML (fig. 22). This macro adds the item "SPOT" to the input cover's attribute table, executes the specified command, and then removes "SPOT" from the input (and, if the command is "ARCPOINT," the output) coverages' attribute tables in order to maintain their ease of comprehension.

In addition, ARCPOTIN.AML can perform an optional secondary function. If the input command is '*ARCTIN*' and a polygon name is specified, ARCPOTIN.AML will take the triangular irregular network (TIN) that is created by *ARCTIN* and create a polygon cover based upon it. It creates this cover as a line cover, then *BUILD*s the line cover as a polygon cover. This feature simplifies the process of creating a triangular mesh polygon cover based on a point cover. If the secondary function is not desired, it may be bypassed by not specifying a polygon name. As do all of the macros, ARCPOTIN.AML performs a rudimentary check of the input data to ensure that all required inputs are present.

At the beginning of the AML program, any file with the name specified as the output cover name or the output polygon cover name will be *KILL*ed. Common errors include misspecifying the input cover type or using an input cover type that is inappropriate for the *TIN* command chosen.

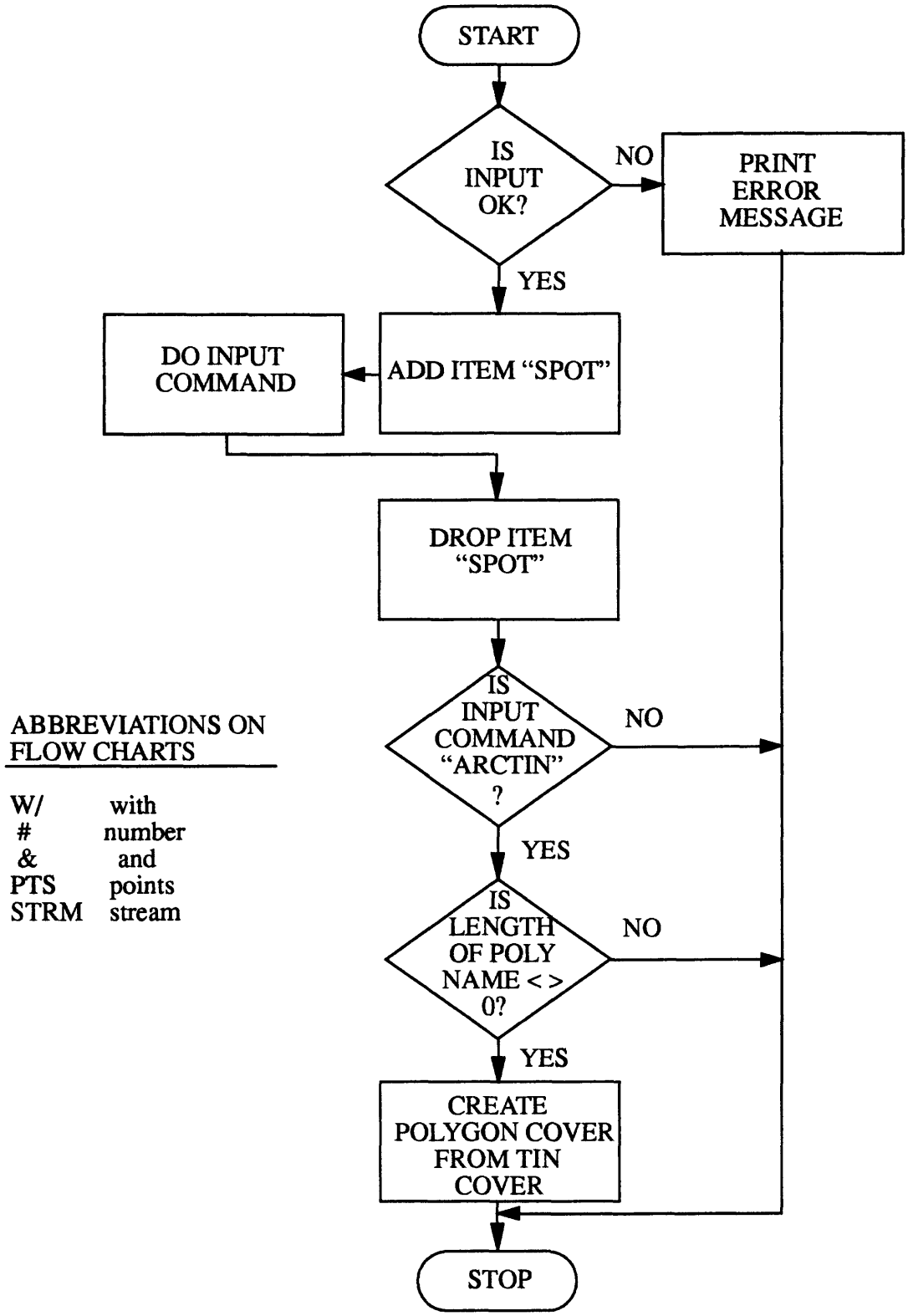


Figure 22.--Flowchart for ARCPOTIN.AML

```
/* MACRO: Use the TIN commands, ARCPOINT or ARCTIN, automatically
/*              adding and deleting the item 'spot' as necessary. Optionally,
/*              if a TIN is created, BUILD it as a poly cover.
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*    COMM: The command to be executed
/*    INCOV: The input cover name
/*  OUTCOV: The output cover name
/*         TIP: The input file type
/*POLYNAM: The name of the polygon cover to be created from the tin
/* COMMAN: The capitalized command name
/*         TYP: The capitalized file type
/*    INTAB: The input file attribute table
/*   OUTAB: The output file attribute table

&echo &off
&args comm incov outcov tip polynam

/* -Prepare the error-indication file-C
&setvar i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 1]
&setvar i [close %filun%]

/* -Test to see if all arguments are present as expected-O
&if [type %comm%] ne 1 &then &goto badentry
&if [type %incov%] ne 1 &then &goto badentry
&if [type %outcov%] ne 1 &then &goto badentry
&if [type %tip%] ne 1 &then &goto badentry
&if [length %tip%] eq 0 &then &goto badentry
&if [type %polynam%] ne 1 &then &goto badentry

/* -Eliminate old occurances of the output files-M
&severity &error &ignore
kill %outcov% all
&if [length %polynam%] ne 0 &then kill %polynam% all
&severity &error &fail

/* -Capitalize inputs as necessary-E
&setvar comman [translate %comm%]
&setvar typ [translate %tip%]

/* -Add the item 'spot' as necessary-D
&if %typ% eq 'LINE' &then &setvar intab %incov%.AAT
&if %typ% ne 'LINE' &then &setvar intab %incov%.PAT
&severity &error &ignore
additem %intab% %intab% spot 4 4 i
&severity &error &fail

/* -Execute the command-Y
```

```
%comman% %incov% %outcov% %typ%

/* -Drop the item 'spot' as necesary-
dropitem %intab% %intab% spot
&if %comman% eq 'ARCTIN' &then &goto atinonly
&setvar outab %outcov%.pat
dropitem %outab% %outab% spot
&goto endit
&label atinonly
&if [length %polynam%] eq 0 &then &goto endit
tinarc %outcov% %polynam% line
build %polynam% poly
createlabels %polynam%
&goto endit

/* -Print the error message-
&label badentry
&type Usage:  ARCPOTIN <TIN command> <in_cover (existing)> <out_cover or
&type              out_tin (created)> <type of input cover (point,line)>
&type              {output polygon cover name if ARCTIN is used and a poly
&type              cover is to be created from the out_tin (created)}
&goto errend

&label endit
/* -Prepare the error-indication file-
&setvar i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]

&label errend
&type End of ARCPOTIN
```

BUFFNSHINE.AML

Description

BUFFNSHINE.AML is essentially an extension of the *BUFFER* command in ARC/INFO. It performs two distinct functions, either of which may be performed separately if desired (fig. 23).

The primary function of BUFFNSHINE.AML is to create *BUFFER*s of specified size around the input cover's features. In addition, a user-named identifying item is added to the PAT of the *BUFFER*ed output file. The output file name is simply the input file name with the extension ".BUF" added.

The secondary function of BUFFNSHINE.AML is to eliminate features from a second cover that fall within the *BUFFER*ed areas created by BUFFNSHINE's primary function. The output for this operation is written to a cover whose name is the input second cover name with the extension ".CL." Though technically a function unrelated to *BUFFER*ing, deletion of this sort is often the rationale behind *BUFFER*ing and so this function is included in BUFFNSHINE.AML.

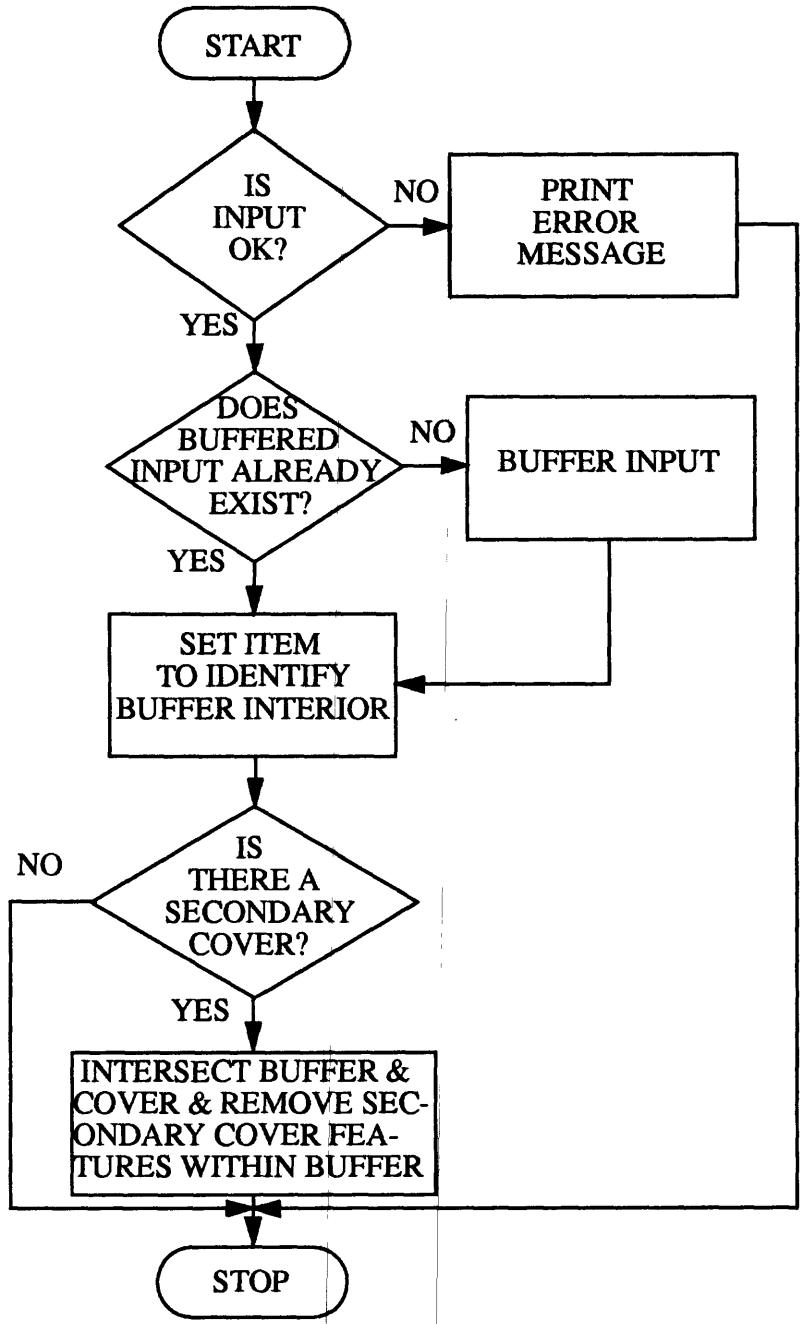


Figure 23.--Flowchart for BUFFNSHINE.AML.

If no feature removal is desired, then *BUFFER*ing can be performed alone by not giving BUFFNSHINE.AML any deletion cover name, deletion cover type, or display type. Similarly, deletion may be performed alone, based on a *BUFFER*ed cover that has been created by the macro on a previous run. This is accomplished by entering a cover name for which a ".BUF" file already exists. This combination of available operations is designed to make BUFFNSHINE.AML as flexible as possible.

If a second cover is specified, one whose features are to be removed, then any file under the name specified as the second cover and with the extension ".CL" will be *KILL*ed. Common errors include not *SPLINE*ing at a great enough distance for the buffer size chosen, and not having the second cover built (*BUILD*) as the type specified.

Program Listing

```
/* MACRO: Create buffer regions around features in a cover and
/*            assign an item with a value of 1 to indicate areas inside buffers.
/*            Also, optionally, delete another cover's features which fall
/*            within these buffers, or buffers created by an earlier run of
/*            this program.

/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky
/* VARIABLE LIST

/*             CV: The input cover
/*            TIP: The feature type of the input cover
/*         BUFDIS: The buffering distance
/*             IT: The item denoting the input cover
/*   OFFENSIVE: The cover whose features which lie within the buffers
/*                  are to be removed
/*           OTIP: The feature type of the above cover
/*          DTYPE: The display type
/*            COV: The capitalized version of CV
/*            TYP: The capitalized version of TIP
/*           ITEM: The capitalized version of IT
/*           OTYP: The capitalized version of OTIP
/*YULBRYNNER: The buffered version of COV
/*       BUFTPAT: The attribute table for YULBRYNNER
/*       OFFCLIP: The clipped version of OFFENSIVE
/*        OFFTAB: The attribute table for OFFCLIP
/*       DUMMY1: One of the variables used to remove items from attribute tables
/*       DUMMY2: The other of the variables mentioned above

&echo &off
&args cv tip bufdis it offensive otip dtype

/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 2]
&setvar i [close %filun%]

/* -Check the computer type (by Leonard L. Orzol)-O
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
```

```
&else
 &do
  &s .computer_type unix
 &end

/* -Test to see if all arguments are present as expected-M
&if [type %cv%] ne 1 &then &goto badentry
&if [type %tip%] ne 1 &then &goto badentry
&if [type %bufdis%] >= 0 &then &goto badentry
&if [type %it%] ne 1 &then &goto badentry
&if [len %it%] eq 0 &then &goto badentry
&if [type %offensive%] ne 1 &then &goto badentry

/* -Translate variable names to capitals for use in ARC/INFO-E
&setvar cov [translate %cv%]
&setvar typ [translate %tip%]
&setvar item [translate %it%]
&setvar otyp [translate %otip%]

/* -Buffer the features in the coverages-D
&setvar yulbrynner %cov%.BUF
&if [exists %yulbrynner% -coverage] &then &goto nextsec
&if %typ% eq 'POLY' &then &goto nextsec
buffer %cov% %yulbrynner% # # %bufdis% 40 %typ%

/* -Use ARC/INFO to set the item indicating areas inside buffers-Y
&label nextsec
&severity &error &ignore
&setvar buftpat %yulbrynner%.PAT
additem %buftpat% %buftpat% %item% 4 4 i
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SEL %BUFTPAT%
 RESEL FOR INSIDE = 100
 CALC %ITEM% = 1
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %BUFTPAT%
RESEL FOR INSIDE = 100
CALC %ITEM% = 1
Q STOP
QUIT
&end
&end

/* -Check to see if removal is desired-
```

```
&if [len %offensive%] eq 0 &then &goto endit

/* -If so, IDENTITY the coverages-H
&setvar offclip %offensive%.CL
&if [exists %offclip% -coverage] &then kill %offclip% all
identity %offensive% %yulbrynner% %offclip% %otyp% 40

/* -Use ARCEDIT to remove the desired areas-O
ae
mape %offclip%
disp %dtype%
editc %offclip%
&if %otyp% = 'POINT' &then
 &do
  editf label
  drawe label
 &end
&else
 &do
  editf arc
  drawe arc
 &end
draw
&severity &error &ignore
select screen
resel for %item% = 1
delete
&severity &error &fail
save
q

/* -Clean up the files-U
&if %otyp% = 'LINE' &then &setvar offtab %offclip%.AAT
&if %otyp% ne 'LINE' &then &setvar offtab %offclip%.PAT
dropitem %offtab% %offtab% %item%
&setvar dummy1 %offensive%#
&setvar dummy2 %offensive%-ID
dropitem %offtab% %offtab% %dummy1%
dropitem %offtab% %offtab% %dummy2%
dropitem %offtab% %offtab% INSIDE
&goto endit

/* -Print the error message-R
&Label badentry
&type Usage:  BUFFNSHINE <name of cover to be buffered (existing)> <type of
&type                cover (line,point,poly)> <buffer distance> <name of
&type                item to be added to output poly cover to indicate
&type                areas inside the polygon (created)> {cover whose
&type                features lying within the output polygon are to be
&type                removed (existing)} {cover type (line,point,poly)}
&type               {display type}
&goto errend

&label endit
```

```
/* -Prepare the error-indication file-
&s i[delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]

&label errend
&type End of BUFFNSHINE
```

CHICPOX.AML

Description

This macro creates a point cover based on an input line cover and ensures a perfect match between the vertices of the line cover and the points of the point cover. CHICPOX.AML first calls ARCPOTIN.AML to run the command *ARCPOINT* to create the point cover (fig. 24). It then *SNAP*s the arcs in the line cover to the points in the point cover to ensure a perfect match. This matching function is included because in the mesh generation process, CHICPOX.AML creates the points representing the model boundary. These points need to duplicate exactly the model *CLIP*ping cover in order for *CLIP*ping to occur properly.

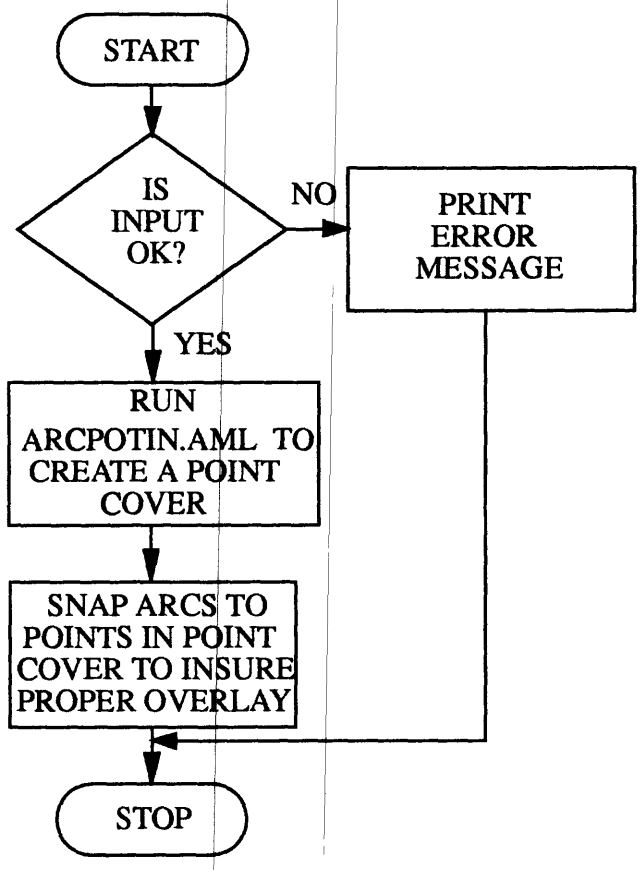


Figure 24.--Flowchart for CHICPOX.AML.

CHICPOX.AML *KILL*s any file with the name specified as the output point cover name. The most common error, especially when CHICPOX.AML is called by KITSINK.AML, is that the input line cover is not built (*BUILD*) both as a polygon and a line cover.

Program Listing

```
/* MACRO:  Take an input line (arc) cover and return an output
/*              point cover based upon the input. Additionally, snap the input
/*              arc cover vertices so that they match the point cover nodes
/* CODED BY:  Robert Lowther
/* SUPERVISED BY:  Eve L. Kuniansky

/* VARIABLE LIST
/*   DTYPE:  The display type
/*   COVIN:  The input polygon cover
/*     DIST:  The maximum distance across which arc vertices will be snapped
/*              to match the point cover nodes
/* COVOUT:  The output point cover

&echo &off
&args dtype covin dist covout
/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 3]
&setvar i [close %filun%]

/* -Test to see if all arguments are present as expected-O
&if [type %covin%] ne 1 &then &goto badentry
&if [type %dist%] ne -1 &then &goto badentry
&if [type %covout%] ne 1 &then &goto badentry
&if [length %covout%] eq 0 &then &goto badentry

/* -Eliminate old occurances of necessary files-M
&severity &error &ignore
kill %covout% all
&severity &error &fail

/* -Create the point cover-E
&run arcpotin arcpoint %covin% %covout% line
ae
mape %covin%
disp %dtype%
editc %covin%
drawe arc
snapc %covout%
backc %covout%
backe label
draw
editf arc
snapf arc label
snapping closest %dist%
select screen
```

```
snap
save
q
clean %covin%
&goto endit

/* -Print the error message-D
&label badentry
&type Usage:  CHICPOX <display type> <input line cover(existing)> <maximum
&type              distance output points might deviate from input arcs>
&type             <output point cover (created)>
&goto enderr
&label endit

/* -Prepare the error-indication file-Y
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]

&label enderr
&type End of CHICPOX
```

CLIPIT.AML

Description

This macro functions as an extended *CLIP* function (fig. 25). It will "*CLIP*" up to 10 coverages in a single run, based on one polygon "*CLIP*ping" cover. The output files have the same name as the input files, but with a ".CL" extension. Users must specify whether they want the parts of the input coverages inside or outside to remain. In addition, CLIPIT.AML can create a *BUFFER* region around the *CLIP*ping cover and remove any features from the input coverages that lie within this region. Although this secondary function has limited applications for line and polygon input coverages, it can be quite useful for point coverages. If this function is not needed, it can be bypassed by entering a value of zero for the *BUFFER*ing distance. If a buffer for the *CLIP*ping cover with the name "*CLIP*ping cover name.BUF" already exists, then that cover will be used for the point removal operation. Such a cover must already have an identifying item. If not, a cover will be created using the same naming convention.

CLIPIT.AML KILLs any files with a name designated as an input cover name with either a ".CL" extension or a "d" extension. Common errors include not having built (*BUILD*) the *CLIP*ping cover as a polygon, not having an identifying item for the *CLIP*ping cover, trying to *CLIP* coverages of different types in the same run of CLIPIT.AML, or misspecifying whether the part of an input cover inside or outside of the *CLIP*ping cover is to remain.

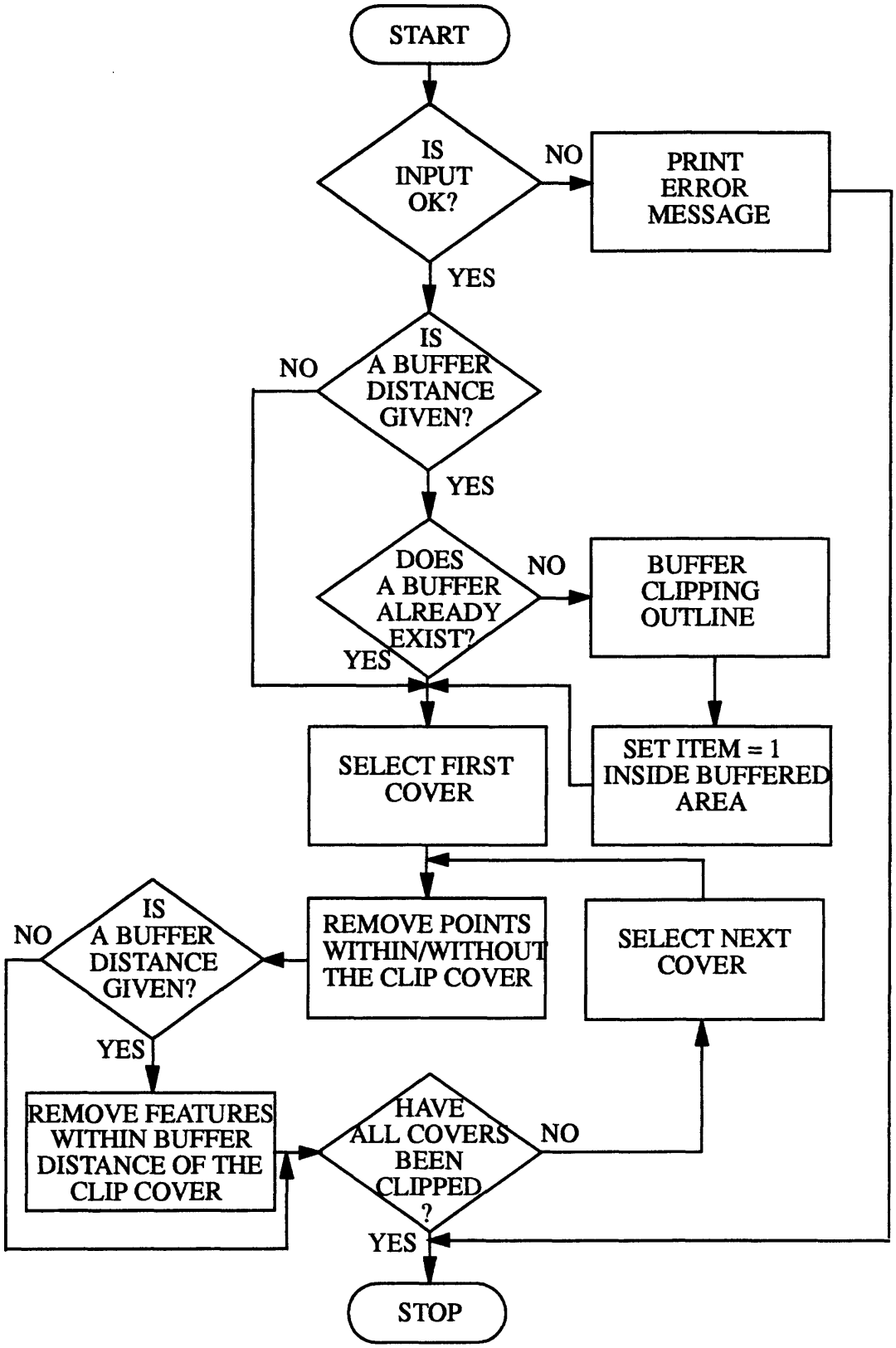


Figure 25.--Flowchart for CLIPIT.AML and CLIPIT2.AML.

```
/* MACRO: Clip up to ten coverages with a clipping cover and remove
/*              any features which are too close to the clip boundaries
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky
/* VARIABLE LIST:/*        DTYPE: The graphic display type being used
/*     CLIPPER: The cover being used as a cutting template for clipping
/*        ITEM: The item indicating the area within the cutting template
/*     CLIPDIS: The size of the buffer zone to be created around
/*                the clipping cover
/*         TYP: The cover type of the coverages to be clipped
/*COV1-COV10: The names of the coverages to be clipped
/*      W1-W10: The corresponding inside or outside clip for the above
/*   WANKEL: The item added to the buffered clip cover to indicate
/*              areas which are within the buffer distance
/*         COV: The name of the cover currently being considered
/*        WNOW: The inside/outside clip variable currently being considered
/*        COVD: The name of the output cover for the clipped version of cov
/*     CLBUFF: The buffered version of the clipping outline

&echo &off
&args dtype clipper item clipdis tip cov1 w1 cov2 w2 cov3 w3 cov4 w4 ~
  cov5 w5 cov6 w6 cov7 w7 cov8 w8 cov9 w9 cov10 w10

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
  &do
    &s .slash >
    &s .computer_type prime
  &end
&else
  &do
    &s .computer_type unix
  &end

/* -Test input to see if all arguments are present as expected-O
&if [type %clipper%] ne 1 &then &goto badentry
&if [type %item%] ne 1 &then &goto badentry
&if [type %clipdis%] >= 0 &then &goto badentry
&if [type %tip%] ne 1 &then &goto badentry
&if [type %cov1%] ne 1 &then &goto badentry
&if [type %w1%] ne 1 &then &goto badentry
&if [length %w1%] = 0 &then &goto badentry

&setvar typ [translate %tip%]
&if typ eq 'POLY' &then &goto badentry

/* -Create a buffer zone around the clipping cover-M
&if %clipdis% eq 0 &then &goto looop
&type 'Enter the identifying item name for the buffer area (Note: if the'
```

```
&setvar itmb [response 'buffer cover already exists, so must this item)']
&setvar itemb [translate %itmb%]
&setvar clapper [translate %clipper%]
&setvar clbuff %clapper%.BUF
&if [exists %clbuff% -COVERAGE] &then &goto buffered
buffer %clipper% %clbuff% # # %clipdis% 40 line
&label buffered
build %clbuff% poly
&setvar clbuffpat %clbuff%.PAT
&severity &error &ignore
additem %clbuffpat% %clbuffpat% %itemb% 4 4 i
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %clbuffpat%
RESEL FOR INSIDE = 100
CALC %itemb% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %clbuffpat%
RESEL FOR INSIDE = 100
CALC %itemb% = 1
Q STOP
QUIT
&end
&end

&label looop
&do cov &list %cov1% %cov2% %cov3% %cov4% %cov5% %cov6% ~
  %cov7% %cov8% %cov9% %cov10%
&if [length %cov%] eq 0 &then &goto endloop
&if %cov% eq %cov1% &then &setvar wnow %w1%
&if %cov% eq %cov2% &then &setvar wnow %w2%
&if %cov% eq %cov3% &then &setvar wnow %w3%
&if %cov% eq %cov4% &then &setvar wnow %w4%
&if %cov% eq %cov5% &then &setvar wnow %w5%
&if %cov% eq %cov6% &then &setvar wnow %w6%
&if %cov% eq %cov7% &then &setvar wnow %w7%
&if %cov% eq %cov8% &then &setvar wnow %w8%
&if %cov% eq %cov9% &then &setvar wnow %w9%
&if %cov% eq %cov10% &then &setvar wnow %w10%
&setvar covd %cov%d
&setvar sav %cov%.CL
&severity &error &ignore
kill %covd% all
kill %sav% all
&severity &error &fail
```

```
/*  -Clip the coverages-E

&severity &error &ignore
identity %cov% %clipper% %covd% %typ% 40
&severity &error &fail
ae
disp %dtype%
mape %covd%
editc %covd%
&if %typ% eq 'LINE' &then drawe arc
&if %typ% eq 'LINE' &then editf arc
&if %typ% eq 'POINT' &then drawe label
&if %typ% eq 'POINT' &then editf label
draw
select screen
resel for %item% eq 1
&if [length %wnow%] eq 2 &then nsel
&severity &error &ignore
delete
&severity &error &fail
save
q

&severity &error &ignore
&if %clipdis% ne 0 &then identity %covd% %clbuff% %sav% %typ% 40
&severity &error &fail
&if %clipdis% eq 0 &then copy %covd% %sav%
kill %covd% all

/*  -Remove any cover features which are too close to the clipping cover-D
&if %clipdis% eq 0 &then &goto lateloop
ae
disp %dtype%
mape %sav%
editc %sav%
&if %typ% eq 'LINE' &then drawe arc
&if %typ% eq 'LINE' &then editf arc
&if %typ% eq 'POINT' &then drawe label
&if %typ% eq 'POINT' &then editf label
draw
select screen
resel for %itemb% = 1
delete
save
q
&label lateloop
build %sav% %typ%
&if %typ% eq 'LINE' &then &setvar savtab %sav%.AAT
&if %typ% ne 'LINE' &then &setvar savtab %sav%.PAT
&if %typ% eq 'LINE' &then dropitem %savtab% %savtab% AREA
&if %typ% eq 'LINE' &then dropitem %savtab% %savtab% PERIMETER
&severity &error &ignore
dropitem %savtab% %savtab% %item%
```

```
&setvar dummy1 %cov%#
&setvar dummy2 %cov%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&setvar dummy1 %clipper%#
&setvar dummy2 %clipper%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&if %clipdis% eq 0 &then &goto endloop
dropitem %savtab% %savtab% INSIDE
dropitem %savtab% %savtab% %itemb%
&setvar dummy1 %covd%#
&setvar dummy2 %covd%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&setvar dummy1 %clbuff%#
&setvar dummy2 %clbuff%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&label endloop
&severity &error &fail
&end
&goto endit

/* -Print error message-Y
&label badentry
&type Usage: CLIPIT <display type> <clipping cover (existing)> <item denoting
&type        area inside the clipping cover(existing)> <distance from
&type        the edge within which features should be removed(or 0 if
&type        no removal desired) <type of coverages to be clipped (line,
&type        point)> <cover #1><in/out #1>....{cover #10} {in/out #10}

&label endit
&type End of CLIPIT
```

CLIPIT2.AML

Description

This macro is simply a slightly modified version of CLIPIT.AML designed to run from within KITSINK.AML. In this version, no screen prompt is displayed for the item designating areas too close to the *CLIP*ping border. Here the name is entered as an argument. CLIPIT2.AML follows the same flowchart as CLIPIT.AML, presented in figure 25.

Program Listing

```
/* MACRO: Clip up to ten coverages with a clipping cover and remove
/*        any features which are too close to the clip boundaries
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*      DTYPE: The graphic display type being used
```

```
/*      CLIPPER:  The cover being used as a cutting template for clipping
/*          ITEM:  The item indicating the area within the cutting template
/*       CLIPDIS:  The size of the buffer zone to be created around
/*                 the clipping cover
/*           TYP:  The cover type of the coverages to be clipped
/*COV1-COV10:  The names of the coverages to be clipped
/*       W1-W10:  The corresponding inside or outside clip for the above
/*     WANKEL:  The item added to the buffered clip cover to indicate
/*                areas which are within the buffer distance
/*          COV:  The name of the cover currently being considered
/*        WNOW:  The inside/outside clip variable currently being considered
/*         COVD:  The name of the output cover for the clipped version of cov
/*      CLBUFF:  The buffered version of the clipping outline

&echo &off
&args itmb dtype clipper item clipdis tip cov1 w1 cov2 w2 cov3 w3 cov4 w4 ~
  cov5 w5 cov6 w6 cov7 w7 cov8 w8 cov9 w9 cov10 w10

/*  -Prepare the error-indication file-C
&setvar i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 4]
&setvar i [close %filun%]

/*  -Check the computer type (by Leonard L. Orzol)-O
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
  &do
    &s .slash >
    &s .computer_type prime
  &end
&else
  &do
    &s .computer_type unix
  &end

/*  -Test input to see if all arguments are present as expected-M
&if [type %clipper%] ne 1 &then &goto badentry
&if [type %item%] ne 1 &then &goto badentry
&if [type %clipdis%] >= 0 &then &goto badentry
&if [type %tip%] ne 1 &then &goto badentry
&if [type %cov1%] ne 1 &then &goto badentry
&if [type %w1%] ne 1 &then &goto badentry
&if [length %w1%] = 0 &then &goto badentry

&setvar typ [translate %tip%]
&if typ eq 'POLY' &then &goto badentry

/*  -Create a buffer zone around the clipping cover-E
&if %clipdis% eq 0 &then &goto looop
&setvar itemb [translate %itmb%]
&setvar clapper [translate %clipper%]
```

```
&setvar clbuff %clapper%.BUF
&if [exists %clbuff% -COVERAGE] &then &goto buffered
buffer %clipper% %clbuff% # # %clipdis% 40 line
&label buffered
build %clbuff% poly
&setvar clbuffpat %clbuff%.PAT
&severity &error &ignore
additem %clbuffpat% %clbuffpat% %itemb% 4 4 i
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %clbuffpat%
RESEL FOR INSIDE = 100
CALC %itemb% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %clbuffpat%
RESEL FOR INSIDE = 100
CALC %itemb% = 1
Q STOP
QUIT
&end
&end

&label looop
&do cov &list %cov1% %cov2% %cov3% %cov4% %cov5% %cov6% ~
  %cov7% %cov8% %cov9% %cov10%
&if [length %cov%] eq 0 &then &goto endloop
&if %cov% eq %cov1% &then &setvar wnow %w1%
&if %cov% eq %cov2% &then &setvar wnow %w2%
&if %cov% eq %cov3% &then &setvar wnow %w3%
&if %cov% eq %cov4% &then &setvar wnow %w4%
&if %cov% eq %cov5% &then &setvar wnow %w5%
&if %cov% eq %cov6% &then &setvar wnow %w6%
&if %cov% eq %cov7% &then &setvar wnow %w7%
&if %cov% eq %cov8% &then &setvar wnow %w8%
&if %cov% eq %cov9% &then &setvar wnow %w9%
&if %cov% eq %cov10% &then &setvar wnow %w10%
&setvar covd %cov%d
&setvar sav %cov%.CL
&severity &error &ignore
kill %covd% all
kill %sav% all
&severity &error &fail

/*  -Clip the coverages-D
```

```
&severity &error &ignore
identity %cov% %clipper% %covd% %typ% 40
&severity &error &fail
ae
disp %dtype%
mape %covd%
editc %covd%
&if %typ% eq 'LINE' &then drawe arc
&if %typ% eq 'LINE' &then editf arc
&if %typ% eq 'POINT' &then drawe label
&if %typ% eq 'POINT' &then editf label
draw
select screen
resel for %item% eq 1
&if [length %wnow%] eq 2 &then nsel
&severity &error &ignore
delete
&severity &error &fail
save
q

&severity &error &ignore
&if %clipdis% ne 0 &then identity %covd% %clbuff% %sav% %typ% 40
&severity &error &fail
&if %clipdis% eq 0 &then copy %covd% %sav%
kill %covd% all

/*  -Remove any cover features which are too close to the clipping cover-Y
&if %clipdis% eq 0 &then &goto lateloop
ae
disp %dtype%
mape %sav%
editc %sav%
&if %typ% eq 'LINE' &then drawe arc
&if %typ% eq 'LINE' &then editf arc
&if %typ% eq 'POINT' &then drawe label
&if %typ% eq 'POINT' &then editf label
draw
select screen
resel for %itemb% = 1
delete
save
q
&label lateloop
build %sav% %typ%
&if %typ% eq 'LINE' &then &setvar savtab %sav%.AAT
&if %typ% ne 'LINE' &then &setvar savtab %sav%.PAT
&if %typ% eq 'LINE' &then dropitem %savtab% %savtab% AREA
&if %typ% eq 'LINE' &then dropitem %savtab% %savtab% PERIMETER
&severity &error &ignore
dropitem %savtab% %savtab% %item%
&setvar dummy1 %cov%#
&setvar dummy2 %cov%-ID
dropitem %savtab% %savtab% %dummy1%
```

```
dropitem %savtab% %savtab% %dummy2%
&setvar dummy1 %clipper%#
&setvar dummy2 %clipper%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&if %clipdis% eq 0 &then &goto endloop
dropitem %savtab% %savtab% INSIDE
dropitem %savtab% %savtab% %itemb%
&setvar dummy1 %covd%#
&setvar dummy2 %covd%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&setvar dummy1 %clbuff%#
&setvar dummy2 %clbuff%-ID
dropitem %savtab% %savtab% %dummy1%
dropitem %savtab% %savtab% %dummy2%
&severity &error &fail
&label endloop
&end
&goto endit

/* -Print error message-
&label badentry
&type Usage: CLIPIT2 <item denoting area within buffer> <display type>
&type        <clipping cover> <item denoting area>
&type        <buffering distance> <type of cover (line,point)>
&type        <cover #1> <in/out #1>....{cover #10} {in/out #10}
&goto enderr

&label endit
/* -Prepare the error-indication file-
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]
&label enderr
&type End of CLIPIT2
```

ELEVATE.AML

Description

ELEVATE.AML assigns elevations to points within a point cover, presumably but not necessarily the stream points within a point cover composed of many different feature points. As input and output, ELEVATE.AML requires very specialized coverages. It requires as input a point cover whose points are defined to be the intersections of the stream (or other feature) arcs representing the points to be elevated with the topographic contour lines of the region in question. As output, it requires a point cover with an item in its PAT that is set equal to "1" for each point that is to have an elevation assigned. The flowchart for ELEVATE.AML is shown in figure 26. The lines that are used to determine the topographic line intersection points should be the same lines that are used to generate the points in the output point cover. The elevation of each of the topographic intersection points should be recorded as an item value for each of those points. The elevations for the output points that do not fall on topographic lines are interpolated from the two nearest topographic intersection points.

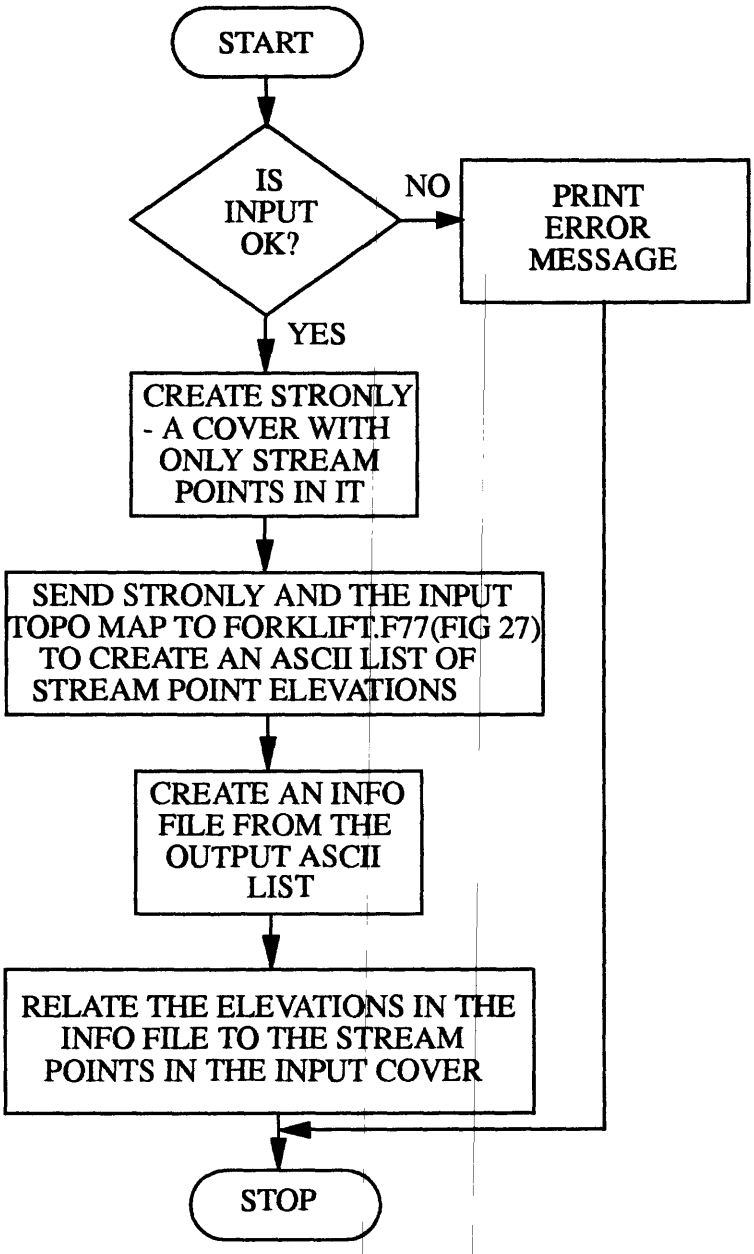


Figure 26.--Flowchart for ELEVATE.AML.

The files, "STRPTS," "ELEVPTS," "STRELEVS," and the cover "STRONLY" are deleted or *KILL*ed by ELEVATE.AML at the beginning of its run. Common errors include not having an identifying item in the output cover to indicate which points are to be elevated, not using a point cover as the output cover, not having an item in the topographic intersection cover (TI cover) representing the elevation, or not using the same arc cover to generate, in the TI cover, the intersection points with the topographic contour lines and to generate the points indicated by the identifying item in the output cover.

```
/* MACRO:  Add river elevations to a point cover (with an item identifying
/*                the river points) based on a point cover of intersections of
/*                topographic map contour lines with the rivers
/* CODED BY:  Robert Lowther
/* SUPERVISED BY:  Eve L. Kuniansky

/* VARIABLE LIST

/*      DTYPE:  The screen graphic display device type
/*      COVER:  The point cover containing points to be elevated
/*   STRWORD:  The item which is equal to one for each point to be elevated
/*   OUTELEV:  The output cover item to contain the elevation info
/*        TOPO:  The cover with points at the intersections of topographiccontour
/*                  lines and the lines along which points to be elevated lie
/*     INELEV:  The input cover item to contain the elevation info
/*  COVERPAT:  COVER, capitalized, with the extension, ".PAT"
/*    COVERID:  COVER, capitalized, with the extension, "-ID"
/* WHEREOUT:  The name of the output cover, complete with its entire path

&echo &off
&args dtype cover strword outelev topo inelev

/* -Check the computer type (by Leonard I. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test input to see if all arguments are present as expected-O
&if [type %dtype%] ne -1 &then &goto badentry
&if [type %cover%] ne 1 &then &goto badentry
&if [type %strword%] ne 1 &then &goto badentry
&if [type %outelev%] ne 1 &then &goto badentry
&if [type %topo%] ne 1 &then &goto badentry
&if [type %inelev%] ne 1 &then &goto badentry

&if [length %inelev%] eq 0 &then &goto badentry

/* -Define some variables-M
&setvar coverpat [translate %cover%].PAT
&setvar coverid [translate %cover%]-ID
&setvar topopat [translate %topo%].PAT
&setvar topoid [translate %topo%]-ID
&setvar whereout [pathname STRELEVS]
```

```
&setvar outel [translate %outelev%]
&setvar inel [translate %inelev%]

/* -Delete any old occurances of files-E
&s i [delete strpts]
&s i [delete elevpts]
&s i [delete strelevs]
&severity &error &ignore
kill stronly all
&severity &error &fail

/* -Prepare the input coverages to dump data onto the FORKLIFT-D
additem %topopat% %topopat% temp 4 5 b
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SEL %topopat%
 CALC TEMP = %topoid%
 CALC %topoid% = %inel%
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %topopat%
CALC TEMP = %topoid%
CALC %topoid% = %inel%
Q STOP
QUIT
&end
&end
idedit %topo% point
copy %cover% stronly
ae
 disp %dtype%
 mape stronly
 editc stronly
 editf label
 drawe label
 draw
 select screen
 resel for %strword% = 1
 nsel
&SEVERITY &ERROR &IGNORE
 delete
&SEVERITY &ERROR &FAIL
 save
q
&severity &error &ignore
additem %coverpat% %coverpat% %outel% 7 7 i
additem %coverpat% %coverpat% dumrelat 7 7 i
```

```
&severity &error &fail
ungenerate point stronly strpts
ungenerate point %topo% elevpts

/* -Use FORKLIFT to interpolate between TOPO points and add the appropriate
/*  elevation to each point in STRONLY-Y
&if .computer_type = 'prime' &then &sys r forklift
&else &sys forklift.out

/* -Create WADE, an info file to hold the stream point elevation data-H
&if %.computer_type% = 'prime' &then
&do
&Data ARC INFO
 SELECT WADE
 PURGE
 Y
 ERASE WADE
 Y
 DEFINE WADE
 DUMRELAT,7,7,I
 %outel%,7,7,I

 GET %whereout% COPY
 SELECT %coverpat%
 CALC DUMRELAT = %coverid%
 SELECT WADE
 RELATE %coverpat% BY DUMRELAT
 CALC $1%outel% = %outel%
 Q STOP
&end
&end
&else
&do
&Data ARC
INFO
ARC
SELECT WADE
PURGE
Y
ERASE WADE
Y
DEFINE WADE
DUMRELAT,7,7,I
%outel%,7,7,I

GET %whereout% COPY ASCII
SELECT %coverpat%
CALC DUMRELAT = %coverid%
SELECT WADE
RELATE %coverpat% BY DUMRELAT
CALC $1%outel% = %outel%
Q STOP
QUIT
&end
```

```
&end

/* -Restore the input cover-O
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SEL %topopat%
 CALC %topoid% = TEMP
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %topopat%
CALC %topoid% = TEMP
Q STOP
QUIT
&end
&end
dropitem %topopat% %topopat% temp

/* -Eliminate unecessary coverages-U
dropitem %coverpat% %coverpat% DUMRELAT
&s i [delete strpts]
&s i [delete elevpts]
&s i [delete strelevs]
kill stronly all
&goto endit

/* -Print the error message-R
&label badentry
&type Usage: ELEVATE <display type> <output cover to be elevated (existing)>
&type          <output item indicating points to be elevated(existing)>
&type          <output cover item to contain elevation (created)><input
&type           cover w/elevation points near the points to be elevated
&type           (existing)><input item containing elevation (existing)>
&label endit
&type End of ELEVATE
```

Fortran Program FORKLIFT.F77

Description

FORKLIFT.F77 performs the actual interpolation of elevations for ELEVATE.AML. It examines each relevant point in the input cover in turn (fig. 27). First, it selects the two nearest topographic line intersection points for each output point. After selection, it examines their positions and the position of the output point to determine relative distances between the three points. If the output point lies between the two closest points, then its elevation is calculated as a linear interpolation between the two topographic line intersection points. If, however, the output point lies to one side of the two nearest points, then its elevation is defined to be equal to the elevation of the nearest point.

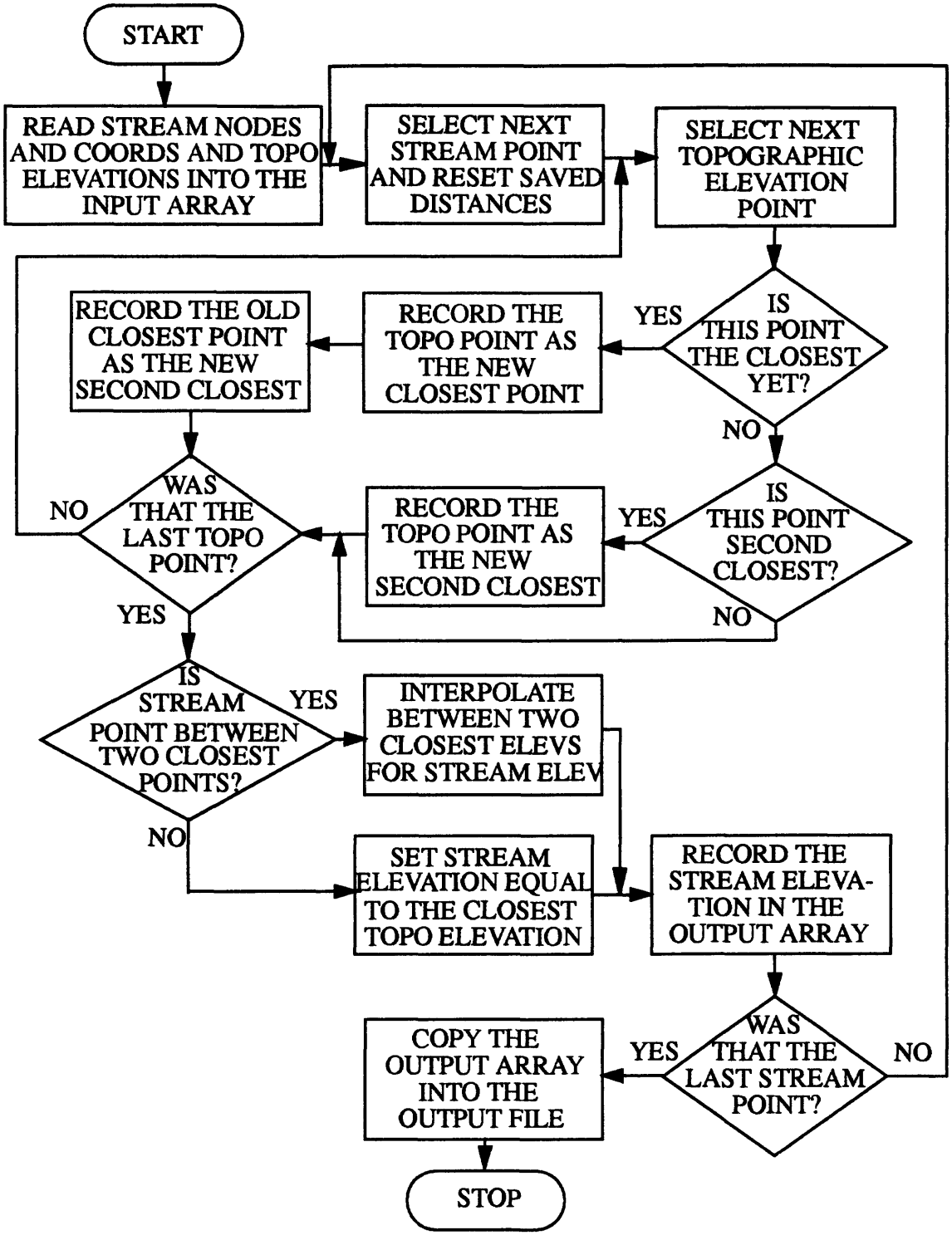


Figure 27.--Flowchart for FORKLIFT.F77.

Program listing

```
C*************************************************************************C
C      PROGRAM: Interpolate elevations to stream points                   C
C      CODED BY: Robert Lowther                                           C
C      SUPERVISED BY: Eve L. Kuniansky                                    C
C*************************************************************************C

      PROGRAM FORKLIFT

C*************************************************************************C
C   VARIABLE LIST                                                         C
C                                                                         O
C        ARGS: THE DUMMY INPUT AND OUPUT ARGUMENT                         C
C           I,J: COUNTER VARIABLES                                        M
C      CNTSTR: THE NUMBER OF STREAM POINTS                                C
C     STRNODE: THE NODE NUMBER OF THE INPUT STREAM POINTS                 C
C     STRDATA: THE COORDINATES OF THE INPUT STREAM POINTS                 H
C     CNTELEV: THE NUMBER OF STREAM/TOPO LINE INTERSECTIONS               C
C     ELEVATE: THE ELEVATION OF THE INPUT STREAM/TOPO LINE                R
C                 INTERSECTION POINTS                                     C
C   ELEVDATA: THE COORDINATES OF THE NODES ABOVE                          C
C       CLOSE1: THE CLOSEST DISTANCE BETWEEN A GIVEN STREAM POINT         S
C               AND ANY STREAM/TOPO LINE INTERSECTION                     C
C       CLOSE2: THE SAME AS ABOVE, BUT THE SECOND CLOSEST                 O
C          CL1J: THE INPUT ARRAY POSN OF THE PT AT DISTANCE, CLOSE1       C
C          CL2J: THE INPUT ARRAY POSN OF THE PT AT DISTANCE, CLOSE2       F
C          DIST: THE DISTANCE BETWEEN A GIVEN STREAM POINT AND A          C
C                STREAM/TOPO LINE INTERSECTION, USED TO COMPARE           T
C                AGAINST AND POSSIBLY REPLACE CL1J OR CL2J                C
C        DISTP1: THE DISTANCE FROM A GIVEN STREAM POINT TO CL1J           W
C        DISTP2: THE DISTANCE FROM A GIVEN STREAM POINT TO CL2J           C
C        DIST12: THE DISTANCE FROM CL1J TO CL2J                           A
C        PELEV: THE STREAM POINT ELEVATION, INTERPOLATED FROM THE         C
C               CL1J AND CL2J ELEVATIONS                                  R
C       OUTAR: THE OUTPUT ARRAY WHICH CONTAINS THE STREAM NODE            C
C                NUMBER AND THE ELEVATION                                 E
C*************************************************************************C

      COMMON /C1/ELEVATE(25000),STRDATA(25000,2)
      COMMON /C2/ELEVDATA(25000,2),OUTAR(25000,2),STRNODE(25000)

      INTEGER I,J,CNTSTR,CNTELEV,STRNODE,ELEVATE
      INTEGER CL1J,CL2J,PELEV,OUTAR
      REAL*8 STRDATA,ELEVDATA,CLOSE1
      REAL*8 CLOSE2,DIST,DISTP1,DISTP2,DIST12
      CHARACTER*32 ARGS

100   FORMAT (I10,2F13.6)
110   FORMAT (2I7)

C=====Open the input and output files==========C

      OPEN (10,FILE= 'strelevs')
      OPEN (11,FILE= 'strpts')
```

```
      OPEN (12,FILE= 'elevpts')

C=====Load the input data into the input arrays========C

      I = 1
      CNTSTR = 0
10    READ (11,100,ERR= 20) STRNODE(I),STRDATA(I,1),
     C STRDATA(I,2)
       I = I + 1
       CNTSTR = CNTSTR + 1
      GO TO 10
20    I=1
      CNTELEV = 0
25    READ (12,100,ERR= 30) ELEVATE(I),ELEVDATA(I,1),ELEVDATA(I,2)
       I=I+1
       CNTELEV = CNTELEV + 1
      GO TO 25

C=====EXAMINE EACH STREAM POINT IN TURN=====C

30    DO 40 I=1,CNTSTR
       CLOSE1 = 999999999
       CLOSE2 = 999999999
       CL1J = 0
       CL2J = 0

C---FIND THE TWO ELEVATION POINTS CLOSEST TO THE GIVEN STREAM POINT----C
       DO 50 J=1,CNTELEV
        DIST = SQRT((ELEVDATA(J,1) - STRDATA(I,1))**2 +
     C  (ELEVDATA(J,2) - STRDATA(I,2))**2)
        IF (DIST .GE. CLOSE1) GO TO 33
         CLOSE2 = CLOSE1
         CLOSE1 = DIST
         CL2J = CL1J
         CL1J = J
         GO TO 50
33       IF (DIST .GE. CLOSE2) GO TO 50
         CLOSE2 = DIST
         CL2J = J
50     CONTINUE

C-CALCULATE THE DISTANCES BETWEEN THE STREAM AND THE 2 CHOSEN PTS--C
       DISTP1 = SQRT((ELEVDATA(CL1J,1) - STRDATA(I,1))**2 +
     C (ELEVDATA(CL1J,2) - STRDATA(I,2))**2)
       DISTP2 = SQRT((ELEVDATA(CL2J,1) - STRDATA(I,1))**2 +
     C (ELEVDATA(CL2J,2) - STRDATA(I,2))**2)
       DIST12 = SQRT((ELEVDATA(CL1J,1) - ELEVDATA(CL2J,1))**2 +
     C (ELEVDATA(CL1J,2) - ELEVDATA(CL2J,2))**2)

C------INTERPOLATE, BASED ON RELATIVE POSITIONS, TO FIND THE ELEVATION OF
C------THE STREAM POINT-----------------------------------------------------C
       IF (DIST12 .LE. DISTP1) GO TO 53
        IF (ELEVATE(CL1J) .LT. ELEVATE(CL2J)) GO TO 55
         PELEV = INT(ELEVATE(CL2J) + (ELEVATE(CL1J) -
```

```
   C  ELEVATE(CL2J)) *  DISTP2/DIST12)
      GO TO 60
55     PELEV = INT(ELEVATE(CL1J) + (ELEVATE(CL2J) -
   C  ELEVATE(CL1J)) * DISTP1/DIST12)
      GO TO 60
53     PELEV = ELEVATE(CL1J)

C------WRITE THE STREAM POINT ELEVATION TO THE OUTPUT ARRAY---------C
60     OUTAR(I,1) = STRNODE(I)
       OUTAR(I,2) = PELEV
40     CONTINUE

C=====WRITE THE OUTPUT ARRAY TO THE OUTPUT FILE==============C

       DO 70 I=1,CNTSTR
       WRITE (10,110) OUTAR(I,1), OUTAR(I,2)
70     CONTINUE

C=====CLOSE ALL FILES AND EXIT=====================C

       CLOSE (10)
       CLOSE (11)
       CLOSE (12)

       END
```

FIXSNAP.AML

Description

This AML program attempts to correct for a peculiar operational characteristic of the *SNAP* command. In a sufficiently large cover, if two points lie extremely close together, then they will not be affected by the *SNAP* command. It, in effect, does not interpret them as separate points. In order to correct for this, FIXSNAP.AML first *BUFFER*s all points in the input cover with, relatively, very "small" *BUFFER*s (fig. 28). It then creates a point cover from the label points of these *BUFFER*s. Because label points occur at the center of their polygons, then for normal, single points the output point will be in the exact same position as the input point. For the "double" points, however, the individual *BUFFER*s around the points will be merged into one larger *BUFFER* and, correspondingly, only one output point will be created. That output point will lie between the original input points. The main difference between FIXSNAP.AML and SNAPPY.AML is that SNAPPY.AML *SNAP*s points with a user-supplied, relatively "large" *SNAP* distance. FIXSNAP.AML only uses a "small" *SNAP* distance and hence should not be used in place of the *SNAP* command, but only if the extremely close point problem described above occurs.

An output mesh with very acute triangles may result if the problem of extremely close points is not resolved. Our example did not show this, but other experimenting with mesh generation has. If this does occur, then FIXSNAP.AML can be included in KITSINK.AML just after all point coverages have been combined into the master point cover and before the master polygon cover is created using ARCPOTIN.AML and *ARCTIN*. FIXSNAP.AML is also useful as a stand-alone module.

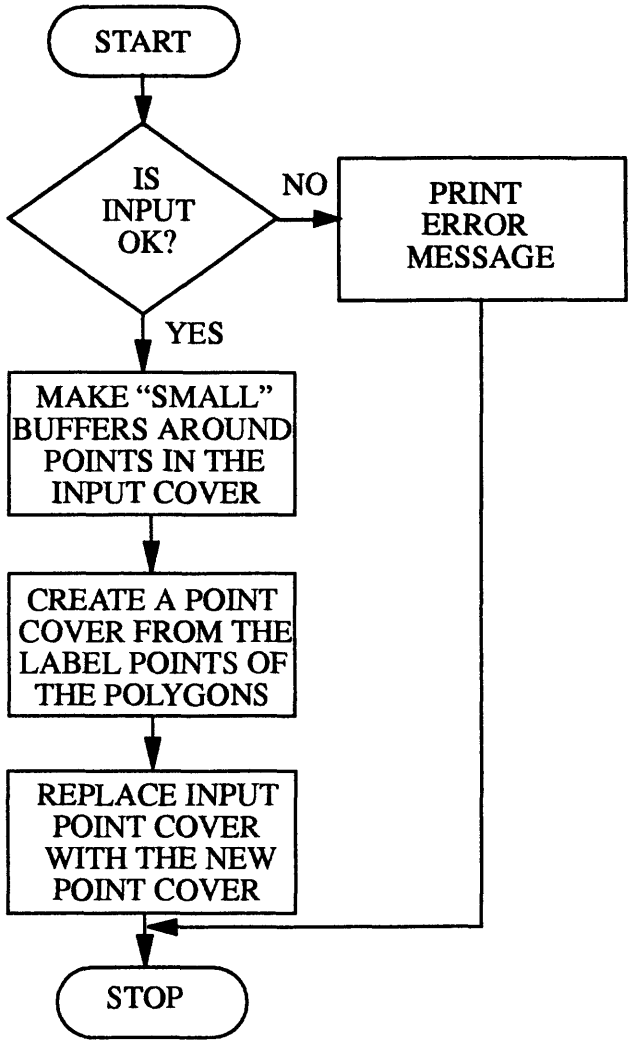


Figure 28.--Flowchart for FIXSNAP.AML.

Program Listing

```
/* MACRO: Take an input cover and eliminate any points which lie too close
/*              for SNAP to detect as distinct points
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*        DTYPE: The graphic display terminal type code
/*          COV: The cover to be snapped
/* TOLERANCE: The maximum distance across which snapping may occur
/*      WEEBUF: The cover containing very small buffered areas surrounding
/*                  each of the point locations from cov
```

```
&echo &off
&args dtype cov tolerance

/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 7]
&setvar i [close %filun%]

/* -Test the input to see if all arguments are present as expected-O
&if [type %dtype%] ne 1 &then &goto proceed
&if [len %dtype%] eq 0 &then &goto badentry
&label proceed
&if [type %cov%] ne 1 &then &goto badentry
&if [length %cov%] eq 0 &then &goto badentry
&if [type %tolerance%] ne -1 &then goto badentry

/* -Eliminate unecessary coverages-M
&severity &error &ignore
kill weebuf all
&severity &error &fail

/* -Establish a relatively "small" distance based on the snap distance-E
&setvar bufdis %tolerance% * 0.4

/* -Use buffers in order to remove multiple points with VERY close
/* location, a condition which prevents the SNAP command from
/* working properly-D
remepf -prg -na -nq -nvfy
build %cov% point
&SEVERITY &ERROR &IGNORE
buffer %cov% weebuf # # %bufdis% 40 point
&SEVERITY &ERROR &FAIL
build weebuf point
kill %cov% all
copy weebuf %cov%
kill weebuf all
&goto endit

/* -Print the error message-Y
&label badentry
&type Usage: FIXSNAP<display type><point cover(existing)><snapping tolerance>
&goto enderr

&label endit
/* -Prepare the error-indication file-
&s i[delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]

&label enderr
&type End of FIXSNAP
```

FREUD.AML

Description

FREUD.AML simplifies the relation between an arc cover and its associated point cover. It modifies the arc attribute table of the arc cover so that the from- and to- node numbers for each arc correspond to the node numbers in the point attribute table of the node cover (fig. 29). This direct correspondence allows selection of desired arcs by intersecting a polygon outline with the associated nodes in the node cover. FREUD.AML changes the node numbers in the AAT file only. It does not change the node-ID's in the binary ARC file. Any AML program, such as MODEL.AML, that reads the node-ID's from the binary ARC file and assumes them to be equal to the node numbers in the AAT file should not be used on a cover once FREUD.AML has been run. FREUD.AML should not be used on a cover that has had linear discontinuities added to it. For example, such discontinuities could be added to represent geologic fault lines. A discontinuity consists of two separate points that are defined at the same location in order to allow an abrupt change in mesh properties.

Common errors in using FREUD.AML include attempting to use it on a cover with linear discontinuities, and using an arc cover whose nodes do not map directly onto the point cover nodes.

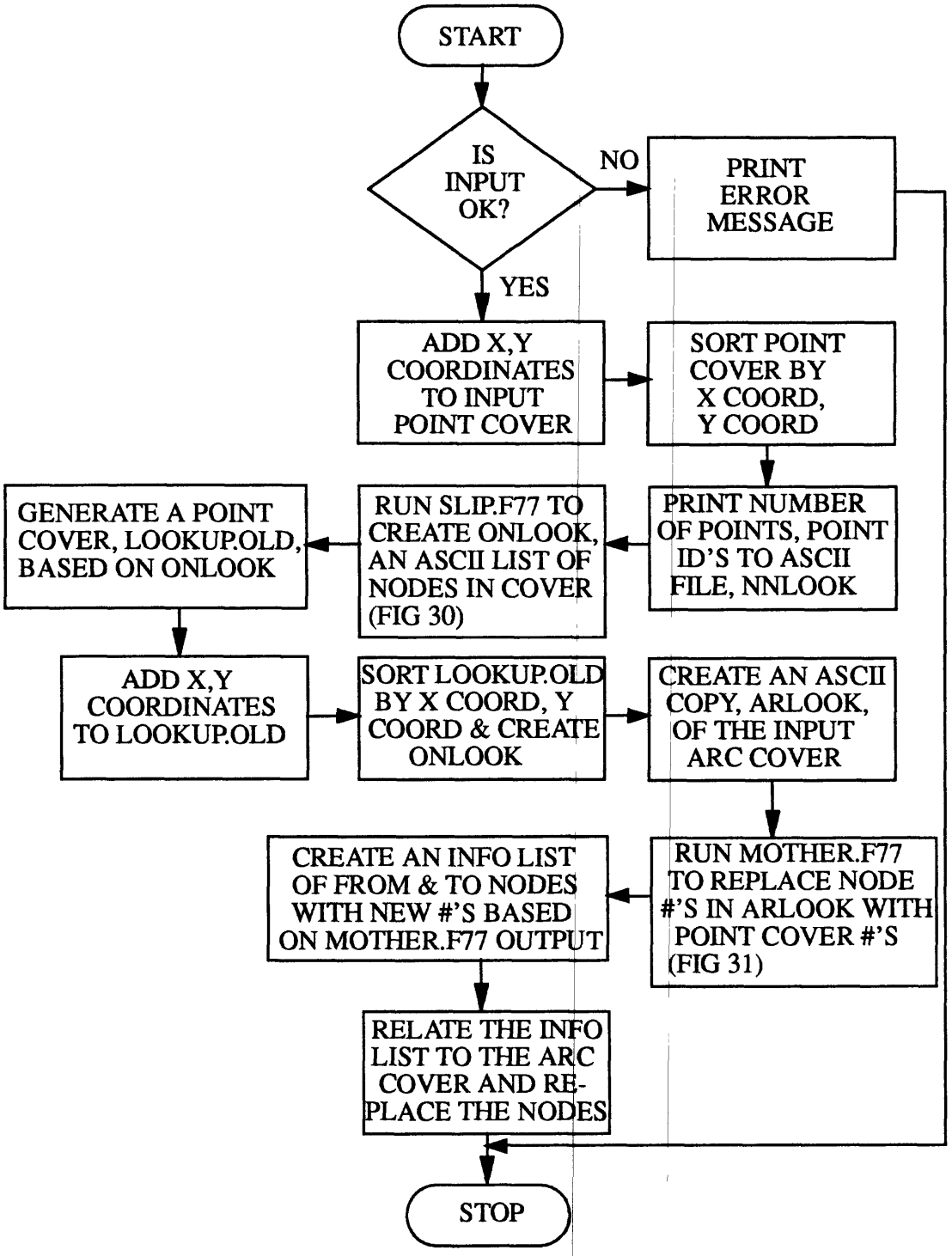


Figure 29.--Flowchart for FREUD.AML.

```
/*    MACRO: Change the node numbers in an arc cover to match the node
/*                numbers in a related point cover.
/*    CODED BY: Robert Lowther
/*    SUPERVISED BY: Eve L. Kuniansky

/*    VARIABLE LIST:
/*      PNTCOV: The input point cover with correct node numbers
/*      ARCCOV: The input arc cover with incorrect node numbers
/*      ARCAAT: The name of ARCCOV's PAT file
/*     GOODPAT: The name of PNTCOV's PAT file
/*      X-COORD: The x-coordinate of a node as listed in the PAT file
/*      Y-COORD: The y-coordinate " " "   " "    " " " "
/*        NEWID: The node number from the point cover
/*       NNLOOK: The ASCII file containing arc cover-based node numbers
/*                ordered by their x and y-coordintes
/*       ONLOOK: The ASCII file containing node cover-based node numbers,
/*                both before and after sorting by x and y-coordinates
/* LOOKUP.OLD: The point cover based on the unsorted ONLOOK
/*       ARLOOK: The ASCII file containing the from and to nodes as ordered
/*                in the arc cover
/*      NEWARC: The ASCII file created containing the from and to nodes from
/*                the point cover as ordered in the arc cover
/*        DANTE: The INFO file based on NEWARC

&echo &off
&args pntcov arccov

/* -Check the computer type (by Leonard l. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test to see if all arguments are present as expected-O
&if [type %pntcov%] ne 1 &then &goto badentry
&if [type %arccov%] ne 1 &then &goto badentry
&if [length %arccov%] eq 0 &then &goto badentry

/* -Prepare the error-indication file-M
&s i [delete coderr]
&setvar filunit [open coderr openstatus -w]
&setvar i [write %filunit% 8]
&setvar i [close %filunit%]
```

```
/*  -Construct the new node number lookup table-E
&setvar goodpat [translate %pntcov%].PAT
addxy %pntcov%

/*  -Sort the point cover-D
/*  -Generate the necesary ASCII file-Y
&setvar newid [translate %pntcov%]-ID
&setvar newint [translate %pntcov%]#
&setvar nnlook [translate [pathname NNLOOK]]
&s i [delete %nnlook%]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 CALC $COMMA-SWITCH = -1
 SELECT %goodpat%
 SORT ON X-COORD,Y-COORD
 OUTPUT %nnlook% INIT
 PRINT [TRIM $NOREC]
 PRINT %newid%
 SORT ON %newint%
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
CALC $COMMA-SWITCH = -1
SELECT %goodpat%
SORT ON X-COORD,Y-COORD
OUTPUT %nnlook% INIT
PRINT [TRIM $NOREC]
PRINT %newid%
SORT ON %newint%
Q STOP
QUIT
&end
&end

/*  -Construct the old node number lookup table-
&s i [delete onlook]
&setvar arcnam %arccov%
remepf -prg -na -nq -nvfy

/*  -Create an ASCII file based on the node numbers from the ARC file-H
&s i [delete slipinfo]
&s filunit [open slipinfo openstatus -w]
&s i [write %filunit% %arcnam%]
&s i [close %filunit%]
&if .computer_type = 'prime' &then &sys r slip
&else &sys slip.out
```

```
/*  -Create a point cover based on the ASCII file-O
&severity &error &ignore
kill lookup.old all
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
generate lookup.old
 input onlook
 points
quit
&end
&else
&do
&data arc generate lookup.old
input onlook
points
quit
&end
&end
build lookup.old point
addxy lookup.old

/*  -Create the necessary ASCII file based on the point cover-U
&s i [delete onlook]
&setvar onlook [pathname ONLOOK]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SELECT LOOKUP.OLD.PAT
 CALC $COMMA-SWITCH = -1
 SORT ON X-COORD,Y-COORD
 OUTPUT %onlook% INIT
 PRINT LOOKUP.OLD-ID
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT LOOKUP.OLD.PAT
CALC $COMMA-SWITCH = -1
SORT ON X-COORD,Y-COORD
OUTPUT %onlook% INIT
PRINT LOOKUP.OLD-ID
Q STOP
QUIT
&end
&end

/*  -Create a facsimile of the mesh AAT which F77 can read-R
&setvar arlook [pathname ARLOOK]
&s i [delete %arlook%]
```

```
&setvar arcaat [translate %arccov%].AAT
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 CALC $COMMA-SWITCH = -1
 SELECT %arcaat%
 OUTPUT %arlook% INIT
 PRINT [TRIM $NOREC]
 PRINT FNODE#,TNODE#
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
CALC $COMMA-SWITCH = -1
SELECT %arcaat%
OUTPUT %arlook% INIT
PRINT [TRIM $NOREC]
PRINT FNODE#,TNODE#
Q STOP
QUIT
&end
&end

/*  -Pace through the AAT facsimile and create a new facsimile using the
/*   node numbers from the node cover instead of those from the arc cover-
&setvar dumarg DummY
&severity &error &ignore
&s i [delete newarc]
&severity &error &fail
&if .computer_type = 'prime' &then &sys r mother
&else &sys mother.out

/*  -Create an INFO file based on the new facsimile-S
&setvar newarc [pathname NEWARC]
&severity &error &ignore
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SELECT DANTE
 PURGE
 Y
 ERASE DANTE
 Y
 DEFINE DANTE
  FNODE,6,6,I
  TNODE,6,6,I

GET %newarc% COPY
Q STOP
```

```
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT DANTE
PURGE
Y
ERASE DANTE
Y
DEFINE DANTE
FNODE,6,6,I
TNODE,6,6,I

GET %newarc% COPY ASCII
Q STOP
QUIT
&end
&end

/*  -Prepare the original arc file and the facsimile-based INFO file
/*   for relate-O
additem %arcaat% %arcaat% reln 5 5 i
additem dante dante reln 5 5 i

/*  -Replace the from and to node numbers in the original arc file with the
/*   new node numbers obtained from the INFO file based on the orig node cover-F
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 SELECT %arcaat%
 CALC RELN = $RECNO
 SELECT DANTE
 CALC RELN = $RECNO
 RELATE %arcaat% BY RELN
 CALC $1FNODE# = FNODE
 CALC $1TNODE# = TNODE
 PURGE
 Y
 ERASE DANTE
 Y
 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %arcaat%
CALC RELN = $RECNO
SELECT DANTE
```

```
CALC RELN = $RECNO
RELATE %arcaat% BY RELN
CALC $1FNODE# = FNODE
CALC $1TNODE# = TNODE
PURGE
Y
ERASE DANTE
Y
Q STOP
QUIT
&end
&end

/* -Eliminate all temporary files-T
dropitem %arcaat% %arcaat% reln
dropitem %goodpat% %goodpat% x-coord
dropitem %goodpat% %goodpat% y-coord
&s i [delete slipinfo]
&s i [delete onlook]
&s i [delete nnlook]
&s i [delete arlook]
&s i [delete newarc]
kill lookup.old all

/* -Prepare the error-indication file-W
&s i [delete coderr]
&setvar filunit [open coderr openstatus -w]
&setvar i [write %filunit% noerr]
&setvar i [close %filunit%]

&goto endit

/* -Print the error message-A
&label badentry
&type Usage: FREUD <point cover with correct node numbers (existing)> <arc
&type        cover whose from and tonodes are to be changed(existing)>

/* -End of program message-R
&label endit
&type End of FREUD
/* -End of program-E
```

Fortran Program SLIP.F77

Description

This program extracts needed information from the arc cover's binary file and writes it to an ASCII file that can be read by FREUD.AML (fig. 30). The binary file contains the from- and to- nodes for each arc, as well as their X- and Y-coordinates. SLIP.F77 compiles an ASCII list of all nodes in the current cover. It writes one output record for each node instead of one for each occurrence of a node as is done in the binary file. SLIP.F77 therefore eliminates the repetition of node numbers, which occurs when information is stored by ARC/INFO.

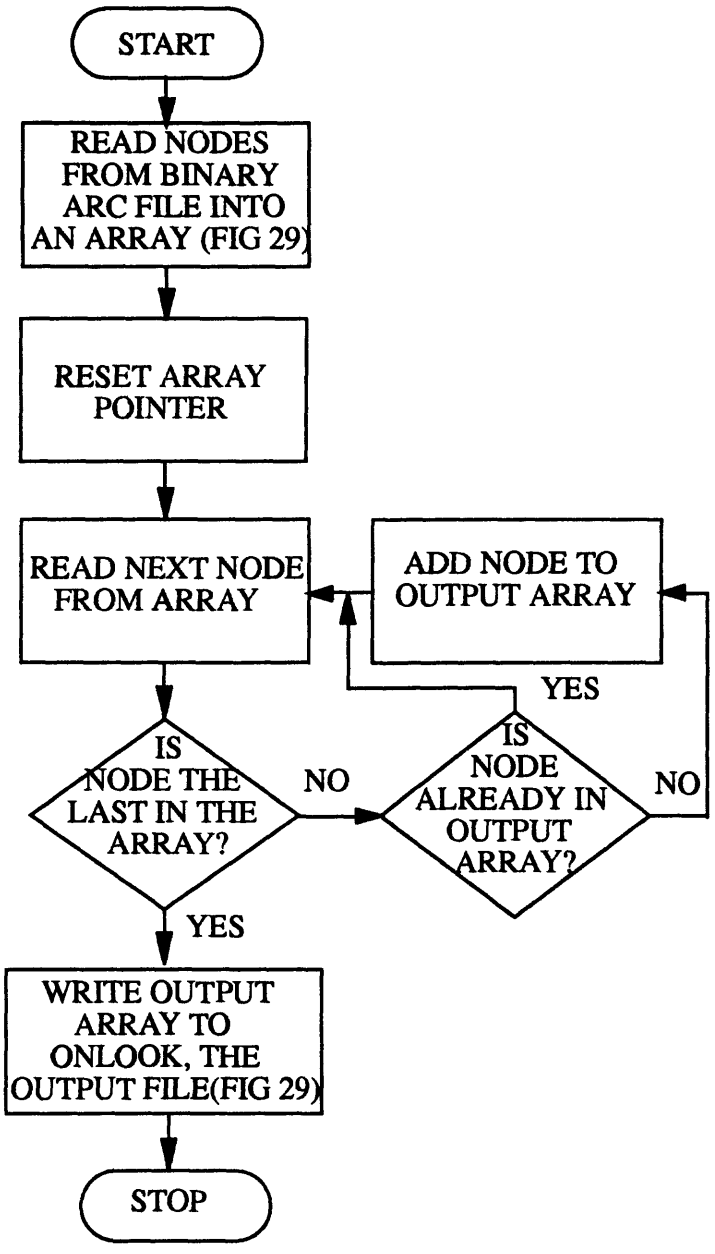


Figure 30.--Flowchart for SLIP.F77.

Program Listing

```
C*********************************************************************C
C   PROGRAM: Read the binary ARC file and create an ASCII file w/ the info      C
C   CODED BY: Robert Lowther                                                C
C   SUPERVISED BY: Eve L. Kuniansky                                          C
C*********************************************************************C

    PROGRAM SLIP

    COMMON /C1/OUTIN(25000),USERID(25000),FRNODE(25000),TONODE(25000)
    COMMON /C2/LPOLY(25000),RPOLY(25000),NPT(25000)
    COMMON /C3/COORDS(25000,6),OUTARY(25000,2)

C*********************************************************************C
C   VARIABLE LIST:                                                          C
C                                                                           C
C      IABUFF: The standard ARC record, as read from the ARC file           O
C    USERDUM: The integer equivalent to the first IABUFF record              C
C       FRDUM: The IABUFF record corresponding to FRNODE                     M
C       TODUM: Corresponds to TONODE                                    C
C       LPDUM: The record representing the left polygon                      C
C       RPDUM: The "    "       " right polygon                             H
C     NPTDUM: The "    "       " number of points on an arc                 C
C      ACCESS: The access mode for opening the ARC file                      R
C      TEMPRY: Code number to indicate a normal ARC file                     C
C       OUTIN: The output integerarray used to create the ASCII ouput file   C
C      OUTFND: A flag indicating that the tested item is already in OUTIN     C
C      OUTFND: A flag indicating the tested item is already in the outarray   S
C      FRNODE: The from node array                                          C
C      TONODE: The to node array                                            O
C      KANARC: The ARCFILE channel number                                    C
C          I,J,K: Counting variables                                        F
C      IERROR: Flag indicating trouble while opening a file                   C
C   CORDDUM: Corresponds to COORDS                                          T
C      COORDS: The coordinate array read from the input ARC file             C
C      OUTARY: The output real number array                                 W
C      ARCCOV: The name of the ARC cover                                    C
C      FILENM: The output file name for the old node lookup file              A
C      DUMRET: The dummy argument returned to the macro                      C
C*********************************************************R

    INTEGER IABUFF(2006),USERDUM,FRDUM,TODUM,LPDUM,RPDUM,NPTDUM
    INTEGER ACCESS,TEMPRY,OUTIN,OUTFND,FRNODE,TONODE,KANARC,I,J,K
    INTEGER IERROR,badarc,KAN2,USERID,LPOLY,RPOLY,NPT
    DOUBLE PRECISION CORDDUM(1000),COORDS,OUTARY
    CHARACTER*128 ARCCOV,FILENM,DUMRET,FILE2

    EQUIVALENCE (IABUFF(1),USERDUM)
    EQUIVALENCE (IABUFF(2),FRDUM)
    EQUIVALENCE (IABUFF(3),TODUM)
    EQUIVALENCE (IABUFF(4),LPDUM)
    EQUIVALENCE (IABUFF(5),RPDUM)
    EQUIVALENCE (IABUFF(6),NPTDUM)
```

```
      EQUIVALENCE (IABUFF(7),CORDDUM(1))

      EXTERNAL LUNINI,MINIT,ARCOPN,ARCRD,AOPEN,MESINI,
     &ACLOSE,ARCCLS,AMLFNA,ACREAT,AENTER,VINIT

      CALL AENTER
      CALL LUNINI
      CALL MINIT
      CALL VINIT
      CALL MESINI

100   FORMAT (I6,',',F15.3,',',F15.3)
200   FORMAT (A3)
      FILENM = 'onlook'
      ACCESS = 2
      TEMPRY = 1

C-----Determine the ARC cover name----------------------------------------C
      OPEN (7,FILE= 'slipinfo')
      READ (7,*) ARCCOV
      CLOSE (7)

C-----Open the input and output files-------------------------------------C
      CALL ARCOPN (KANARC,ARCCOV,ACCESS,TEMPRY,IERROR)
      CALL ACREAT (LOUT,FILENM,IER)

C-----Read the ARC binary file--------------------------------------------C

      NOIN = 1
      irec = 1
      badarc = 0
10    CALL ARCRDr (KANARC,IREC,IABUFF,IERROR)
      if (nptdum .gt. 2) print *,nptdum
      USERID(NOIN) = USERDUM
      FRNODE(NOIN) = FRDUM
      TONODE(NOIN) = TODUM
      LPOLY(NOIN) = LPDUM
      RPOLY(NOIN) = RPDUM
      NPT(NOIN) = NPTDUM
      DO 9 I = 1,NPTDUM*2
       COORDS(NOIN,I) = CORDDUM(I)
9     CONTINUE
      NOIN = NOIN + 1
      IF (IERROR .EQ. -1) GO TO 13
      IF (IERROR .EQ. -2) GO TO 15
     irec = irec + 1
     GO TO 10
13    NOIN = NOIN - 1
C-----Check the output file to see if tested record is already there------C

      NOOUT = 0
      DO 17 I=1,NOIN
       OUTFND = 0
       DO 18 J=1,NOOUT
```

```
      IF (OUTIN(J) .NE. FRNODE(I)) GO TO 18
       OUTFND = 1
18    CONTINUE

C------Write to the output file if OK----------------------------------C

      IF (OUTFND .EQ. 1) GO TO 19
       NOOUT = NOOUT + 1
       OUTIN(NOOUT) = FRNODE(I)
       OUTARY(NOOUT,1) = COORDS(I,1)
       OUTARY(NOOUT,2) = COORDS(I,2)

C------Check and write the to node--------------------------------------C

19    OUTFND = 0
      DO 21 J=1,NOOUT
       IF (OUTIN(J) .NE. TONODE(I)) GO TO 21
       OUTFND = 1
21    CONTINUE
      IF (OUTFND .EQ. 1) GO TO 22
       NOOUT = NOOUT + 1
       OUTIN(NOOUT) = TONODE(I)
       OUTARY(NOOUT,1) = COORDS(I,3)
       OUTARY(NOOUT,2) = COORDS(I,4)
22    OUTFND = 0
17    CONTINUE

C-----Write the output array to the output file-----------------------C

      DO 23 I=1,NOOUT
       WRITE (LOUT,100) OUTIN(I),(OUTARY(I,J),J=1,2)
23    CONTINUE
      WRITE (LOUT,'(A3)') 'END'

      GO TO 20

15    WRITE (*,'(A)') 'Error occurred during ARCRD'

20    ENDFILE(LOUT)
      CALL ACLOSE (LOUT)
      CALL ARCCLS (KANARC)
      END
```

Description

This program translates the node numbers in an ASCII file based on an arc AAT file into the node numbers from the node cover PAT file (fig. 31). MOTHER.F77 uses three input files. The first of these files is an ASCII file of node numbers written in the format that ARC/INFO uses: it is based upon the arcs and, therefore, repeats node numbers wherever two or more arcs meet at the same node. The second and third input files are lists of the nodes in the arc cover in question, with each node listed only once. The second file contains arc-cover-based numbers, the node numbers used in the original arc cover, those used in the first input file. The third contains node-cover-based numbers, the node numbers used in the original node file. These two input files are written, side by side, into a "lookup table." This table becomes, in effect, a "dictionary" that translates from arc-cover-based numbers to node-cover-based numbers. The node numbers in the first ASCII file, those from the arc-cover-formatted file, can be "translated" as follows.

A node number is read from the input arc-cover-formatted file and located in the first column of the lookup table. The corresponding node in the table's second column is then written in place of the original node number in the arc-cover-formatted file. In this manner, all of the node numbers in the arc-cover-formatted file can be replaced with the corresponding node-cover-based numbers. The arc-cover-formatted file is then used to change the node numbers in the line cover so that they match those in the corresponding point cover.

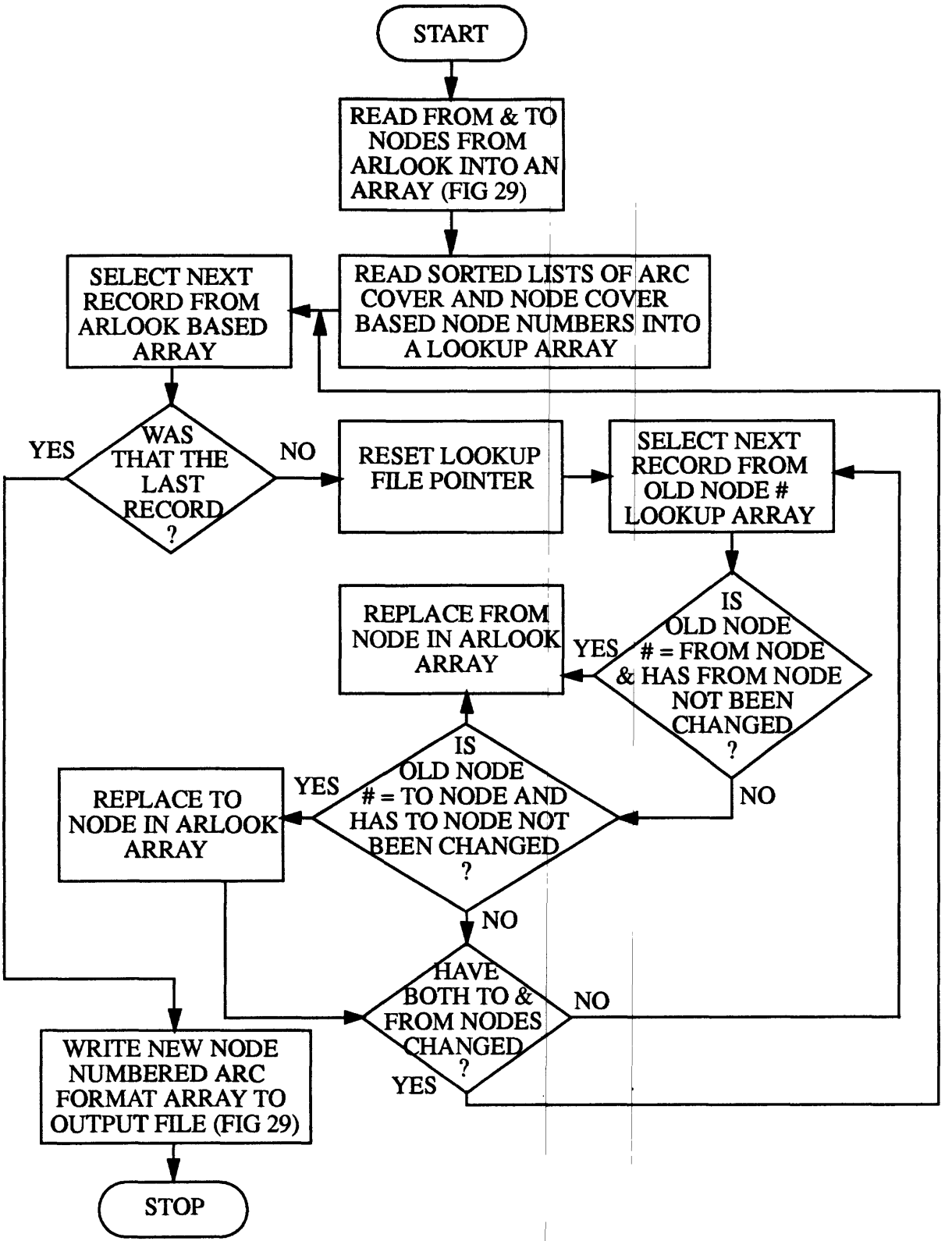


Figure 31.--Flowchart for MOTHER.F77.

Program Listing

```
C*****************************************************************C
C   PROGRAM: Create an ASCII file based on arc file with node file node #s      C
C   CODED BY: Robert Lowther                                        C
C   SUPERVISED BY: Eve L. Kuniansky                                 C
C*****************************************************************C

    PROGRAM MOTHER

    COMMON /C1/ ARCARY(25000,2),NODREL(25000,2)

C*****************************************************************C
C   VARIABLE LIST:                                                H
C                                                                 S
C    NUMARC: The number of arcs in the AAT file                     C
C    NUMNOD: The number of nodes in the PAT file                    C
C     FNODE: The from node as read from the AAT file                C
C     TNODE: The to node as read from the AAT file                  C
C     NNODE: The node number in the ASCII file as read from the PAT file      C
C     ONODE: The " " " " " " " " " " AAT "                          H
C    ARCARY: The array of node numbers, as read from the AAT file     S
C    NODREL: The array of node numbers, both old and new             C
C       FND: A flag indicating that translations for both nodes in an arc     C
C            have been found                                        C
C      FNDF: A flag indicating that the from node has been found       C
C      FNDT: A " " " " to " " " "                                     C
C       I,J: Counting variables                                       H
C*****************************************************************S

    INTEGER NUMARC,NUMNOD,FNODE,TNODE,NNODE,ONODE,ARCARY
    INTEGER NODREL,FND,FNDF,FNDT,I,J

    OPEN (UNIT= 9,FILE= 'arlook')
    OPEN (UNIT= 10,FILE= 'onlook')
    OPEN (UNIT= 7,FILE= 'nnlook')
    OPEN (UNIT= 8,FILE= 'newarc')

100  FORMAT (I6)
200  FORMAT (2I6)

C-----Read in the size of the arc and the node input files----------C

    READ (9,*) NUMARC
    READ (7,*) NUMNOD

C-----Read the input arc-based node numbers into an array----------C

    DO 1 I=1,NUMARC
     READ (9,200) FNODE, TNODE
     ARCARY(I,1) = FNODE
     ARCARY(I,2) = TNODE
1    CONTINUE
```

```
C-----Read the arc-based and node-based lookup files into a lookup array-C

      DO 2 I=1,NUMNOD
       READ (10,100) ONODE
       READ (7,100) NNODE
       NODREL(I,1) = ONODE
       NODREL(I,2) = NNODE
2     CONTINUE

C=====Build a file containing the new node numbers in the same order as the AAT

      DO 3 I=1,NUMARC
       J = 1
       FND = 0
       FNDF = 0
       FNDT = 0
C------Check the selected lookup item against both items in the current line
C     of the AAT-based node file----------------------------------------C

4     IF (NODREL(J,1) .NE. ARCARY(I,1)) GO TO 5
      IF (FNDF .NE. 0) GO TO 5
      ARCARY(I,1) = NODREL(J,2)
      FNDF = 1
5     IF (NODREL(J,1) .NE. ARCARY(I,2)) GO TO 6
      IF (FNDT .NE. 0) GO TO 6
      ARCARY(I,2) = NODREL(J,2)
      FNDT = 1

C------Increment the test line and check to see if both changes are made---C

6     J = J + 1
      FND = FNDF + FNDT
      IF (FND .LT. 2) GO TO 4
3     CONTINUE

C-----Write the output file---------------------------------------------C

      DO 7 I=1,NUMARC
       WRITE (8,200) ARCARY(I,1),ARCARY(I,2)
7     CONTINUE
      CLOSE (9)
      CLOSE (10)
      CLOSE (7)
      CLOSE (8)

      END
```

IDENTIFY.AML

Description

Identification of features is essential when dealing with several input coverages. It is for this reason that IDENTIFY.AML is useful (fig. 32). It can add one user-specified item each to up to 10 coverages and assign those items a value of "1" so that when all of the input features are combined into one output cover, the output features can be identified and grouped according to input cover origin. It has one further, optional feature. If a target cover is specified and if the (up to) 10 coverages are polygon coverages, then IDENTIFY.AML will perform an intersection of its newly identified polygon cover with the target cover. This effectively uses the given identifying item to identify parts of the target cover by the polygon that overlays them.

At the beginning of its run, IDENTIFY.AML *KILL*s a cover called "CAESER." One common error in using IDENTIFY.AML is to try to identify coverages of more than one type. All 10 (or as many as used) coverages must be of the same type.

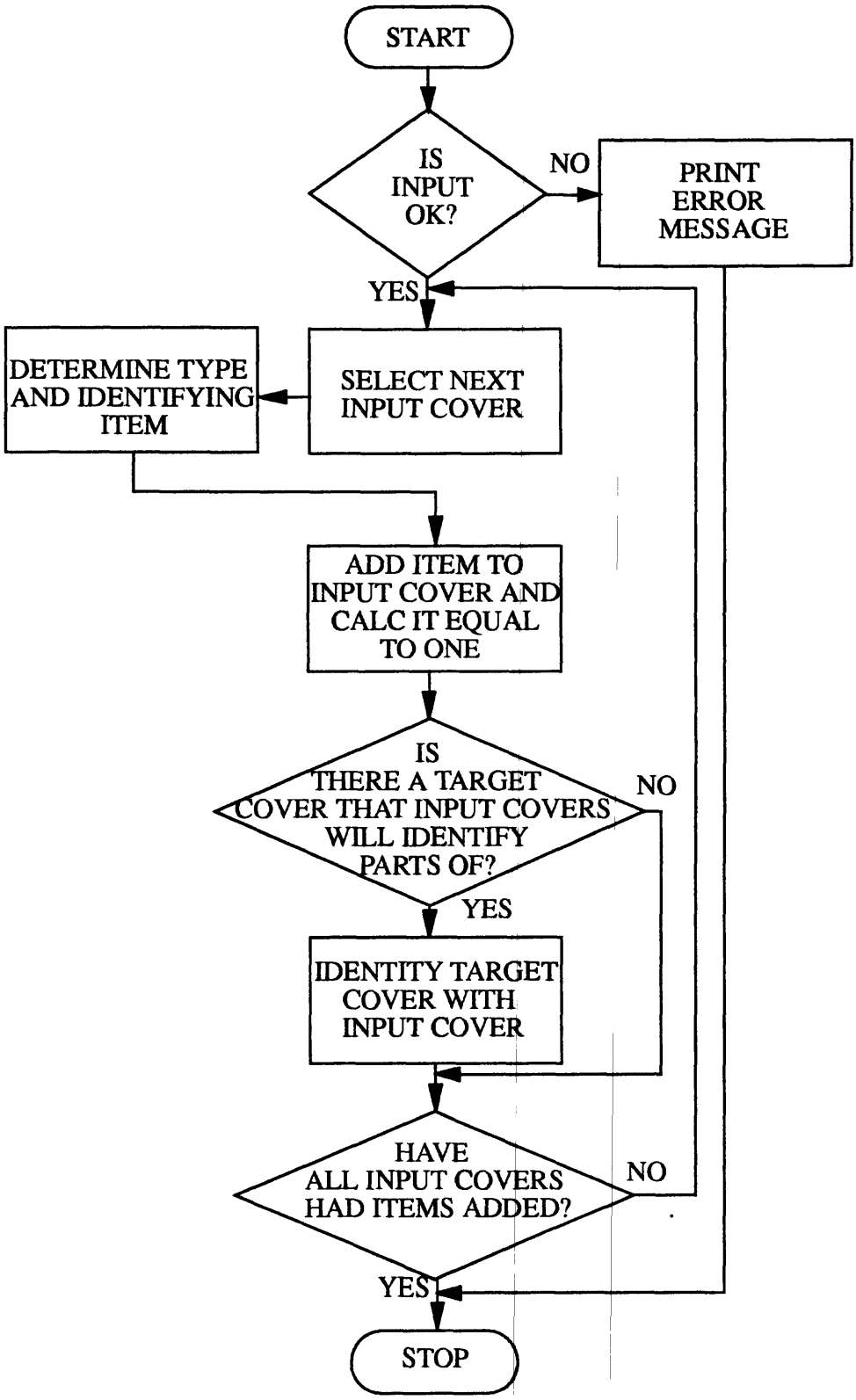


Figure 32.--Flowchart for IDENTIFY.AML and IDENTI2.AML.

```
/* MACRO:  Add an item to up to ten coverages of the same type and
/*              "calc" that item equal to one to identify the elements.
/*              Optionally, it will identity a specified cover with each
/*           input cover, given that the input coverages are polygonal.
/*           Note:  This macro will not buffer point or line coverages in order
/*           to be able to identify the specified cover's items.  In order
/*           to do that, use IDENTILOTS

/* CODED BY:  Robert Lowther
/* SUPERVISED BY:  Eve L. Kuniansky

/* VARIABLE LIST:
/*        TIP:  The feature type, as input by the user
/*     C1-C10:  The cover names, as input by the user
/*      I1-I10:  The item names, as input by the user
/*        TYP:  The capitalized version of "tip"
/*        COV:  The name of the cover currently being considered
/*      COVER:  The capitalized version of "cov"
/*       ITEM:  The name of the item currently being considered
/*    COVTAB:  The name of the attribute table to which the item is added
/*     DIFFER:  The name of the cover to be identitied with each input cover
/*    CAESER:  A temporary cover used to identity the specified cover
/*      SHINY:  Are the poly coverages buffers of other coverages?

&echo &off
&args tip c1 i1 c2 i2 c3 i3 c4 i4 c5 i5 c6 i6 c7 i7 c8 i8 ~
 c9 i9 c10 i10

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test the input to see if all arguments are present as expected-O
&if [type %tip%] ne 1 &then &goto badentry
&if [type %c1%] ne 1 &then &goto badentry
&if [type %i1%] ne 1 &then &goto badentry
&if [length %i1%] eq 0 &then &goto badentry

&setvar typ [translate %tip%]
&if %typ% ne 'POLY' &then &goto postq
&setvar shiny [response 'Are these coverages buffers of other coverages? (y/n)']
&if %shiny% eq 'y' &then &setvar typ 'POLD'
```

```
&if %shiny% eq 'Y' &then &setvar typ 'POLD'
&label postq
&setvar differ [response 'Enter the cover which is to be differentiated (or ~
null)']
&if [length %differ%] ne 0 &then
&do
 &setvar dtip [response 'Enter the cover type']
 &setvar dtyp [translate %dtip%]
&end

/* -Eliminate any unecessary coverages-M
&severity &error &ignore
kill Caeser all
&severity &error &fail

/* -Loop through all of the coverages-E
&do cov &list %c1% %c2% %c3% %c4% %c5% %c6% %c7% %c8% %c9% %c10%
&if [length %cov%] eq 0 &then &goto endloop

&setvar cover [translate %cov%]

/* -Identify and translate the item to be added-D
&if %cov% eq %c1% &then &setvar item [translate %i1%]
&if %cov% eq %c2% &then &setvar item [translate %i2%]
&if %cov% eq %c3% &then &setvar item [translate %i3%]
&if %cov% eq %c4% &then &setvar item [translate %i4%]
&if %cov% eq %c5% &then &setvar item [translate %i5%]
&if %cov% eq %c6% &then &setvar item [translate %i6%]
&if %cov% eq %c7% &then &setvar item [translate %i7%]
&if %cov% eq %c8% &then &setvar item [translate %i8%]
&if %cov% eq %c9% &then &setvar item [translate %i9%]
&if %cov% eq %c10% &then &setvar item [translate %i10%]
&if %typ% eq 'LINE' &then &setvar covtab %cover%.AAT
&if %typ% eq 'POINT' &then &setvar covtab %cover%.PAT
&if %typ% eq 'POLY' &then &setvar covtab %cover%.PAT
&if %typ% eq 'POLD' &then &setvar covtab %cover%.PAT

&severity &error &ignore
additem %covtab% %covtab% %item% 4 4 i
&severity &error &fail

/* -Set the item equal to one-Y
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %covtab%
&if %typ% eq 'POLY' &then RESEL FOR AREA > 0
&if %typ% eq 'POLD' &then RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
&end
&end
&else
&do
```

```
&data ARC
INFO
ARC
SELECT %covtab%
&if %typ% eq 'POLY' &then RESEL FOR AREA > 0
&if %typ% eq 'POLD' &then RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
QUIT
&end
&end

&if [length %differ%] eq 0 &then &goto endloop
copy %differ% Caeser
kill %differ% all
identity Caeser %cover% %differ% %dtyp% 40
kill Caeser all
&label endloop
&end
&goto endit

/* -Print the error message-
&label badentry
&type Usage: IDENTIFY <type (point,line,poly) of coverages to have identifying
&type              items added> <cover to have item added #1 (existing)>
&type             <identifying  item #1 (created)> ... {cover to have
&type              item added #10 (existing)} {item #10 (created)}
&label endit
&type End of IDENTIFY
```

IDENTI2.AML

Description

This is a modified version of IDENTIFY.AML, designed to be called by KITSINK.AML. It shares a flowchart with IDENTIFY.AML, shown in figure 32. Like CLIPIT2.AML, IDENTI2.AML replaces a screen-prompted input with an argument. This is to enable KITSINK.AML to run uninterrupted.

Program Listing

```
/* MACRO: Add an item to up to ten coverages of the same type and
/*              "calc" that item equal to one to identify the elements.
/*              Optionally, it will identity a specified cover with each
/*              input cover, given that the input coverages are polygonal.
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*       TIP: The feature type, as input by the user
/*    C1-C10: The cover names, as input by the user
/*     I1-I10: The item names, as input by the user
/*       TYP: The capitalized version of "tip"
```

```
/*      COV:  The name of the cover currently being considered
/*    COVER:  The capitalized version of "cov"
/*     ITEM:  The name of the item currently being considered
/*  COVTAB:  The name of the attribute table to which the item is added
/*   DIFFER:  The name of the cover to be identified with each input cover
/*  CAESER:  A temporary cover used to identity the specified cover

&echo &off
&args tip differ dtip c1 i1 c2 i2 c3 i3 c4 i4 c5 i5 c6 i6 c7 i7 c8 i8 ~
 c9 i9 c10 i10

/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 5]
&setvar i [close %filun%]

/* -Check the computer type (by Leonard L. Orzol)-O
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test the input to see if all arguments are present as expected-M
&if [type %tip%] ne 1 &then &goto badentry
&if [type %c1%] ne 1 &then &goto badentry
&if [type %i1%] ne 1 &then &goto badentry
&if [length %i1%] eq 0 &then &goto badentry

/* -Eliminate any unecessary files-E
&severity &error &ignore
kill Caeser all
&severity &error &fail

&setvar typ [translate %tip%]
&setvar dtyp [translate %dtip%]

/* -Loop through all of the coverages-D
&do cov &list %c1% %c2% %c3% %c4% %c5% %c6% %c7% %c8% %c9% %c10%
&if [length %cov%] eq 0 &then &goto endloop

&setvar cover [translate %cov%]

/* -Identify and translate the item to be added-Y
&if %cov% eq %c1% &then &setvar item [translate %i1%]
&if %cov% eq %c2% &then &setvar item [translate %i2%]
```

```
&if %cov% eq %c3% &then &setvar item [translate %i3%]
&if %cov% eq %c4% &then &setvar item [translate %i4%]
&if %cov% eq %c5% &then &setvar item [translate %i5%]
&if %cov% eq %c6% &then &setvar item [translate %i6%]
&if %cov% eq %c7% &then &setvar item [translate %i7%]
&if %cov% eq %c8% &then &setvar item [translate %i8%]
&if %cov% eq %c9% &then &setvar item [translate %i9%]
&if %cov% eq %c10% &then &setvar item [translate %i10%]
&if %typ% eq 'LINE' &then &setvar covtab %cover%.AAT
&if %typ% eq 'POINT' &then &setvar covtab %cover%.PAT
&if %typ% eq 'POLY' &then &setvar covtab %cover%.PAT

&severity &error &ignore
additem %covtab% %covtab% %item% 4 4 i
&severity &error &fail

/* -Set the item equal to one-
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %covtab%
&if %typ% eq 'POLY' &then RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %covtab%
&if %typ% eq 'POLY' &then RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
QUIT
&end
&end
&if [length %differ%] eq 0 &then &goto endloop
copy %differ% Caeser
kill %differ% all
identity Caeser %cover% %differ% %dtyp% 40
kill Caeser all
&label endloop
&end
&goto endit

/* -Print the error message-
&label badentry
&type Usage:  IDENTI2 <type (point,line,poly)> <cover to be differentiated>
&type              <type of said cover> <cover #1>
&type         <identifying item #1> ... <cover #10> <item #10>
&goto enderr
```

```
&label endit
/* -Prepare the error-indication file-
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]

&label enderr
&type End of IDENTI2
```

IDENTILOTS.AML

Description

This AML program is, in effect, a companion program to IDENTIFY.AML. IDENTIFY.AML adds identifying items to up to 10 coverages. As an option, it will *IDENTITY* a specified cover with these 10 coverages if, and only if, the 10 coverages are polygon coverages. IDENTILOTS.AML identifies one cover by up to 10 identifying coverages (fig. 33), just as IDENTIFY.AML optionally does, but it does not require the identifying coverages to be polygon coverages. In contrast to IDENTIFY.AML, which is designed primarily to add identifying items to up to 10 coverages, IDENTILOTS is designed to identify the points in one specified cover based on the location of features in up to 10 other coverages. The identifying coverages for IDENTILOTS.AML are input as simple point or line coverages. IDENTILOTS.AML performs a BUFFERing operation to create polygon coverages from the identifying coverages, using the input item names. This AML program was originally taken from the KITSINK.AML program, and hence is nearly duplicated in that macro. IDENTILOTS.AML is useful if only this function of KITSINK.AML is desired.

One common error that occurs when using IDENTILOTS.AML is to use identifying coverages with too much detail. These coverages will not *BUFFER* properly, and hence cause problems. If this occurs, then the identifying coverages must be *SPLINE*d before the AML program will run. Also, because IDENTILOTS.AML calls BUFFNSHINE.AML, if there already exists a cover with the input cover name root and the ".BUF" extension, no *BUFFER*ing will be performed and the existing .BUF cover will be used. If this is not desired, then the .BUF cover must be *KILL*ed.

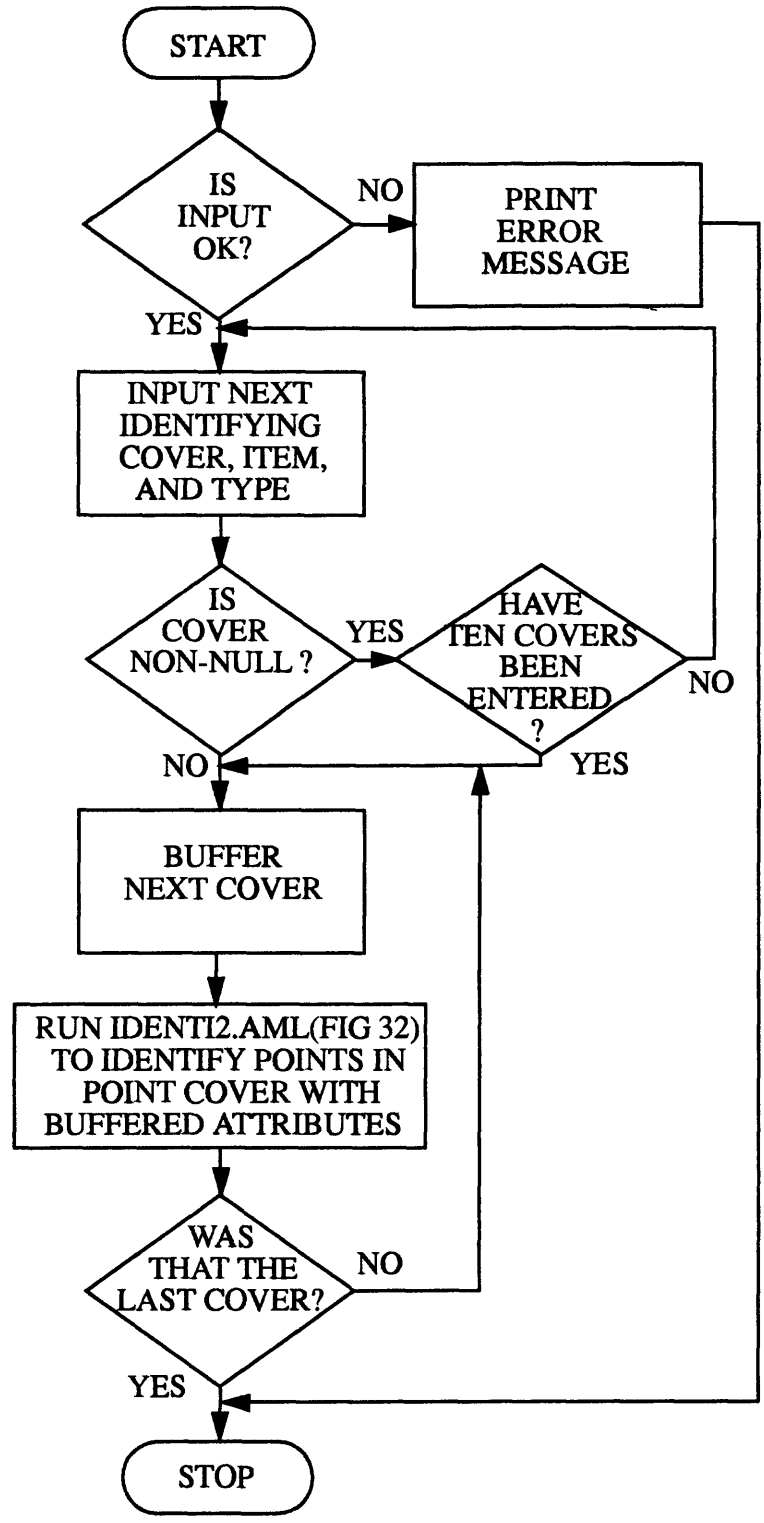


Figure 33.--Flowchart for IDENTILOTS.AML.

```
/* MACRO:  Identify points in one point cover by noting their
/*         proximity to features in many other feature coverages
/*         via an item in the point cover's PAT
/* CODED BY:  Robert Lowther
/* SUPERVISED BY:  Eve L. Kuniansky

/* VARIABLE LIST:
/*   MSTPOL:  The input point cover
/*   MINDIS:  The minimum expected distance between points
/*   MAXITR:  The maximum number of optimizing iterations
/*     C1-C10:  The input line and point coverages
/*       I1-I10:  The item to be associated with each cover to identify it
/*      T1-T10:  The feature type of each cover
/*  NODECRD:  The Node Coordinate Data-based point cover
/*       COV:  The cover currently being considered in a loop
/*     COVER:  The capitalized version of COV
/*      TEMP:  The temporary file associated with the current cover
/*       TIP:  The feature type of the current cover
/*       TYP:  The capitalized version of TIP
/*      ITEM:  The item associated with the current cover
/*      TOL2:  A distance based on MINDIS
/*  ARCHIVE:  The archive files containing the nearly-original versions of
/*            the input coverages
/*   ARCBUF:  The buffered version of ARCHIVE

&echo &off
&args  mstpol mindis

/* -Test to see if all arguments are present as expected-C
&if [type %mstpol%] ne 1 &then &goto badentry
&if [type %mindis%] ne -1 &then &goto badentry
&if [length %mstpol%] eq 0 &then &goto badentry

/* Initialize the cover name variables-O
&setvar c1 ''
&setvar c2 ''
&setvar c3 ''
&setvar c4 ''
&setvar c5 ''
&setvar c6 ''
&setvar c7 ''
&setvar c8 ''
&setvar c9 ''
&setvar c10 ''

&setvar c1 [response 'Enter identifying cover name']
&if [length %c1%] eq 0 &then &goto nocovs
&setvar t1 [response 'Enter cover type (line,point,poly)']
&setvar i1 [response 'Enter item representing cover (to be added)']
&setvar c2 [response 'Enter identifying cover name']
&if [length %c2%] eq 0 &then &goto endentry
&setvar t2 [response 'Enter cover type (line,point,poly)']
```

```
&setvar i2 [response 'Enter item representing cover (to be added)']
&setvar c3 [response 'Enter identifying cover name']
&if [length %c3%] eq 0 &then &goto endentry
&setvar t3 [response 'Enter cover type (line,point,poly)']
&setvar i3 [response 'Enter item representing cover (to be added)']
&setvar c4 [response 'Enter identifying cover name']
&if [length %c4%] eq 0 &then &goto endentry
&setvar t4 [response 'Enter cover type (line,point,poly)']
&setvar i4 [response 'Enter item representing cover (to be added)']
&setvar c5 [response 'Enter identifying cover name']
&if [length %c5%] eq 0 &then &goto endentry
&setvar t5 [response 'Enter cover type (line,point,poly)']
&setvar i5 [response 'Enter item representing cover (to be added)']
&setvar c6 [response 'Enter identifying cover name']
&if [length %c6%] eq 0 &then &goto endentry
&setvar t6 [response 'Enter cover type (line,point,poly)']
&setvar i6 [response 'Enter item representing cover (to be added)']
&setvar c7 [response 'Enter identifying cover name']
&if [length %c7%] eq 0 &then &goto endentry
&setvar t7 [response 'Enter cover type (line,point,poly)']
&setvar i7 [response 'Enter item representing cover (to be added)']
&setvar c8 [response 'Enter identifying cover name']
&if [length %c8%] eq 0 &then &goto endentry
&setvar t8 [response 'Enter cover type (line,point,poly)']
&setvar i8 [response 'Enter item representing cover (to be added)']
&setvar c9 [response 'Enter identifying cover name']
&if [length %c9%] eq 0 &then &goto endentry
&setvar t9 [response 'Enter cover type (line,point,poly)']
&setvar i9 [response 'Enter item representing cover (to be added)']
&setvar c10 [response 'Enter identifying cover name']
&if [length %c10%] eq 0 &then &goto endentry
&setvar t10 [response 'Enter cover type (line,point,poly)']
&setvar i10 [response 'Enter item representing cover (to be added)']
&goto endentry
&label nocovs
&type 'At least one cover must be entered'
&goto endit
&label endentry

&setvar nodecrd %mstpol%

/* -Create buffers around each of the archived coverages-M
&setvar tol2 %mindis% * 0.6
&do cov &list %c1% %c2% %c3% %c4% %c5% %c6% %c7% %c8% %c9% %c10%
&if [length %cov%] eq 0 &then &goto endloop2

&setvar cover [translate %cov%]
&setvar arc %cover%

/* -Identify the temporary cover and the cover type to be used-E
&if %cov% eq %c1% &then &setvar item %i1%
&if %cov% eq %c1% &then &setvar tip %t1%
&if %cov% eq %c2% &then &setvar item %i2%
&if %cov% eq %c2% &then &setvar tip %t2%
```

```
&if %cov% eq %c3% &then &setvar item %i3%
&if %cov% eq %c3% &then &setvar tip %t3%
&if %cov% eq %c4% &then &setvar item %i4%
&if %cov% eq %c4% &then &setvar tip %t4%
&if %cov% eq %c5% &then &setvar item %i5%
&if %cov% eq %c5% &then &setvar tip %t5%
&if %cov% eq %c6% &then &setvar item %i6%
&if %cov% eq %c6% &then &setvar tip %t6%
&if %cov% eq %c7% &then &setvar item %i7%
&if %cov% eq %c7% &then &setvar tip %t7%
&if %cov% eq %c8% &then &setvar item %i8%
&if %cov% eq %c8% &then &setvar tip %t8%
&if %cov% eq %c9% &then &setvar item %i9%
&if %cov% eq %c9% &then &setvar tip %t9%
&if %cov% eq %c10% &then &setvar item %i10%
&if %cov% eq %c10% &then &setvar tip %t10%

&setvar typ [translate %tip%]
&setvar cover [translate %cov%]
&setvar archive [substr %cover% 1 3].A
&if [exists %archive%] &then &goto usearch
&setvar archive %cover%

&label usearch
/*  -Create buffers, identify them, and apply them to the NCD-based cover-D
&run buffnshine %archive% %typ% %tol2% %item%
&setvar filunit [open CODERR openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy
&setvar arcbuf %archive%.BUF
&run identi2 poly %nodecrd% point %arcbuf% %item%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy

&severity &error &ignore
&setvar nodtab %nodecrd%.PAT
dropitem %nodtab% %nodtab% CAESER#
dropitem %nodtab% %nodtab% CAESER-ID
&setvar dummy1 %arcbuf%#
&setvar dummy2 %arcbuf%-ID
dropitem %nodtab% %nodtab% %dummy1%
dropitem %nodtab% %nodtab% %dummy2%

/*  If you want the buffer automatically removed, make the next line active-Y
/*  kill %arcbuf% all

&severity &error &fail
&label endloop2
&end
```

```
dropitem %nodtab% %nodtab% inside
&goto endit

&label badentry
&type Usage:  IDENTILOTS <point cover to be identified by multiple others
&type                (existing)><minimum expected dist between points>
&goto endit

&label bombout
&type 'An error has occurred'
&setvar i [close %filunit%]

&label endit
remepf -prg -na -nq -nvfy
&type End of IDENTILOTS
```

KITSINK.AML

Description

This macro is a shell used to run several other AML programs (fig. 34). It performs approximately the last half of the process of final mesh creation, including optimization of the finite-element node numbering and model file building. Given a final model boundary, a final regularly spaced grid, *SPLINE*d input coverages, and several pieces of "housekeeping" information, KITSINK.AML can complete the process, freeing a large block of time for the user. Because KITSINK.AML uses SNAPPY.AML, however, the user must be present during this part of the AML program (See SNAPPY.AML). Fortunately, this macro is run relatively early in KITSINK.AML, thereby leaving the user free for the bulk of the run time.

KITSINK.AML creates and uses a file called "CODERR" to indicate errors that occur during its sub-AML's. This file is for internal use and need not concern the user. KITSINK.AML's output consists of three files, each with the same user-specified root name and its own extension. The extensions are .ELMS for the finite-element mesh polygons, .ELPT for the polygon label cover that has element numbers, and .NOD for the point cover of the nodes of the mesh.

The model boundary used in KITSINK should be built (*BUILD*) both as a line cover and as a polygon cover. KITSINK will, at different times, look for both a PAT and an AAT for the model boundary cover.

At the beginning of its run, KITSINK.AML *KILL*s any copies from previous runs of files that it creates. These include: the master feature cover, the "allpoints" cover, the master point cover, and the master polygon cover. If together, the input feature coverages create approximately 10,000 points or more, then KITSINK.AML may crash during SNAPPY.AML. This is because SNAPPY.AML uses the *BUFFER* command and attempting to create that many circular buffer
areas is beyond the capability of ARC. If this occurs, the remainder of KITSINK.AML may have to be executed separately.

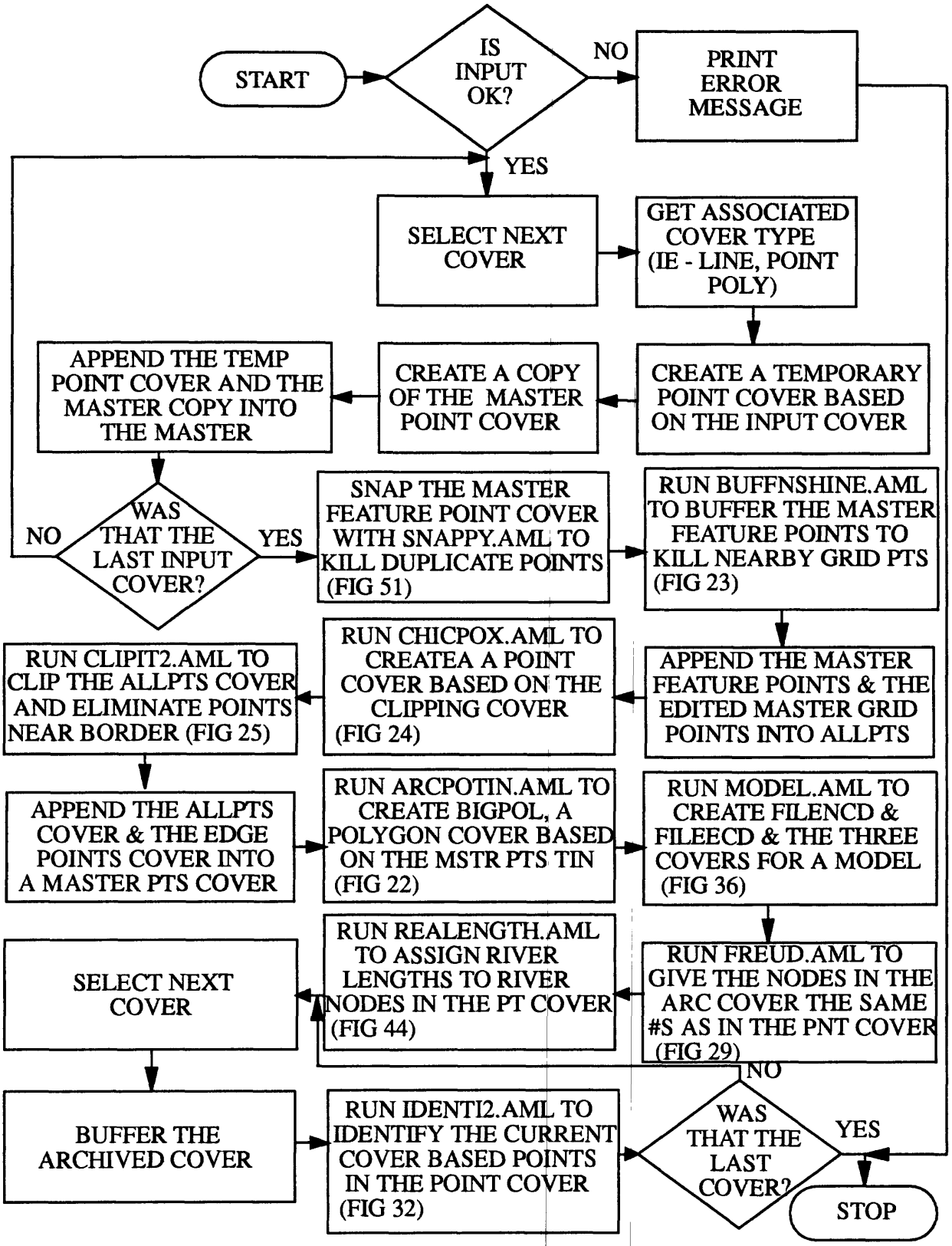


Figure 34.--Flowchart for KITSINK.AML.

```
/* MACRO: Create the output modelling files and ARC coverages based upon
/*            the clipped input line and point coverages, and given the desired
/*            triangular grid.
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*     DTYPE: The display type
/*  MSTFEAT: The master feature point cover
/*     SKALE: The map scale, in feet, of the input coverages
/*    BUFDIS: The minimum distance between feature and grid points. Also,
/*              the snap distance.
/*  MSTGRID: The master triangular grid
/*    ALLPTS: The cover containing all feature and all grid points
/*   MSTCUT: The final clipping cover, the model boundary
/*    EDGDIS: The minimum distance between interior and border points
/*    MSTPTS: The master point cover
/*    MSTPOL: The master polygonal, TINned cover
/*    MAXITR: The maximum number of optimizing iterations
/*     C1-C10: The input line and point coverages
/*       I1-I10: The item to be associated with each cover to identify it
/*      T1-T10: The feature type of each cover
/* ALLPTSCL: The clipped version of ALLPTS
/*  MCUTPTS: The point cover based upon the master clipping cover
/*    MSTTIN: The TIN based upon the master point cover
/*       MESH: The output element cover
/*  MESHLAB: The output element-based point cover
/*  NODECRD: The Node Coordinate Data-based point cover
/*        COV: The cover currently being considered in a loop
/*     COVER: The capitalized version of COV
/*      TEMP: The temporary file associated with the current cover
/*         TIP: The feature type of the current cover
/*        TYP: The capitalized version of TIP
/*      ITEM: The item associated with the current cover
/*      TOL2: A distance based on TOLERANCE
/*  ARCHIVE: The archive files containing the nearly-original versions of
/*              the input coverages
/*   ARCBUF: The buffered version of ARCHIVE

&echo &off
&args dtype mstfeat skale bufdis mstgrd allpts mstcut itcut ~
 edgdis mstpts mstpol maxitr

/* -Test to see if all arguments are present as expected-C
&if [type %mstfeat%] ne 1 &then &goto badentry
&if [type %skale%] ne -1 &then &goto badentry
&if [type %bufdis%] ne -1 &then &goto badentry
&if [type %mstgrd%] ne 1 &then &goto badentry
&if [type %allpts%] ne 1 &then &goto badentry
&if [type %edgdis%] ne -1 &then &goto badentry
&if [type %mstpts%] ne 1 &then &goto badentry
&if [type %mstpol%] ne 1 &then &goto badentry
```

```
&if [type %maxitr%] ne -1 &then &goto badentry
&if [length %maxitr%] eq 0 &then &goto badentry

/*  -Input the rest of the necessary information-O
&s c1 "
&s c2 "
&s c3 "
&s c4 "
&s c5 "
&s c6 "
&s c7 "
&s c8 "
&s c9 "
&s c10 "

&setvar c1 [response 'Enter cover name']
&if [length %c1%] eq 0 &then &goto nocovs
&setvar t1 [response 'Enter cover type (line,point,poly)']
&setvar i1 [response 'Enter item representing cover']
&setvar c2 [response 'Enter cover name']
&if [length %c2%] eq 0 &then &goto endentry
&setvar t2 [response 'Enter cover type (line,point,poly)']
&setvar i2 [response 'Enter item representing cover']
&setvar c3 [response 'Enter cover name']
&if [length %c3%] eq 0 &then &goto endentry
&setvar t3 [response 'Enter cover type (line,point,poly)']
&setvar i3 [response 'Enter item representing cover']
&setvar c4 [response 'Enter cover name']
&if [length %c4%] eq 0 &then &goto endentry
&setvar t4 [response 'Enter cover type (line,point,poly)']
&setvar i4 [response 'Enter item representing cover']
&setvar c5 [response 'Enter cover name']
&if [length %c5%] eq 0 &then &goto endentry
&setvar t5 [response 'Enter cover type (line,point,poly)']
&setvar i5 [response 'Enter item representing cover']
&setvar c6 [response 'Enter cover name']
&if [length %c6%] eq 0 &then &goto endentry
&setvar t6 [response 'Enter cover type (line,point,poly)']
&setvar i6 [response 'Enter item representing cover']
&setvar c7 [response 'Enter cover name']
&if [length %c7%] eq 0 &then &goto endentry
&setvar t7 [response 'Enter cover type (line point,poly)']
&setvar i7 [response 'Enter item representing cover']
&setvar c8 [response 'Enter cover name']
&if [length %c8%] eq 0 &then &goto endentry
&setvar t8 [response 'Enter cover type (line point,poly)']
&setvar i8 [response 'Enter item representing cover']
&setvar c9 [response 'Enter cover name']
&if [length %c9%] eq 0 &then &goto endentry
&setvar t9 [response 'Enter cover type (line,point,poly)']
&setvar i9 [response 'Enter item representing cover']
&setvar c10 [response 'Enter cover name']
&if [length %c10%] eq 0 &then &goto endentry
&setvar t10 [response 'Enter cover type (line,point,poly)']
```

```
&setvar i10 [response 'Enter item representing cover']
&goto endentry
&label nocovs
&type 'At least one cover must be entered'
&goto endit
&label endentry

&setvar strm [response 'Enter the name of the stream cover (or null)']
&if [length %strm%] eq 0 &then &goto zero
&goto sethem
&label zero
&setvar strm ''

/* -Initialize variables-M
&label sethem
&setvar allptscl %allpts%.CL
&setvar mcutpts %mstcut%.pts
&setvar msttin %mstpts%.TIN
&setvar mesh %mstpol%.ELMS
&setvar meshlab %mstpol%.ELPT
&setvar nodecrd %mstpol%.NOD
&setvar grdcut %mstgrd%.CL
&setvar chicdist %skale% / 500

/* -Eliminate old occurances of output files-E
&severity &error &ignore
kill %mstfeat% all
kill %allpts% all
kill %mcutpts% all
kill %mstpts% all
kill %mstpol% all
&severity &error &fail

/* -Generate a master feature point cover from each of the input coverages-D
&do cov &list %c1% %c2% %c3% %c4% %c5% %c6% %c7% %c8% %c9% %c10%
&if [length %cov%] eq 0 &then &goto endloop1

&setvar cover [translate %cov%]

/* -Identify the temporary cover and the cover type to be used-Y
&if %cov% eq %c1% &then &setvar temp temp1
&if %cov% eq %c1% &then &setvar tip %t1%
&if %cov% eq %c2% &then &setvar temp temp2
&if %cov% eq %c2% &then &setvar tip %t2%
&if %cov% eq %c3% &then &setvar temp temp3
&if %cov% eq %c3% &then &setvar tip %t3%
&if %cov% eq %c4% &then &setvar temp temp4
&if %cov% eq %c4% &then &setvar tip %t4%
&if %cov% eq %c5% &then &setvar temp temp5
&if %cov% eq %c5% &then &setvar tip %t5%
&if %cov% eq %c6% &then &setvar temp temp6
&if %cov% eq %c6% &then &setvar tip %t6%
&if %cov% eq %c7% &then &setvar temp temp7
&if %cov% eq %c7% &then &setvar tip %t7%
```

```
&if %cov% eq %c8% &then &setvar temp temp8
&if %cov% eq %c8% &then &setvar tip %t8%
&if %cov% eq %c9% &then &setvar temp temp9
&if %cov% eq %c9% &then &setvar tip %t9%
&if %cov% eq %c10% &then &setvar temp temp10
&if %cov% eq %c10% &then &setvar tip %t10%

&setvar typ [translate %tip%]

/*  -Create point coverages and append them-
&severity &error &ignore
kill %temp% all
&severity &error &fail
&if %typ% eq 'POINT' &then &goto pointit
&if %typ% eq 'POLY' &then build %cover% line
&run arcpotin arcpoint %cover% %temp% %typ%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy
&goto appendit
&label pointit
copy %cover% %temp%
&label appendit
&if %cov% ne %c1% &then &goto nthrun
copy %temp% %mstfeat%
kill %temp% all
&goto endloop1
&label nthrun
append dummy
%mstfeat%
%temp%
end
kill %mstfeat% all
copy dummy %mstfeat%
kill dummy all
kill %temp% all
&label endloop1
&end
build %mstfeat% point
remepf -prg -na -nq -nvfy

/*  -Snap the master point cover to eliminate points which are too close-H
&run snappy %dtype% %mstfeat% %skale% %bufdis%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy
&if [length %strm%] eq 0 &then &goto elimgrd
snapcover point %mstfeat% arc %strm% # %bufdis%

/*  -Create buffers around each of the master feature points to eliminate
```

```
/*  grid points which lie too close-O
&label elimgrd
&run buffnshine %mstfeat% point %bufdis% proximity %mstgrd% point %dtype%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&setvar i [read %filunit% %cod%]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy

/*  -Append the master feature points and the master grid points-U
append %allpts%
%mstfeat%
%grdcut%
end
build %allpts% point
remepf -prg -na -nq -nvfy

/*  -Create edge points based on the final cutting template-R
&run chicpox %dtype% %mstcut% %chicdist% %mcutpts%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy

/*  -Clip the all points file, eliminating points too close to the edge-
&run clipit2 itemb %dtype% %mstcut% %itcut% %edgdis% point %allpts% in
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy

/*  -Append the all points file and the edge points into the master points-S
&r fixsnap %dtype% %allptscl% %bufdis%
append %mstpts%
%allptscl%
%mcutpts%
end
build %mstpts% point
remepf -prg -na -nq -nvfy

/*  -Create a TIN-based polygon from the master point file-O
&severity &error &ignore
kill bigpol all
kill bigpol.bak all
kill mstpol.bak all
&severity &error &fail
&run arcpotin arctin %mstpts% %msttin% point bigpol
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
```

```
remepf -prg -na -nq -nvfy
copy bigpol bigpol.bak
remepf -prg -na -nq -nvfy
clip bigpol %mstcut% %mstpol%
remepf -prg -na -nq -nvfy
copy %mstpol% mstpol.bak
remepf -prg -na -nq -nvfy
kill bigpol all
remepf -prg -na -nq -nvfy
build %mstpol% poly
remepf -prg -na -nq -nvfy

/* -Create the files needed for modelling-F
&run model %mstpol% %maxitr% modata.prt ~
 %mesh% %meshlab% %nodecrd%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&setvar i [close %filunit%]
&if [length %cod%] eq 1 &then &goto bombout
remepf -prg -na -nq -nvfy

/* -Give the nodes in the arc file the same numbers as in the node file-T
&run freud %nodecrd% %mesh%
remepf -prg -na -nq -nvfy
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&setvar i [close %filunit%]
&if [length %cod%] eq 1 &then &goto bombout

/* -Assign lengths of stream to the points defining streams-W
&if [length %strm%] eq 0 &then &goto nostream
&run realength %dtype% %strm% %bufdis% %nodecrd%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy
&label nostream

/* -Identify the output node cover by each of the input feature coverages-A
/*      *       *       *       *       *       *

/* -Create buffers around each of the archived coverages-R
&setvar tol2 %bufdis% * 0.6
&do cov &list %c1% %c2% %c3% %c4% %c5% %c6% %c7% %c8% %c9% %c10%
&if [length %cov%] eq 0 &then &goto endloop2

&setvar cover [translate %cov%]
&setvar arc %cover%.A

/* -Identify the temporary cover and the cover type to be used-E
&if %cov% eq %c1% &then &setvar item %i1%
&if %cov% eq %c1% &then &setvar tip %t1%
&if %cov% eq %c2% &then &setvar item %i2%
```

```
&if %cov% eq %c2% &then &setvar tip %t2%
&if %cov% eq %c3% &then &setvar item %i3%
&if %cov% eq %c3% &then &setvar tip %t3%
&if %cov% eq %c4% &then &setvar item %i4%
&if %cov% eq %c4% &then &setvar tip %t4%
&if %cov% eq %c5% &then &setvar item %i5%
&if %cov% eq %c5% &then &setvar tip %t5%
&if %cov% eq %c6% &then &setvar item %i6%
&if %cov% eq %c6% &then &setvar tip %t6%
&if %cov% eq %c7% &then &setvar item %i7%
&if %cov% eq %c7% &then &setvar tip %t7%
&if %cov% eq %c8% &then &setvar item %i8%
&if %cov% eq %c8% &then &setvar tip %t8%
&if %cov% eq %c9% &then &setvar item %i9%
&if %cov% eq %c9% &then &setvar tip %t9%
&if %cov% eq %c10% &then &setvar item %i10%
&if %cov% eq %c10% &then &setvar tip %t10%

&setvar typ [translate %tip%]
&setvar cover [translate %cov%]
&setvar archive [substr %cover% 1 3].A

/* -Create buffers, identify them, and apply them to the NCD-based cover-
&run buffnshine %archive% %typ% %tol2% %item%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy
&setvar arcbuf %archive%.BUF
&run identi2 poly %nodecrd% point %arcbuf% %item%
&setvar filunit [open coderr openstatus -r]
&setvar cod [read %filunit% rdst]
&if [length %cod%] eq 1 &then &goto bombout
&setvar i [close %filunit%]
remepf -prg -na -nq -nvfy

&severity &error &ignore
&setvar nodtab %nodecrd%.PAT
dropitem %nodtab% %nodtab% CAESER#
dropitem %nodtab% %nodtab% CAESER-ID
&setvar dummy1 %arcbuf%#
&setvar dummy2 %arcbuf%-ID
dropitem %nodtab% %nodtab% %dummy1%
dropitem %nodtab% %nodtab% %dummy2%
kill %arcbuf% all
&severity &error &fail
&label endloop2
&end
dropitem %nodtab% %nodtab% inside
&goto endit

&label badentry
&type Usage:  KITSINK <display type> <master feature pnt cover (created)>
```

```
&type          <mapscale as if to plot on a 24" plotter (ft)>
&type          <min distance between grid & feature pts> <master grid
&type           (existing)> <cover w/ all points (created)>
&type          <study area model boundary poly cover name (existing)>
&type          <item name denoting area inside modl boundary(existing)>
&type          <min dist between interior and edge points>
&type          <master point cover (created)> <root name for output
&type           mesh and node coverages (created)> <max # of
&type           optimizing iterations>
&goto endit

&label bombout
&if %cod% eq 1 &then &type 'An error has occurred during ARCPOTIN'
&if %cod% eq 2 &then &type 'An error has occurred during BUFFNSHINE'
&if %cod% eq 3 &then &type 'An error has occurred during CHICPOX'
&if %cod% eq 4 &then &type 'An error has occurred during CLIPIT2'
&if %cod% eq 5 &then &type 'An error has occurred during IDENTI2'
&if %cod% eq 6 &then &type 'An error has occurred during MODEL'
&if %cod% eq 7 &then &type 'An error has occurred during FIXSNAP or SNAPPY'
&if %cod% eq 8 &then &type 'An error has occurred during FREUD'
&if %cod% eq 9 &then &type 'An error has occurred during REALENGTH'
&setvar i [close %filunit%]

&label endit
&s i [delete coderr]
remepf -prg -na -nq -nvfy
&type End of KITSINK
```

MAKOUTLIN.AML

Description

MAKOUTLIN.AML creates polygon "*CLIP*ping" coverages or polygon outlines of the type described in the subsection concerning CLIPIT.AML (fig. 35). The input cover for this macro is a polygon cover composed of one or several polygons. The output cover is a single polygon whose perimeter is that of the conglomerate of input polygons. MAKOUTLIN.AML creates an output file with an identifying item with a value of "1" for its interior area. It also, if desired, removes all of the interior lines from the input cover. If no interior line removal is desired, give the "#" character in place of the output polygon outline name. This option will add the identifying item to the input cover.

One common error in using MAKOUTLIN.AML is to use an input cover that has not been built (*BUILD*) as a polygon cover. An input cover must have a PAT.

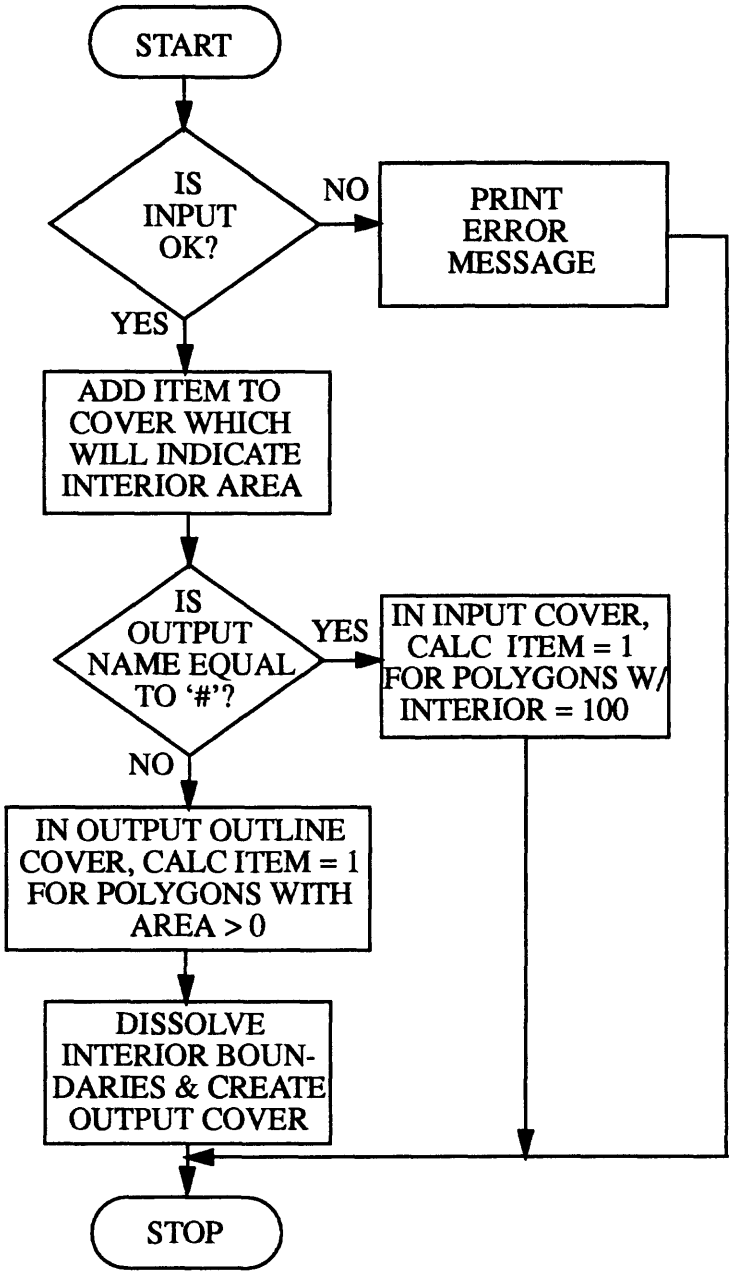


Figure 35.--Flowchart for MAKEOUTLIN.AML.

```
/* MACRO: Take a polygon cover and remove all of the internal lines,
/*              leaving a polygon outline which is written as an output cover
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST:
/*        COV: The name of the polygon cover from which the outline
/*               will be made
/*       FINE: The name of the outline to be created
/*         IT: The originally input name for the item indicating the
/*             area inside of the outline
/*      COVER: The capitalized version of COV
/*  FINEMESH: The capitalized version of FINE
/*       ITEM: The capitalized version of IT
/*  COVERPAT: The PAT name for cover
/*    FINEPAT: The PAT name for finemesh
/*      SHINY: Is the cover a buffer of another?

&echo &off
&args cov fine shiny it

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test input to see if all arguments are present as expected-O
&if [length %cov%] eq 0 &then &goto badentry
&if [length %fine%] eq 0 &then &goto badentry
&if [length %it%] eq 0 &then &goto badentry

/* -Capitalize input filenames for use in ARC/INFO-M
&setvar cover [translate %cov%]
&setvar finemesh [translate %fine%]
&setvar item [translate %it%]

/* -Reset output file-E
&if [exists %fine% -COVERAGE] &then kill %finemesh% all

/* -Prepare the input file for dissolution-D
&setvar coverpat %cover%.PAT
&severity &error &ignore
additem %coverpat% %coverpat% %item% 4 4 i
```

```
&severity &error &fail
&if %shiny% = 'y' &then &goto incheck
&if %shiny% = 'Y' &then &goto incheck
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SEL %coverpat%
RESEL FOR AREA > 0
CALC %item% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %coverpat%
RESEL FOR AREA > 0
CALC %item% = 1
Q STOP
QUIT
&end
&end
&goto checkdis

&label incheck
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SEL %coverpat%
RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %coverpat%
RESEL FOR INSIDE = 100
CALC %item% = 1
Q STOP
QUIT
&end
&end

&label checkdis
&if %finemesh% = '#' &then &goto endit

/* -Dissolve boundaries and build output as a polygon cover-Y
&label dodis
```

```
dissolve %cover% %finemesh% %item%
build %finemesh% poly

/* -Mark output polygon cover to indicate area within outline-
&setvar finepat %finemesh%.PAT
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SEL %finepat%
RESEL FOR AREA > 0
CALC %item% = 1
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SEL %finepat%
RESEL FOR AREA > 0
CALC %item% = 1
Q STOP
QUIT
&end
&end
dropitem %coverpat% %coverpat% %item%
createlabels %cover%
&goto endit

/* -Macro entry error handling-
&label badentry
&type Usage: MAKOUTLIN <input polygon cover(existing)><output polygon outline
&type        name (or '#' if not removing internal lines> <Is the
&type        in_cover a buffer of another cover? (y/n)> <name to
&type        designate the interior of the output cover (created)>

&label endit
&type End of MAKOUTLIN
```

MODEL.AML

Description

This macro takes an input polygon cover and creates the files that are necessary for modeling, the node coordinate data file, FILENCD, and the element connection data file, FILEECD. It also creates new coverages based on the element and node numbering optimization that it performs (fig. 36). It creates a mesh polygon cover having labels equal to the element numbers and having the suffix ".ELMS," a point cover having points at the center of each element labeled with the element numbers and having the suffix ".ELPT" and a point cover having points at nodes labeled with the node numbers and having the suffix ".NOD."

Once a polygon cover has been input to this macro, the output coverages should be used rather than the original cover. This is recommended because the output coverages now reflect the node and element numbering changes that MODEL.AML performs, and therefore correspond to the modeling files that are created.

MODEL.AML deletes files called "VERTICES," "FILENCD," and "FILEECD" at the beginning of its run. These are superceded copies of its working and output files.

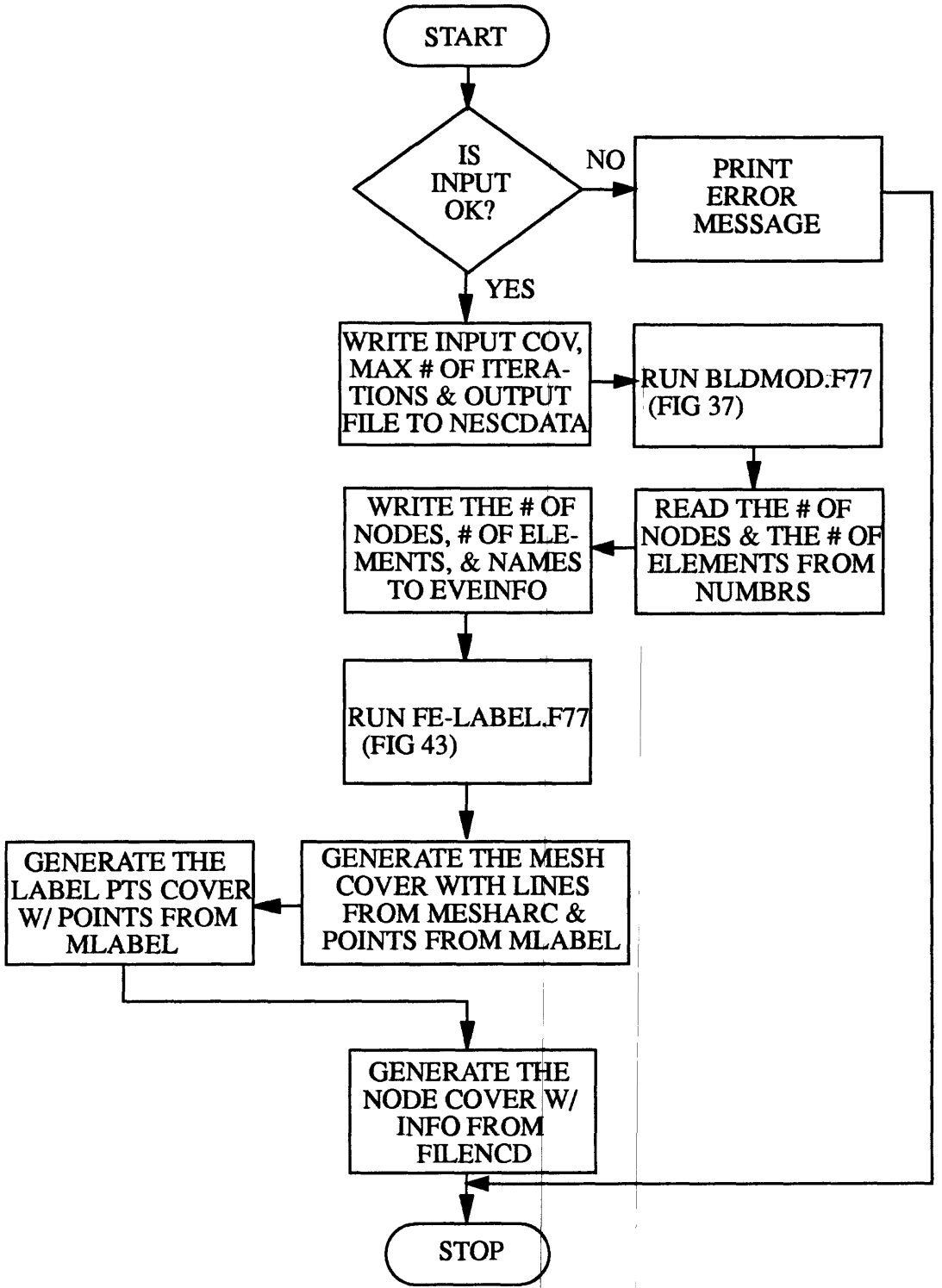


Figure 36.--Flowchart for MODEL.AML.

```
/* MACRO: Build the Node Coordinate Data file and the Element Connection
/*        Data file needed for modelling, based upon the input cover.
/*        Also, build a printable file detailing the optimization of the
/*        nodes performed, and build ARC files based on the re-labeled arc
/*        file, re-labeled label file, and the re-labeled node coordinate
/*        data file.
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST
/*    COVER: The input cover name
: The maximum number of optimizing iterations
/*    FPRINT: The name of the printable file to be created
/*    NNODE: The number of nodes in the input cover
/*    NELEM: The number of elements in the input cover
/*      MESH: The name of the mesh polygon cover with labels equal
/*            to the element number
/*  MESHLAB: The name of the point cover with point ID at the
/*            center of each element labeled with the element number
/*  NODECRD: The name of the point cover whose ID equals the
/*            node number
/*   FILUNIT: The unit number of the data file, EVEINFO
/*   EVEINFO: The data file used to pass data to FE-LABEL.F77
/*  MESHARC: The output element arc file from FE-LABEL.F77
/*    MLABEL: The output element node file from FE-LABEL.F77
/*  FE-LABEL: The Fortran77 program which creates the basis for the outputs

&echo &off
&args cover maxitr fprint mesh meshlab nodecrd

/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 6]
&setvar i [close %filun%]

/* -Check the computer type (by Leonard L. Orzol)-O
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test input to see if all arguments are present as expected-M
&if [type %cover%] ne 1 &then &goto badentry
```

```
&if [type %maxitr%] ne -1 &then &goto badentry
&if [type %fprint%] ne 1 &then &goto badentry
&if [type %mesh%] ne 1 &then &goto badentry
&if [type %meshlab%] ne 1 &then &goto badentry
&if [type %nodecrd%] ne 1 &then &goto badentry
&if [length %nodecrd%] eq 0 &then &goto badentry

/* -Delete any old occurances of the bldmod output files-E
&label beginit
&s i [delete filencd]
&s i [delete fileecd]
&s i [delete vertices]
&s i [delete badans]
&s i [delete nescdata]

/* -Build the file which bldmod needs-D
&setvar filunit [open nescdata openstatus -write]
&setvar i [write %filunit% %cover%]
&setvar i [write %filunit% %maxitr%]
&setvar i [write %filunit% %fprint%]
&if [close %filunit%] = 0 &then &type File written successfully.

&if %.computer_type% = 'prime' &then &sys r bldmod
&else &sys bldmod.out
&setvar filunit [open badans openstatus -read]
&setvar ans [read %filunit% readstatus]
&setvar i [close %filunit%]

&label go_on
/* -Read the file created by Optimize which is necessary to create the
/* output coverages-Y
&setvar filunit [open numbrs openstatus -r]
&setvar nnode [read %filunit% readstatus]
&setvar nelem [read %filunit% readstatus]
&setvar i [close %filunit%]

/* -Create the data file which FE-LABEL can read as input-
&setvar filunit [open eveinfo openstatus -w]
&setvar i [write %filunit% %nnode%]
&setvar i [write %filunit% %nelem%]
&setvar i [write %filunit% mesharc]
&setvar i [write %filunit% mlabel]
&if [close %filunit%] eq 0 &then &type File created successfully.

/* -Create the ASCII output files-H
&if %.computer_type% = 'prime' &then &sys r fe-label
&else &sys fe-label.out

/* -Delete the FE-LABEL input data file-O
&s i [delete eveinfo]

/* -Delete any old occurances of the output coverages-U
&severity &error &ignore
kill %mesh% all
```

```
kill %meshlab% all
kill %nodecrd% all
&severity &error &fail

/*  -Create the mesh polygon output cover-R
&if %.computer_type% = 'prime' &then
&do
generate %mesh%
input mesharc
line
input mlabel
point
q
&end
&else
&do
&data arc generate %mesh%
input mesharc
line
input mlabel
point
q
&end
&end
clean %mesh%
build %mesh% poly
build %mesh% line

/*  -Create the label point output cover-
&if %.computer_type% = 'prime' &then
&do
generate %meshlab%
input mlabel
point
q
&end
&else
&do
&data arc generate %meshlab%
input mlabel
point
q
&end
&end
build %meshlab% point

/*  -Delete the ASCII ouput files-
&s i [delete mesharc]
&s i [delete mlabel]

/*  -Create the node point output file-
&if %.computer_type% = 'prime' &then
&do
generate %nodecrd%
```

```
input filencd
&severity &error &ignore
point
&severity &error &fail
q
&end
&else
&do
&data arc generate %nodecrd%
input filencd
&severity &error &ignore
point
&severity &error &fail
q
&end
&end
build %nodecrd% point

&s i [delete nescdata]
&s i [delete numbrs]
&goto endit

&label badentry
&type Usage:  MODEL <name of point cover on which to base model(existing)><max
&type         # of optimizing iterations> <name of printed output file
&type         (created)> <output mesh polygon cover (created)> <output
&type         label point cover (created)> <output node point cover
&type         (created)>
&goto enderr

&label endit
/*  -Prepare the error-indication file-
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]
&goto bigend

&label enderr
&type There was an error!

&label bigend
&type End of MODEL
```

Fortran Program BLDMOD.F77

Description

Because the data structure of ARC/INFO does not store the information about polygons in the way that a finite-element model needs information about each element and the nodes that make up the elements, this program renumbers the elements and nodes and creates the data structure necessary for modeling (fig. 37). It is, in actuality, a shell for the three subroutines that compose it. It also calls the file that contains all of the global variable assignments for the programs, variables.

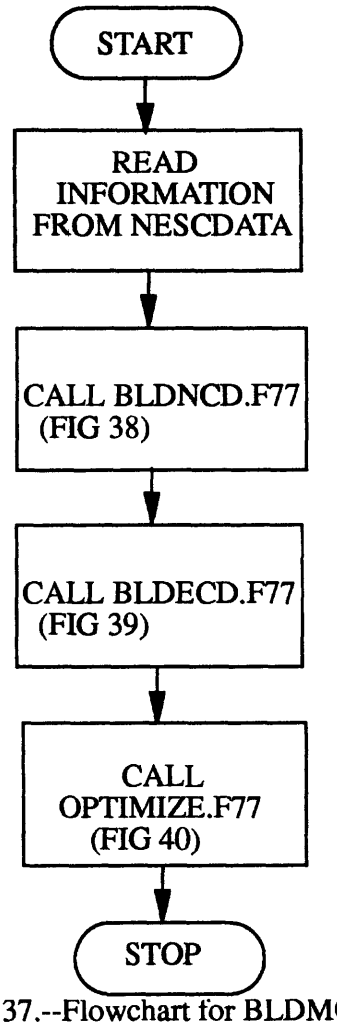


Figure 37.--Flowchart for BLDMOD.F77.

Program listing

```
C*************************************************************************C
C    PROGRAM: Build the output files needed for mesh modeling             C
C    CODED BY: Robert Lowther                                           H
C    SUPERVISED BY: Eve Kuniansky                                        S
C*************************************************************************C

      PROGRAM BLDMOD

C     VARIABLES contains the common statements and corresponding
C     variable definitions which are used by the subroutines.

      INCLUDE 'variables'

      CALL AENTER
      CALL LUNINI
      CALL MINIT
      CALL MESINI
```

```
      OPEN (7,file ='nescdata',status ='OLD',recl = 60)
100   FORMAT (A10)

      READ (7,100) COVER
      READ (7,*) MAXITR
      READ (7,100) FPRINT
      CLOSE (7)

C     BLDNCD obtains the keyboard input and reads the input file
C     into an input array. Manipulations are performed on the input
C     array and written into an output array. The output array
C     is then written to an output file, FILENCD.

      CALL BLDNCD
      IF (BADARC .NE. 0) GO TO 1
      PRINT *,'Building Element Connection Data file...'

C     BLDECD uses the input array from BLDNCD via the commons from
C     VARIABLES. Manipulations are performed on the input array
C     and written to an output array. The output array is then
C     written to an output file, FILEECD.

      CALL BLDECD
      PRINT *,'Optimizing the node numbering scheme...'

C     OPTIMIZE uses the previous two routines' output files as its
C     input. They are read into input arrays, where calculations
C     are performed on them. The output array is then written to
C     a user-specified output file.

      CALL OPTIMIZE

1     OPEN (8,file ='badans',status ='NEW',recl = 60)
      IF (BADARC .NE. 0) THEN
       PRINT *,'The TIN cover has vertices!'
       WRITE (8,100) 'BAD'
       ELSE
       WRITE (8,100) 'GOOD'
      END IF
      CLOSE (8)
      END

C     VARIABLES
C*************************************************************************C
C     VARIABLE LIST:                                                      C
C                                                                         O
C          COVER: The name of the input ARC cover                         C
C       NUMNOD: Number of nodes in the cover (also the number of          M
C                  lines in the FILENCD file)                             C
C         NUMEL: Number of elements in the cover (lines in FILEECD)       E
C         FPRINT: The name of the file that BLDMOD's output is to         C
C                  be written.                                            D
C        MAXITR: The maximum number of iterations that the optimi-        C
C                  zation program will run.                               Y
```

```
C      USERID: The USERIDs of the input arcs. (See ARC/INFO                C
C      FRNODE: The From Node for the input arc                             C
C      TONODE: The To Node for the input arc                               H
C        LPOLY: The polygon to the left of the input arc                   C
C        RPOLY: The polygon to the right of the input arc                  O
C         NPTS: The number of nodes associated with each input arc         C
C         NODE: The node number array for the output file, FILENCD         U
C       COORDS: The coordinate array read from the input ARC file          C
C    OUTCORD: The coordinate array for the output file, FILENCD            R
C      KANARC: The ARCFILE channel number                                  C
C      ACCESS: Code number to indicate a Read/Write ARC file               C
C      TEMPRY: Code number to indicate a Normal ARC file                   C
C       IERROR: Error code return from ARCRD command (-1 = error)          C
C       IABUFF: The standard ARC record, as read from the ARC file         C
C          NOIN: The number of input arcs (Length of input array)          C
C    ELEMENT: The element or polygon number in the output array            C
C          OUT1: The first node of each element in the output array        C
C          OUT2: (And OUT3) are defined similarly to OUT1                  C
C       FILENM: The output file name for BLDNCD and BLDECD                 C
C*********************************************************************C

      COMMON /MESH/COVER,NUMNOD,NUMEL,FPRINT,MAXITR,BADARC
      COMMON /C1/USERID(25000),FRNODE(25000)
      COMMON /C2/TONODE(25000),LPOLY(25000)
      COMMON /C3/RPOLY(25000),NPTS(25000)
      COMMON /C4/NODE(25000),NPT(25000)
      COMMON /C5/COORDS(25000,6)
      COMMON /C6/OUTCORD(25000,2)
      COMMON /C7/FILENM,KANARC,ACCESS,TEMPRY,IERROR
      COMMON /C8/IABUFF(2006),NOIN
      COMMON /C9/ELEMENT(25000),OUT1(25000)
      COMMON /C10/OUT2(25000),OUT3(25000)

      INTEGER IABUFF,NOIN
      INTEGER KANARC,ACCESS,TEMPRY,IERROR
      INTEGER NUMNOD,NUMEL,MAXITR,USERID,FRNODE
      INTEGER TONODE,LPOLY,RPOLY,NPTS,NODE
      INTEGER ELEMENT,OUT1,OUT2,OUT3,BADARC
      DOUBLE PRECISION COORDS,OUTCORD
      CHARACTER*128 COVER,FILENM,ARGS
      CHARACTER*60 FPRINT,BAD
```

Fortran subroutine BLDNCD.F77

Description.--This program builds the node coordinate data file necessary for mathematical modeling (fig. 38). As input, it uses the ARC file that is the final output of the mesh generation procedure. The node coordinate data file, FILENCD, consists of a list of the node numbers and coordinates of each node in the mesh.

To increase run speed, BLDNCD.F77 does all of its operations in memory. The input file is loaded into an input array and the output file is created as an output array. Once the program has completed building the output array, it is written to the output file. The tests we have performed have shown this process to be 9000% faster (decreasing run time from more than a day to approximately two minutes) than the more statically efficient method of manipulating the input and output files directly.

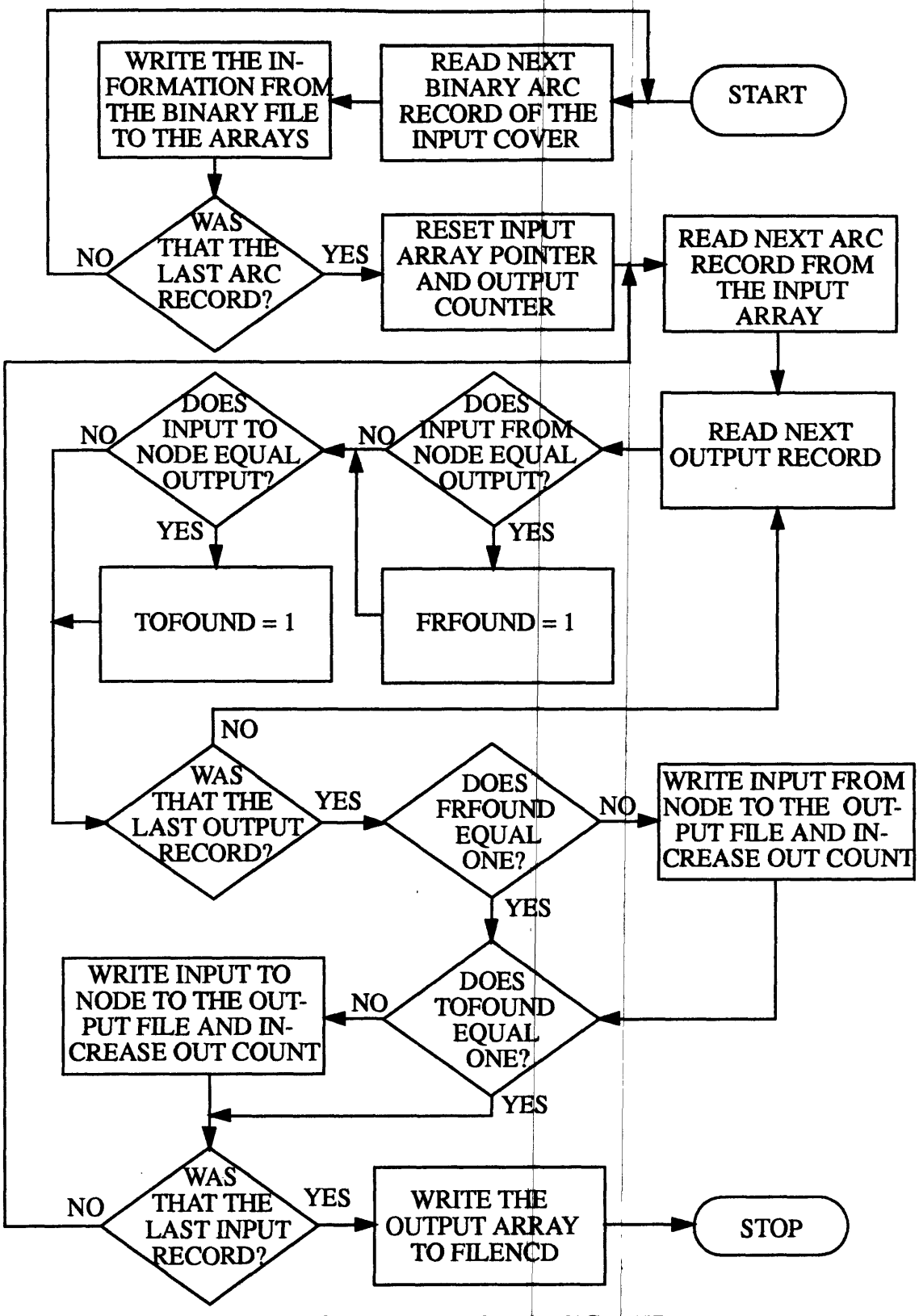


Figure 38.--Flowchart for BLDNCD.F77.

Subroutine listing.--

```
C*****************************************************************C
C     SUBROUTINE:  Build Node Coordinate Data File                C
C     CODED BY:  Robert Lowther                                   C
C     SUPERVISED BY:  Eve L. Kuniansky                            C
C*****************************************************************C
      SUBROUTINE BLDNCD

      INCLUDE 'variables'

C*****************************************************************C
C   VARIABLE LIST                                                 H
C                                                                 S
C         IREC:  The record number just read with the ARCRD command C
C          IER:  Error code for the AOPEN command                 C
C    FRFOUND:  Code showing FRNODE is already in the output array C
C    TOFOUND:  Code showing TONODE is already in the output array C
C          I,J:  Counters                                         C
C    OUTPCNT:  Number of records in the output array              C
C         LOUT:  The device number of the output file, FILENCD    H
C    USERDUM:  Integer equivalent to the first IABUFF component,  S
C                 it corresponds to the array USERID             C
C       FRDUM:  Corresponds to FRNODE                             C
C       TODUM:  Corresponds to TONODE                             C
C       LPDUM:  Corresponds to LPOLY                              C
C       RPDUM:  Corresponds to RPOLY                              C
C     NPTDUM:  Corresponds to NPTS                                C
C    CORDDUM:  Corresponds to COORDS                              H
C*****************************************************************S

      INTEGER IREC,IER,L2
      INTEGER FRFOUND,TOFOUND,I,J,OUTPCNT,LOUT
      INTEGER USERDUM,FRDUM,TODUM,LPDUM,RPDUM,NPTDUM
      DOUBLE PRECISION CORDDUM(1000)


C=====Use Equivalence statements to equate elements of the=======C
C    integer array, IABUFF, with integer and real variables     C
      EQUIVALENCE (IABUFF(1),USERDUM)
      EQUIVALENCE (IABUFF(2),FRDUM)
      EQUIVALENCE (IABUFF(3),TODUM)
      EQUIVALENCE (IABUFF(4),LPDUM)
      EQUIVALENCE (IABUFF(5),RPDUM)
      EQUIVALENCE (IABUFF(6),NPTDUM)
      EQUIVALENCE (IABUFF(7),CORDDUM(1))


      EXTERNAL LUNINI,MINIT,VINIT,ARCOPN,ARCRD,AOPEN,MESINI,
     &PRMSTR,ACLOSE,ARCCLS,AEXIT


100   FORMAT (6I6,4F15.2)
200   FORMAT (I6,2F15.2)
988   FORMAT (2F15.2)
```

```
986  FORMAT (I6)
     NUMNOD = 0

C=====Initialize the modules to be used with this program.========C

     CALL LUNINI
     CALL MINIT
     CALL VINIT
     CALL MESINI

C=====Open the input and output files for the NCD subroutine========C

     FILENM = 'filencd'
     ACCESS = 2
     TEMPRY = 1
     PRINT *,'Building the Node Coordinate Data file...'
     CALL ARCOPN (KANARC,COVER,ACCESS,TEMPRY,IERROR)
     OPEN (7,FILE= 'vertices')
     CALL AOPEN (LOUT,FILENM,IER)
     IF (IER .NE. -1) GO TO 99
      CALL ACREAT (LOUT,FILENM,IER)
      IF (IER .NE. -1) GO TO 99
       GO TO 16

C======Read the input file into input arrays, using the variables====C
C     from the equivalence statements                    C

C Initialize the input array size variable
99    NOIN = 1
10    CALL ARCRD (KANARC,IREC,IABUFF,IERROR)
      if (nptdum .gt. 2) print *,nptdum
      USERID(NOIN) = USERDUM
      FRNODE(NOIN) = FRDUM
      TONODE(NOIN) = TODUM
      LPOLY(NOIN) = LPDUM
      RPOLY(NOIN) = RPDUM
      NPT(NOIN) = NPTDUM
      DO 9 I = 1,NPTDUM*2
       COORDS(NOIN,I) = CORDDUM(I)
9     CONTINUE
C Increment the input array size
      NOIN = NOIN + 1
C If EOF reached, stop reading
      IF (IERROR .EQ. -1) GO TO 13
C If error occurs, report it
      IF (IERROR .EQ. -2) GO TO 15
       GO TO 10
C Adjust input array size variable correctly
13    NOIN = NOIN - 1
      IREC = NOIN

      I=1
      BADARC = 0
989   IF (NPT(I) .EQ. 2) GO TO 1001
```

```
      PRINT *,'The arc from:',FRNODE(I)
      print *,'to        :',TONODE(I)
      PRINT *,'has a vertex at:'
      DO 98 J=1,NPT(I)-2
       PRINT *,COORDS(I,J*2+1)
       PRINT *,COORDS(I,J*2+2)
98     CONTINUE
      DO 987 J=1,NPT(I)-1
       WRITE (7,986) 2*BADARC + J
       WRITE (7,988) COORDS(I,(J-1)*2+1), COORDS(I,(J-1)*2+2)
       WRITE (7,988) COORDS(I,J*2+1), COORDS(I,J*2+2)
987    CONTINUE
      BADARC = BADARC + 1
1001   I=I+1
     IF (I .LE. NOIN) GO TO 989

     IF (BADARC .NE. 0) PRINT *,'Vertices were found!  Bailing out
    c of BLDNCD.  Please wait.'
     IF (BADARC .NE. 0) GO TO 20
C======Read the input array line by line, each time checking to see===C
C     if the from and to nodes are already in the output array and,  C
C     if not, write them and their coordinates to the output array   C

C  Initialize the output array size variable
      OUTPCNT = 0
      DO 14 I = 1,NOIN
C Init the "found in output array" flags
      FRFOUND = 0
      TOFOUND = 0
      DO 11 J = 1,OUTPCNT
C If found, set flags
       IF (FRNODE(I) .EQ. NODE(J)) FRFOUND = 1
       IF (TONODE(I) .EQ. NODE(J)) TOFOUND = 1
11     CONTINUE

C=====If "found in output array" flag not set, write the nodes=======C
C     in question                                          C

C Check/write from node
      IF (FRFOUND .EQ. 1) GO TO 12
C Increment output array size variable
       OUTPCNT = OUTPCNT + 1
C Write information
       NODE(OUTPCNT) = FRNODE(I)
       OUTCORD(OUTPCNT,1) = COORDS(I,1)
       OUTCORD(OUTPCNT,2) = COORDS(I,2)
C Increment number of nodes counter
       NUMNOD = NUMNOD + 1
C Check/write to node
12     IF (TOFOUND .EQ. 1) GO TO 14
       OUTPCNT = OUTPCNT + 1
       NODE(OUTPCNT) = TONODE(I)
       OUTCORD(OUTPCNT,1) = COORDS(I,3)
       OUTCORD(OUTPCNT,2) = COORDS(I,4)
```

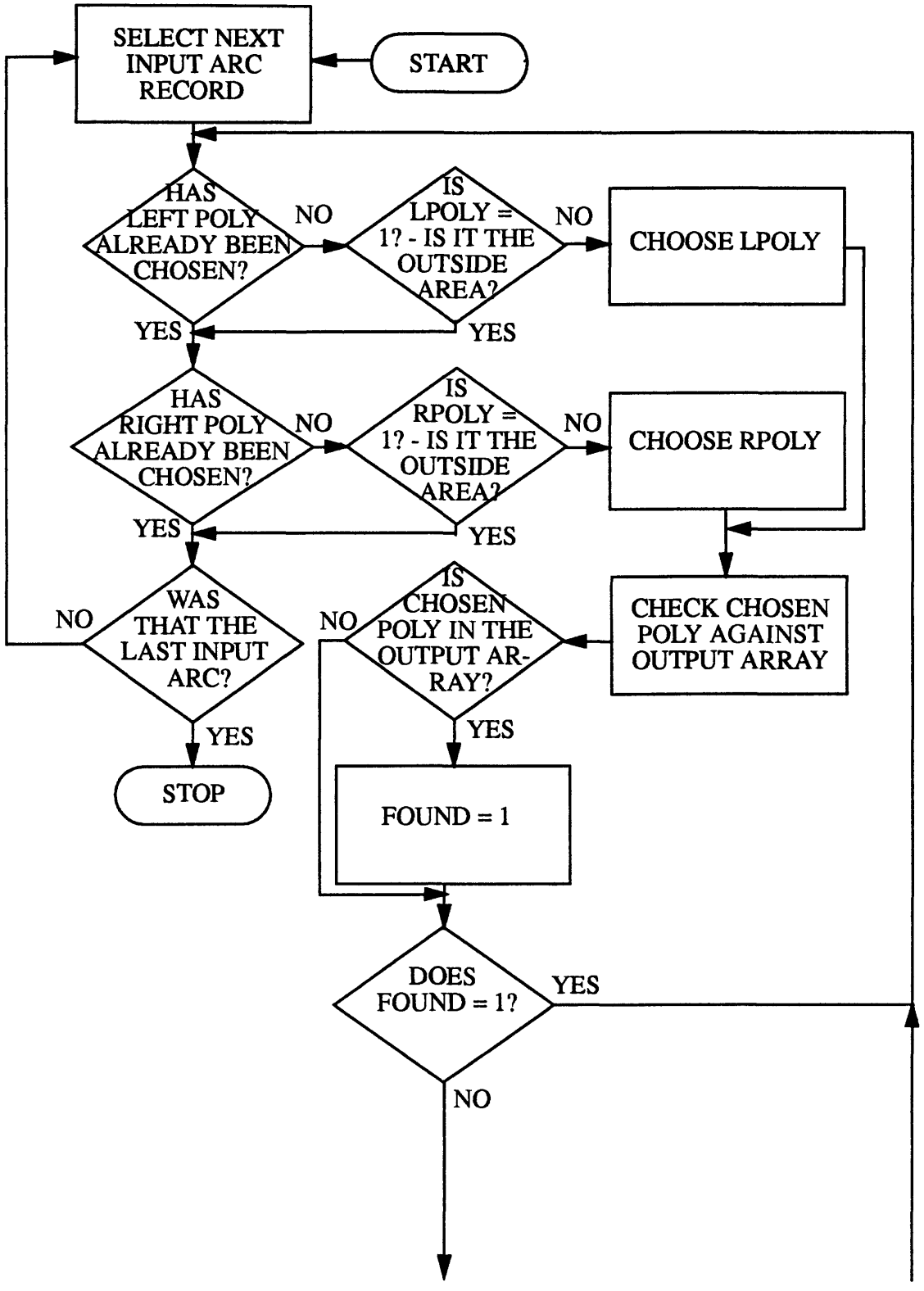
```
      NUMNOD = NUMNOD + 1
14    CONTINUE

C=====Write the output array to the output file=====================C

      DO 17 I=1,OUTPCNT
       WRITE (LOUT,200) NODE(I),(OUTCORD(I,J),J=1,2)
17    CONTINUE
      GO TO 20

C=====Error messages and program exit point=========================C

15    WRITE (*,'(A)') 'Error occurred during ARCRD'
      GO TO 20
16    WRITE (*,'(A)') 'Error occurred during AOPEN'
20    ENDFILE (LOUT)
      CALL ACLOSE (LOUT)
      CALL ARCCLS (KANARC)
      CLOSE (7)
      RETURN
      END
```

Fortran subroutine BLDECD.F77

Description.--This Fortran77 program builds the element connection data file, the other file necessary for mathematical modeling (fig. 39). It uses the same input as BLDNCD.F77. Its output file, FILEECD, consists of a list of the elements, or triangular polygon-ID's, and the node numbers of the three nodes associated with each element.

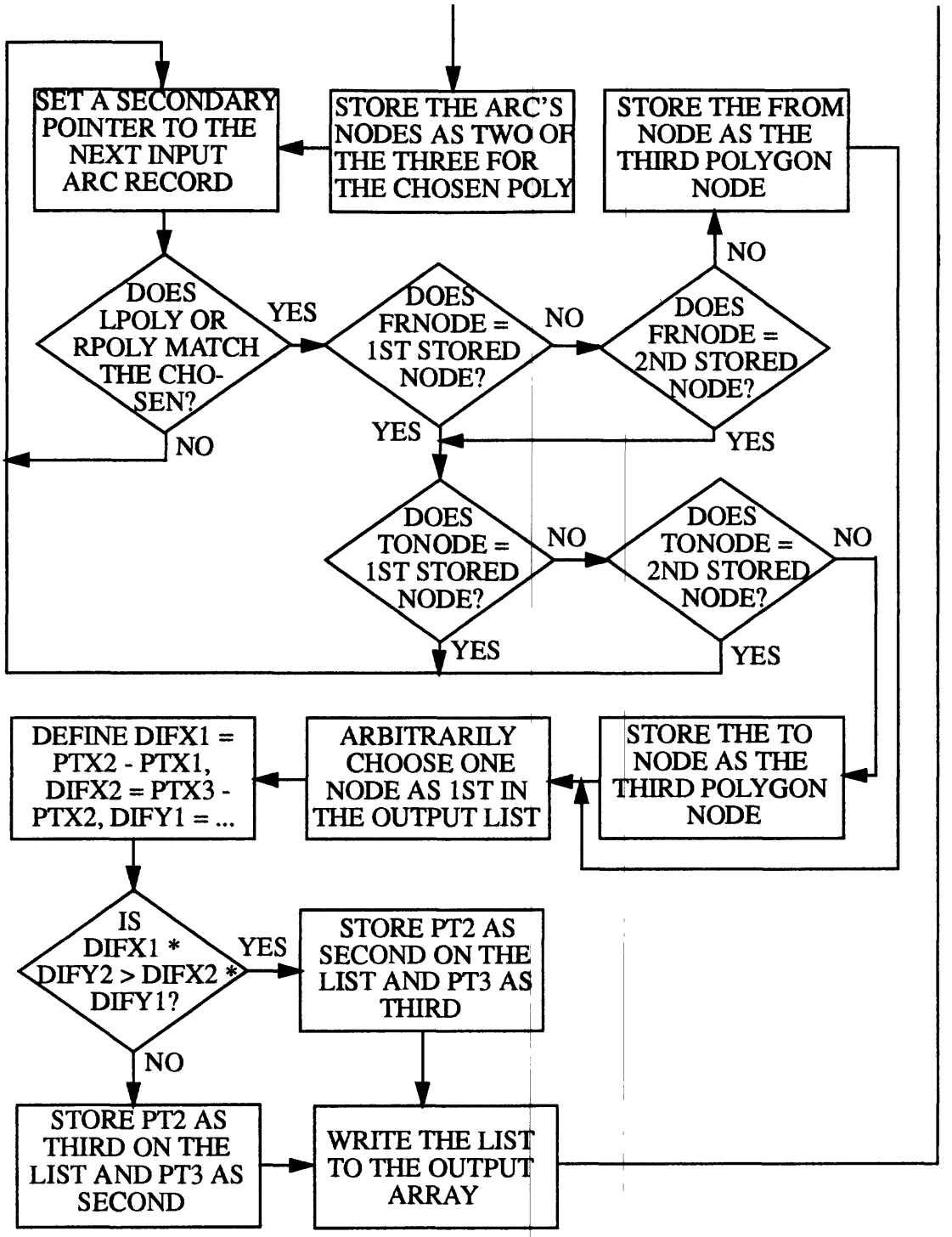
For the model to interpret this table correctly, the nodes listed for each element must be listed in counterclockwise order. To put the nodes in order, BLDECD.F77 creates two lines sharing a common point, that point being chosen arbitrarily and assigned as the first point in the list. The program then calculates the cross product of the two lines. Because the lines are in the same plane, their cross product will lie along a line perpendicular to that plane, either in the positive or negative direction. The sign of the cross product, therefore, determines which line, and accordingly, which endpoint, is counterclockwise from the other, referenced to the lines' common endpoint.

As in BLDNCD.F77, this program processes the data in arrays and writes the output file only after all calculations are finished. Again, this is done to increase dynamic efficiency.



a. logic which loops through all arcs and their associated polygons

Figure 39.--Flowchart for BLDECD.F77.



b. logic which orders the nodes associated with each element

Figure 39.--Flowchart for BLDECD.F77--continued.

Subroutine listing.--

```
C*********************************************************************C
C      PROGRAM:  Build Element Connection Data File                   C
C      CODED BY:  Robert Lowther                                      C
C      SUPERVISED BY:  Eve L. Kuniansky                               C
C*********************************************************************C

   SUBROUTINE BLDECD

   INCLUDE 'variables'

C*********************************************************************C
C   VARIABLE LIST                                                     C
C                                                                     O
C           IER:  Error code return from the AOPEN or ACREAT commands C
C           I,J:  Counters                                            M
C         FOUND:  Flag indicating that the element in question has    C
C                 been found in the output array                      C
C        CHECK1:  The first node of an element to be put into order   H
C  CHECK2 & 3:  The other two nodes to be ordered counter-clockwise   C
C     CHECKPO:  The polygon checked against output array and for      R
C                 which nodes are found                               C
C          DEF1:  Definitely the first node in order in the output    C
C       DEF2 & 3:  Definitely the second and third, counter-clockwise S
C     OUTPCNT:  The output array size variable                        C
C      CHECKX1:  The x coordinate of the first checked node (CHECK1)  O
C CHECKX2 & 3:  The x coordinate of the second and third nodes       C
C      CHECKY1:  Similarly, the first y coordinate                    F
C CHECKY2 & 3:  The second and third y coordinates                   C
C          DELX1:  The difference between x2 and x1                  T
C          DELX2:  x3 - x1                                           C
C          DELY1:  y2 - y1                                           W
C       DELY2:  y3 - y1                                              C
C*********************************************************************R

   INTEGER IER,I,J
   INTEGER FOUND,CHECK1,CHECK2,CHECK3,CHECKPO
   INTEGER DEF1,DEF2,DEF3,OUTPCNT
   DOUBLE PRECISION CHECKX1,CHECKX2,CHECKY1,CHECKY2
   DOUBLE PRECISION CHECKX3,CHECKY3
   DOUBLE PRECISION DELX1,DELX2,DELY1,DELY2

1000  FORMAT (6I6,4F15.1)
2000  FORMAT (4I6)
   NUMEL = 0

C=====Open the output file for the ECD subroutine============C

   FILENM = 'fileecd'
   CALL AOPEN (LOUT,FILENM,IER)
   IF (IER .NE. -1) GO TO 99
    CALL ACREAT (LOUT,FILENM,IER)
    IF (IER .NE. -1) GO TO 99
```

```
      GO TO 500

C=====Select two points, partially defining a polygon,=======C
C     from the input array.  Insure that the polygon        C
C     defined is not already listed in the output array.    C

C Initialize the output array size variable
99    OUTPCNT = 0
C For each input record: ...
      DO 14 I=1,NOIN

C-----Choose LPoly,RPoly, or next record

C Indicate neither L nor RPOLY has been checked
      CHECKPO = 0
C Don't check figure exterior
      IF (LPOLY(I) .EQ. 1) CHECKPO = LPOLY(I)
C If both have been checked, go on
15    IF (CHECKPO .EQ. RPOLY(I)) GO TO 14
C If LPOLY has, check R
      IF (CHECKPO .EQ. LPOLY(I)) CHECKPO = RPOLY(I)
      IF ((RPOLY(I) .EQ. 1) .AND. (CHECKPO .EQ. RPOLY(I))) GO TO 14
C If neither, check LPOLY
      IF (CHECKPO .EQ. 0) CHECKPO = LPOLY(I)
C Initialize the "found in output array" flag
      FOUND = 0
C======Check to see if polygon is already in output array

      DO 30 J=1,OUTPCNT
       IF (ELEMENT(J) .EQ. CHECKPO) FOUND = 1
30    CONTINUE

C=====If "found" flag is not set, select two nodes and their associated pts==C

      IF (FOUND .EQ. 1) GO TO 15
      CHECK1 = FRNODE(I)
      CHECKX1 = COORDS(I,1)
      CHECKY1 = COORDS(I,2)
      CHECK2 = TONODE(I)
      CHECKX2 = COORDS(I,3)
      CHECKY2 = COORDS(I,4)

C=====Find the third point defining the chosen polygon================C

C Set a pointer to the current location in the input array
      J=I
C Increment the secondary input array position indicator
50    J=J+1
C Find another reference to the chosen polygon
      IF ((LPOLY(J) .EQ. CHECKPO) .OR. (RPOLY(J) .EQ.
     & CHECKPO)) GO TO 60
       GO TO 50
C Check node against two known
60     IF (FRNODE(J) .NE. CHECK1) GO TO 70
```

```
      GO TO 80
70    IF (FRNODE(J) .NE. CHECK2) GO TO 100
80    IF (TONODE(J) .NE. CHECK1) GO TO 90
      GO TO 50
90    IF (TONODE(J) .NE. CHECK2) GO TO 110
      GO TO 50
C Choose from node which was not known before
100   CHECK3 = FRNODE(J)
C Get associated points
      CHECKX3 = COORDS(J,1)
      CHECKY3 = COORDS(J,2)
      GO TO 120
C Choose to node which was not known before
110   CHECK3 = TONODE(J)
      CHECKX3 = COORDS(J,3)
      CHECKY3 = COORDS(J,4)

C======Put the points in counter-clockwise order=========================C

C Arbitrarily choose a point to be first
120   DEF1 = CHECK1
C Define the difference variables
      DELX1 = CHECKX2 - CHECKX1
      DELY1 = CHECKY2 - CHECKY1
      DELX2 = CHECKX3 - CHECKX1
      DELY2 = CHECKY3 - CHECKY1


C------Take the cross product of the two vectors created by using the first
C     point as an endpoint to each and the other two points as endpoints
C     to their respective vectors. If the cross product is negative, then
C     the vector which was treated as the "first" vector should in fact be
C     second, and vice-versa. If the cross product is positive, then the
C     vector assignments are correct. The points are ordered according to
C     this determination.

      IF ((DELX1 * DELY2) .GT. (DELX2 * DELY1)) GO TO 130
      DEF2 = CHECK3
      DEF3 = CHECK2
      GO TO 140
130   DEF2 = CHECK2
      DEF3 = CHECK3

C======Write to the output array=========================================C

C Increment the output array size variable
140   OUTPCNT = OUTPCNT + 1
      ELEMENT(OUTPCNT) = CHECKPO
      OUT1(OUTPCNT) = DEF1
      OUT2(OUTPCNT) = DEF2
      OUT3(OUTPCNT) = DEF3
C Increment the number of elements counter
      NUMEL = NUMEL + 1
      GO TO 15
```

```
C Go to the next input array polygon
14   CONTINUE

C=====Write the ouput array into the output file, reassigning the element==C
C=====numbers to be sequential in the file, since the original numbers are=C
C=====no longer necessary=============================================C

     DO 17 I=1,OUTPCNT
      WRITE (LOUT,2000) I,OUT1(I),OUT2(I),OUT3(I)
17   CONTINUE
     GO TO 510

C=====Error messages and program exit point=============C

500  WRITE (*,'(A)') 'Error occurred during AOPEN'
     GO TO 510
20   WRITE (*,'(A)') 'Error occurred during ARCRD'
510  READ (LOUT,2000,END=520) IA,IB,IC,ID
     GO TO 510
520  ENDFILE (LOUT)
     CALL ACLOSE (LOUT)
     RETURN
     END
```

Fortran subroutine OPTIMIZE.F77

Description.--This algorithm was developed by R. J. Collins (1973), and the program that we modified was written by M. L. Maslia (U. S. Geological Survey, written commun., 1987). Originally a stand-alone program, it has been further modified to run as a subroutine of BLDMOD.F77 (fig. 40).

It takes the files created by BLDNCD.F77 and BLDECD.F77 and optimizes the node numbering scheme. It reduces the maximum difference between node numbers associated with any given element, thereby reducing the matrix bandwidth. This greatly reduces roundoff error in the model. SETUP (fig. 41) and OPTNUM (fig. 42) are two subroutines for OPTIMIZE.F77. SETUP "sets up" a particular numbering system and OPTNUM optimizes it. The resulting matrix size of this numbering system is compared to the matrix size of the previous systems. If the new system requires a smaller matrix, then it is stored for later comparisons.

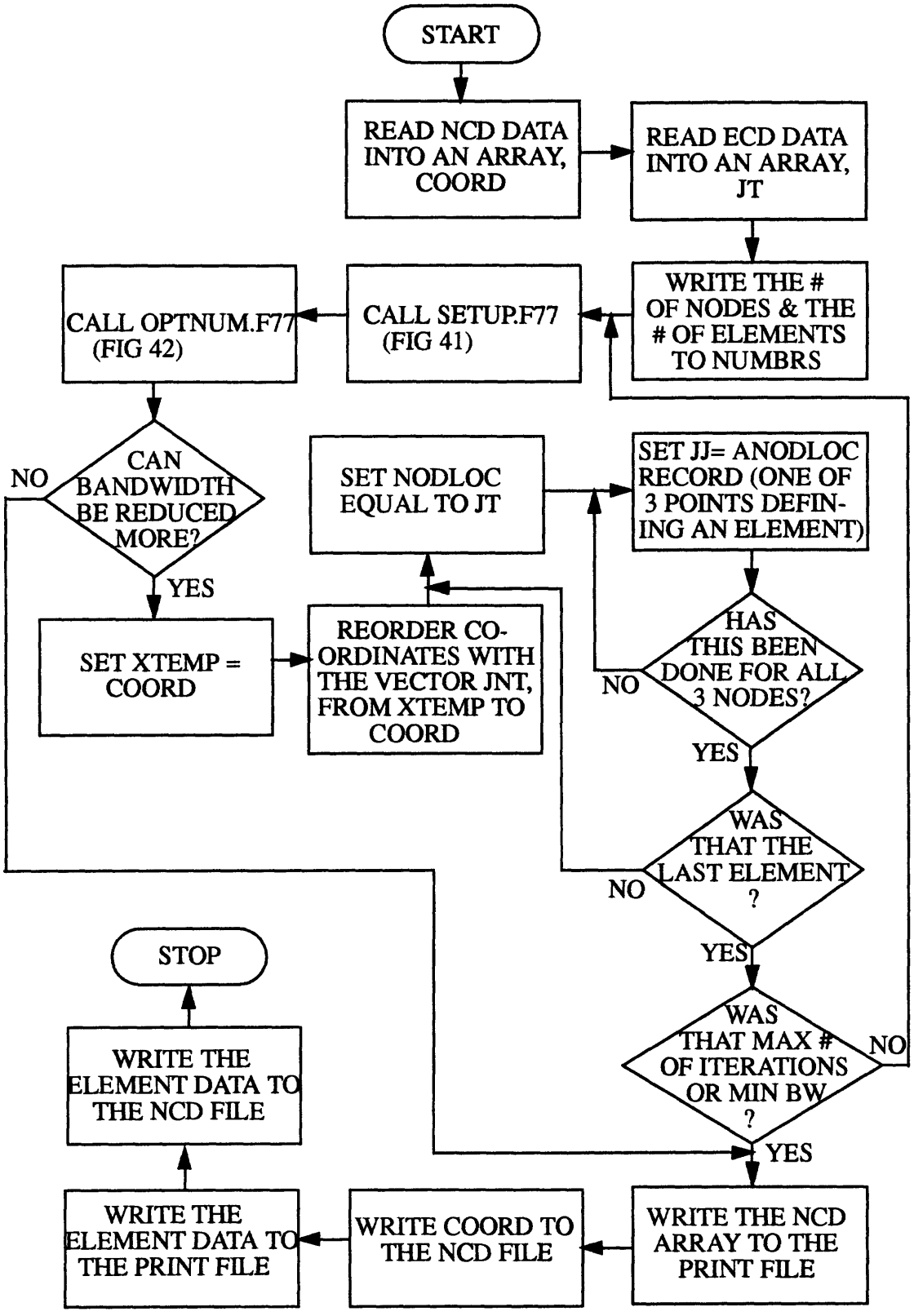


Figure 40.--Flowchart for OPTIMIZE.F77.

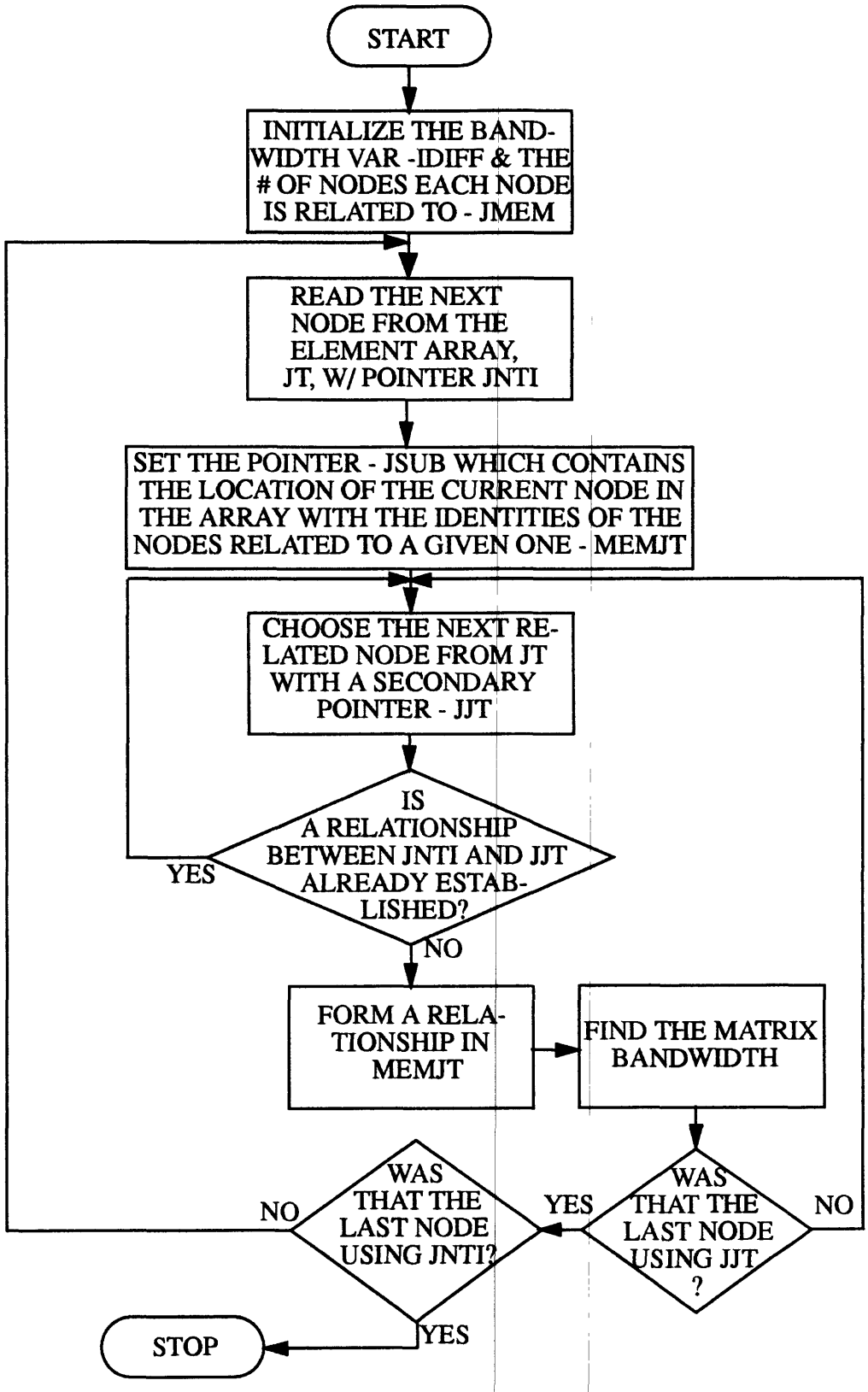


Figure 41.--Flowchart for SETUP.F77.

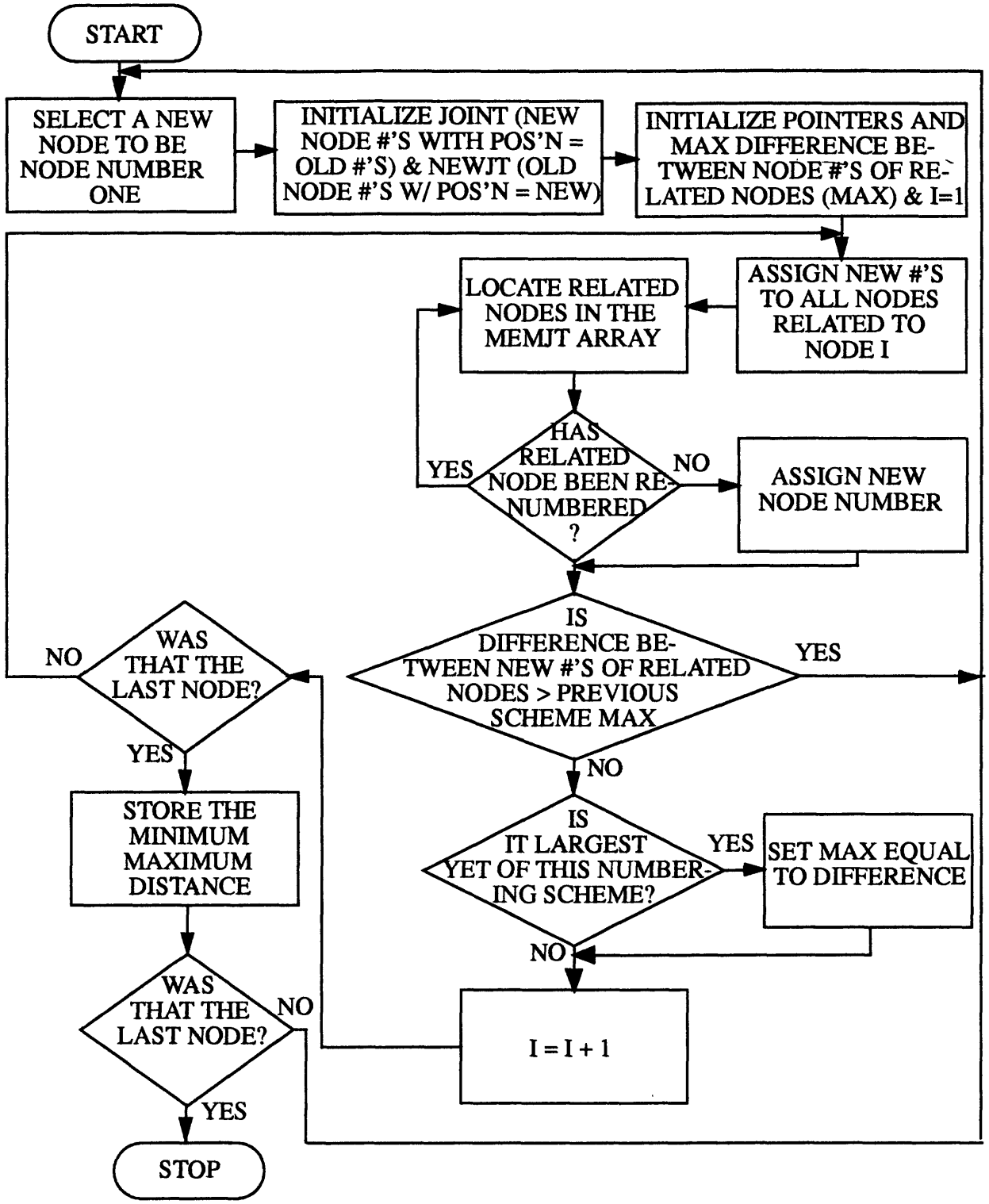


Figure 42.--Flowchart for OPTNUM.F77.

Subroutine listing.--

```
C*************************************************************************C
C    PROGRAM: Optimize the node numbering system of a mesh by minimizing      C
C                  the difference between the node numbers of associated NUMNOD  C
C    PUBLISHED AS: Collins, R.J., 1973                                        C
C    MODIFIED BY: Morris L. Maslia, The United States Geological Survey       C
C    SECONDARY MODIFICATION BY: Robert Lowther, same                          C
C    SUPERVISION OF SECONDARY MODIFICATION BY: Eve L. Kuniansky, same         C
C*************************************************************************C

      SUBROUTINE OPTIMIZE

      INCLUDE 'variables'

      common/bzone/jnt(25000),xtemp(25000,2),nodloc(1,3)
      common/acsolv/jmem(25000),memjt(225000)
      common /gener/coord(25000,3),JT(25000,12),matno(25000),
     &shape(12),NMNOD,NMEL,nnode,ndime
      character*60 fread,FREAD1
      data ncr,npr,NCR1/50,60,70/

c-------------------------------------------------------------------------c
c            renumber NUMDOD to obtain optimal bandwidth     c
c-------------------------------------------------------------------------c

      FREAD = 'filencd'
      FREAD1 = 'fileecd'
      NMNOD = NUMNOD
      NMEL = NUMEL
      open (unit=ncr,file=fread)
      OPEN (UNIT=NCR1,FILE=FREAD1)
      open (unit=npr,file=fprint)
      open (unit=30,file='numbrs')

c---------- read coordinate file and element connection file ----------c
      do 10 i=1,NMNOD
       read (ncr,*) j,coord(j,1),coord(j,2)
   10 continue
      do 30 i=1,NMEL
       read (NCR1,*) K,(JT(K,j), j=1,3)
   30 continue

      WRITE (30,*) NMNOD
      WRITE (30,*) NMEL
      CLOSE (30)

      ncn = 3
      it = 0
      iend = 0
      WRITE(NPR,1070) NMNOD
 1070  FORMAT (/,10X,'The mesh has ',I6,' nodes.',/)
      WRITE(NPR,1071) NMEL
 1071  FORMAT (/,10X,'The mesh has ',I6,' elements.',/)
```

```
    do 9000 kkk = 1,maxitr
     it = it+1
     call setup(NMNOD,NMEL,ncn,IDIFF)
     call optnum(NMNOD,IDIFF,minmax,iend)
     write(npr,1072) it
1072  format(/,10x,'***** OPTIMIZATION ITERATION NO.'i3,' *****'//)
     if(iend .gt. 0) write(npr,1073)
1073  format(/,10x,'further bandwidth reduction not possible !!!!! '/)
     write(npr,1074)IDIFF,minmax
1074  format(/10x,'original bandwidth =',i10/
    1 10x,'bandwidth after renumbering =',i10)
     if(iend .gt. 0) go to 9500

     do 1005 i=1,NMNOD
      xtemp(i,1)=coord(i,1)
1005  xtemp(i,2)=coord(i,2)
     do 2005 i=1,NMNOD
      jr=jnt(i)
      coord(jr,1)=xtemp(i,1)
2005  coord(jr,2)=xtemp(i,2)
     do 3005 i=1,NMEL
      do 4005 j=1,ncn
4005   nodloc(1,j)=JT(i,j)
      do 5005 j=1,ncn
       jj=nodloc(1,j)
       if(jj .eq. 0) go to 5005
       jr=jnt(jj)
       JT(i,j)=jr
5005   continue
3005  continue

     if(it .eq. maxitr) go to 9500

9000 continue

9500 continue
    rewind(ncr)
    do 50 i=1,NMNOD
     write(npr,'(i5,2f15.3)') i,(coord(i,j),j=1,2)
     write(ncr,'(i6,2f15.2)') i,(coord(i,j),j=1,2)
 50 continue
    rewind(ncr1)
    do 60 i=1,NMEL
     write(npr,'(3i5)') (JT(i,j),j=1,3)
     write(ncr1,'(4i6)') i,(jt(i,j),j=1,3)
 60 continue

    CLOSE (NPR)
    CLOSE (NCR)
    CLOSE (NCR1)
    RETURN
    end
```

```
c*********************************************************************c
c                                                                     c
c              subroutine setup(NMNOD,NMEL,NCN,idiff)                 c
c                                                                     c
c                                                                     c
c*********************************************************************c
c                                                                     c
c   generates array memjt,jmem                                        c
c                                                                     c
c   idiff  =  maximum banwidth                                        c
c   jt     =  node connection matrix                                  c
c   jmem   =  vector containig the number of NUMDOD to which any one node  c
c             connected                                               c
c   memjt  =  vector containing identities of all NUMDOD              c
c                                                                     c
c---------------------------------------------------------------------c

    subroutine setup(NMNOD,NMEL,NCN,idiff)

    common/bzone/jnt(25000),xtemp(25000,2),nodloc(1,3)
    common/acsolv/jmem(25000),memjt(225000)
    common /gener/coord(25000,3),jt(25000,12),matno(25000),
   &shape(12),npdum,nedum,nndum,nddum

c---------------------------------------------------------------------c
c                  initialise idiff and jmem                          c
c---------------------------------------------------------------------c

    idiff=0
    do 10 j=1,NMNOD
  10 jmem(j)=0

c---------------------------------------------------------------------c
c                  consider each element in turn                      c
c---------------------------------------------------------------------c

    do 60 j=1,NMEL
     do 50 i=1,NCN
     jnti=jt(j,i)
     if(jnti.eq.0) go to 60
     jsub=(jnti-1)*8

     do 40 ii=1,NCN
      if(ii.eq.i) go to 40
      jjt=jt(j,ii)
      if(jjt.eq.0) go to 50
      mem1=jmem(jnti)
      if(mem1.eq.0) go to 30
      do 20 iii=1,mem1
       if(memjt(jsub+iii).eq.jjt) go to 40
  20   continue
  30   jmem(jnti)=jmem(jnti)+1
       jmemjn=jmem(jnti)
       memjt(jsub+jmemjn)=jjt
```

```
      if(iabs(jnti-jjt).gt.idiff) idiff=iabs(jnti-jjt)
   40 continue
   50 continue
   60 continue
      idiff=idiff+1
      return
      end

c*****************************************************************c
c                                                                 c
c              subroutine optnum(NMNOD,idiff,minmax)              c
c                                                                 c
c*****************************************************************c
c                                                                 c
c    obtains optimum node numbering vectors,jnt.                  c
c                                                                 c
c    jnt    =  optimum node numbering vector                      c
c    joint  =  vector containg the new node numbers, the locations in  c
c              this vector being equal to the old joint numbers     c
c    newjt  =  vector containg the old node numbers,their locations in  c
c              this vector being equal to the new joint numbers     c
c    i,ik   =  variables denoting new and old node numbers respectively  c
c                                                                 c
c-----------------------------------------------------------------c

      subroutine optnum(NMNOD,idiff,minmax,iend)

      common/bzone/jnt(25000),xtemp(25000,2),nodloc(1,3)
      common/acsolv/jmem(25000),memjt(225000)
      common/adpara/newjt(25000),joint(25000)
      common /gener/coord(25000,3),jt(25000,12),matno(25000),
     &shape(12),npdum,nedum,nndum,nddum
      data nprint/60/

      minmax=idiff-1
      iflag = 0
      do 60 ik=1,NMNOD

c-----------------------------------------------------------------c
c                       initialise joint,newjt                    c
c-----------------------------------------------------------------c

      do 20 j=1,NMNOD
       joint(j)=0
   20 newjt(j)=0
      max=0
      i=1
      newjt(1)=ik
      joint(ik)=1
      k=1
   30 jnw=newjt(i)
      if(jnw.ne.0) k4=jmem(jnw)
      if(jnw.eq.0) k4=0
      if(k4.eq.0) go to 45
```

```
      jsub=(newjt(i)-1)*8
      do 40 jj=1,k4
       k5=memjt(jsub+jj)
       if(joint(k5).gt.0) go to 40
       k=k+1
       newjt(k)=k5
       joint(k5)=k
       ndiff=iabs(i-k)
       if(ndiff.ge.minmax) go to 60
       if(ndiff.gt.max) max=ndiff
   40  continue
       if(k.eq.NMNOD) go to 50
   45  i=i+1
       go to 30
   50  minmax=max
       do 55 j=1,NMNOD
   55  jnt(j)=joint(j)
       iflag = iflag + 1
   60 continue
      minmax=minmax+1
      nnode=NMNOD
      if(iflag .eq. 0) iend = 1
      return
      end
```

Fortran Program FE-LABEL.F77

Description

This program uses FILENCD and FILEECD as input, thereby using the optimally renumbered mesh. From these files, it creates two output files, mesharc and mlabel (fig. 43). These output files are simple rearrangements of the input files. They are designed so that they can act as ASCII input files to the *GENERATE* command in ARC, thereby allowing the user to create ARC coverages based upon the optimally renumbered model mesh.

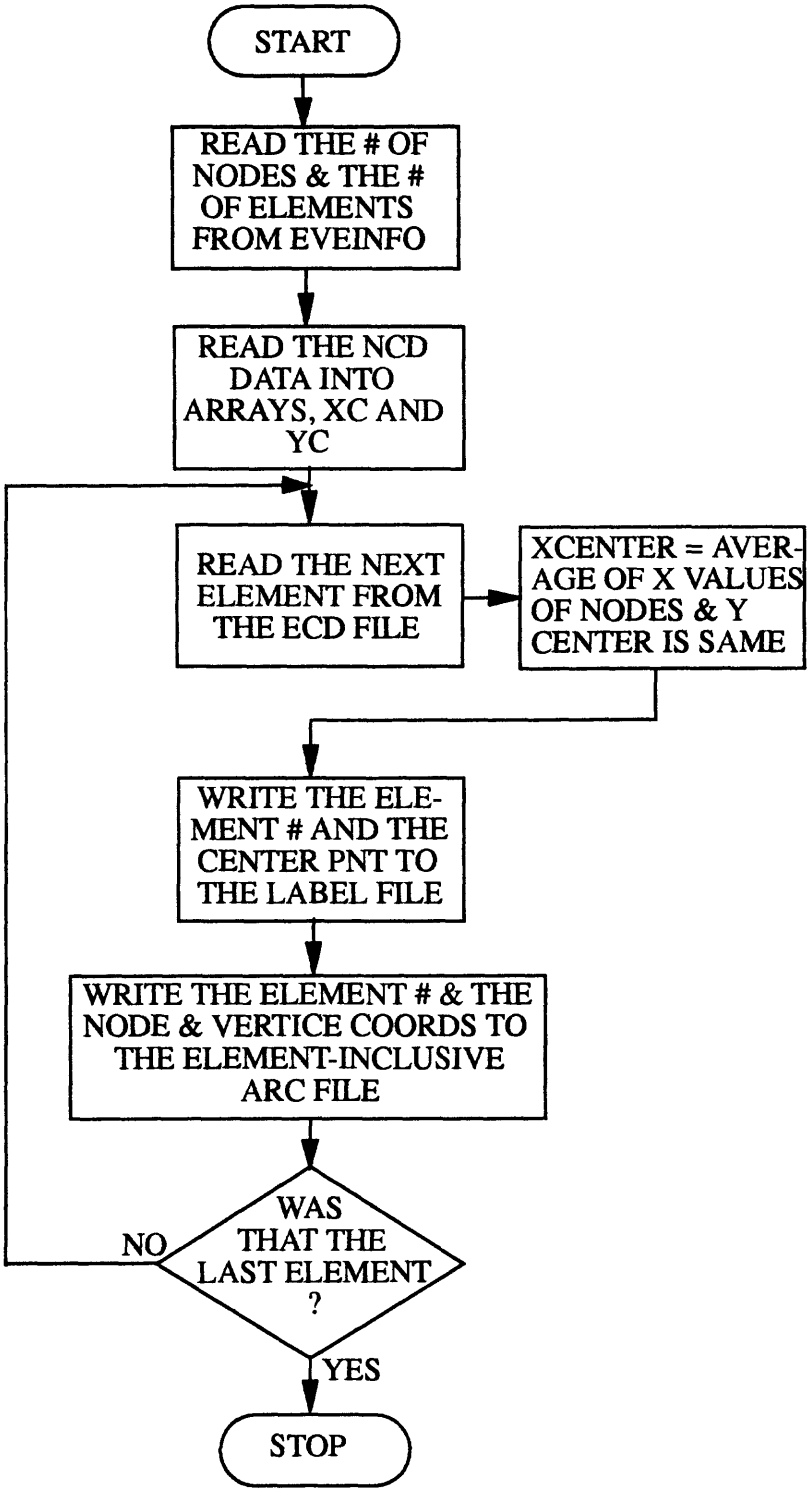


Figure 43.--Flowchart for FE-LABEL.F77.

Program listing

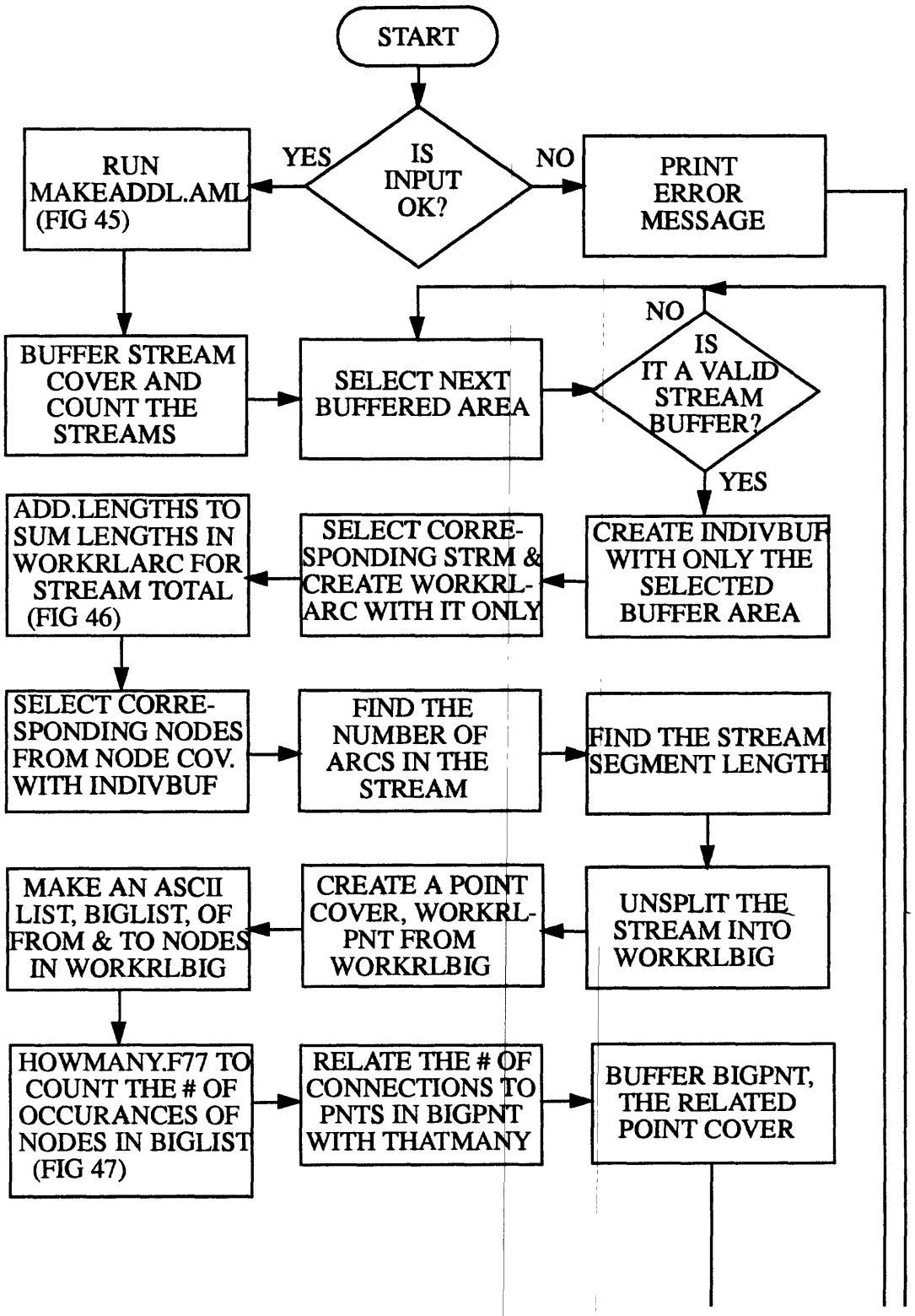
```
************************************************************************
*PROGRAM FOR WRITING ASCII ARCS FILE IN FORMAT FOR ARC/INFO GENERATE
*COMMAND AND A LABEL FILE FOR THE POLYGON COVERAGE OF YOUR FINIT*
*     ELEMENT MESH      E. L. KUNIANSKY  12-18-87                    *
*     MODIFIED BY:  ROBERT LOWTHER  8-15-90                          *
************************************************************************
      COMMON/C1/XC(100000)
      COMMON/C2/YC(100000)
      REAL*8 XC,YC
      CHARACTER*30 FILEN, FILEE, FILEA, FILEL

      OPEN (9,file ='eveinfo',status ='OLD',recl =60)
101    FORMAT (F15.3)
102    FORMAT (A)
      READ(9,*) NNODE
      READ(9,*) NELEM
      FILEN = 'filencd'
      FILEE = 'fileecd' 10   FORMAT(A30)
      READ(9,102) FILEA
      READ(9,102) FILEL

       OPEN(10,FILE=FILEN)
       OPEN(11,FILE=FILEE)
       OPEN(7,FILE=FILEA)
       OPEN(8,FILE=FILEL)
      DO 100 I=1,NNODE
       READ(10,*) jj,XC(jj),YC(jj)
 100  CONTINUE
      DO 200 J=1,NELEM
       READ(11,*) JJ,N1,N2,N3
       XCENTER=(XC(N1)+XC(N2)+XC(N3))/3.
       YCENTER=(YC(N1)+YC(N2)+YC(N3))/3.
       WRITE(8,40) JJ, XCENTER, YCENTER
      J1 = 3*(J-1)+1
      J2 = 3*(J-1)+2
      J3 = 3*(J-1)+3
      WRITE(7,50) J1
      WRITE(7,20) XC(N1),YC(N1)
      WRITE(7,20) XC(N2),YC(N2)
      WRITE(7,60) 'END'
      WRITE(7,50) J2
      WRITE(7,20) XC(N2),YC(N2)
      WRITE(7,20) XC(N3),YC(N3)
      WRITE(7,60) 'END'
      WRITE(7,50) J3
      WRITE(7,20) XC(N3),YC(N3)
      WRITE(7,20) XC(N1),YC(N1)
      WRITE(7,60) 'END'
 200  CONTINUE
  20 FORMAT(5X,2F15.3)
  40 FORMAT(I5,2F15.3)
  50 FORMAT(I5)
```

```
60 FORMAT(A3)
   WRITE(8,60) 'END'
   WRITE(7,60) 'END'
  END
```

REALENGTH.AML

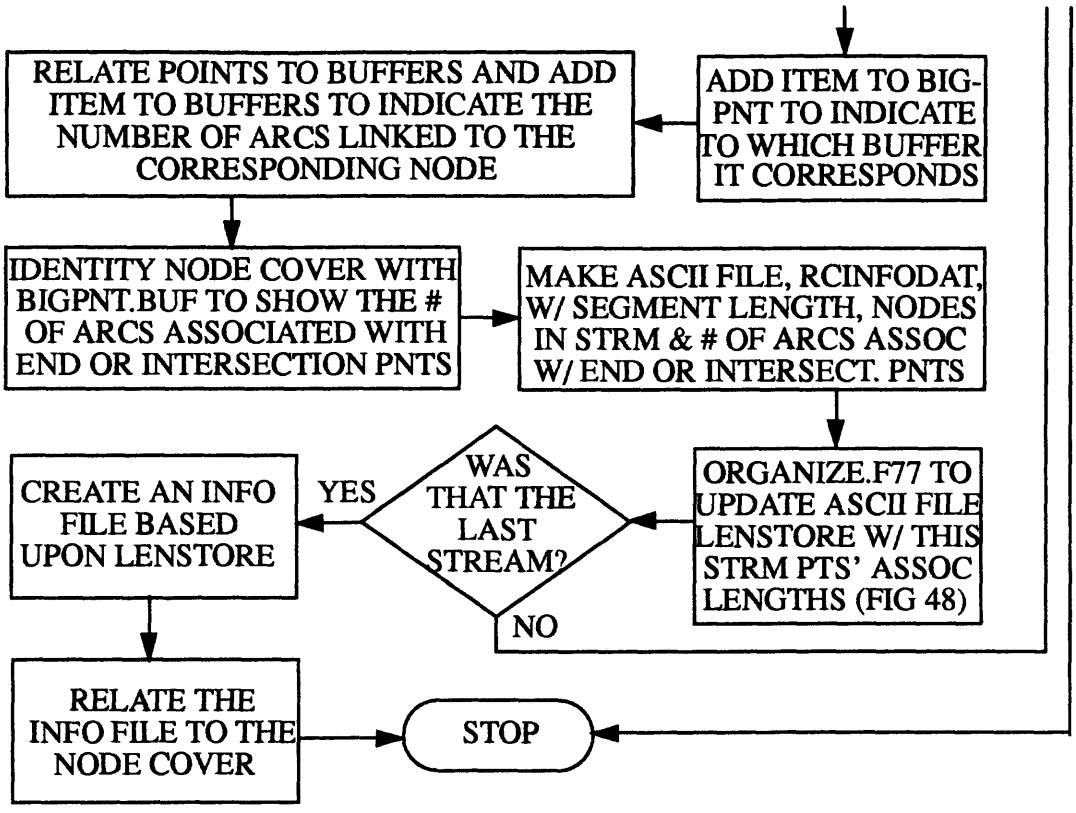
Description

Leakage from a streambed to an aquifer is proportional to the surface area of the streambed. If, for simplicity, streams are assumed to be of uniform width, then leakage is proportional to, among other factors, the length of the stream. The mesh generation process transforms all pertinent features in the study area into representative points. Streams are translated into the points that form the sides of triangular elements. Therefore, in order to examine leakage in the model, a part of the stream length must be associated with each point representing the stream. Because the stream is *SPLINE*d at equal distances along its length, equal parts of the length may be assigned to each representative point. This is the purpose of REALENGTH.AML. The lengths are written into the point cover's PAT as the item "STRMLEN." The REALENGTH flowchart is shown in figure 44.



a. logic which selects a stream and prepares it for length assignment

Figure 44.--Flowchart for REALENGTH.AML.



b. logic which assigns lengths to streams

Figure 44.--Flowchart for REALENGTH.AML--continued.

Program Listing

```
/*   MACRO:  Associate the nodes in a node cover with the length of the
/*           surrounding section of stream
/*   CODED BY:  Robert Lowther
/*   SUPERVISED BY:  Eve L. Kuniansky

/*   VARIABLE LIST:
/*     DTYPE:  The graphic display terminal type
/*     RLARC:  The name of the input stream cover
/*      SNAD:  The snap distance used on the stream cover
/*     MSNOD:  The name of the point cover created using all of the features
/*      BUFD:  The buffering distance used around the stream points
/*     RLBUF:  The buffered version of the stream cover
/* RLBUFPAT:  The PAT for RLBUF
/*  RLBUFAAT:  The AAT for RLBUF
/*     INFOFIL:  The file, RCINFODAT, complete with path name
/*  MSNODINT:  The internal ID for MSNOD
/*     MSCAPS:  MSNOD, capitalized
/*      MSPAT:  The PAT for MSNOD
/*   NUMBUFS:  The number of streams in the cover
```

```
/*     CURRBUF:  The stream buffer currently being considered
/*    CURRMKR:  A marker indicating the current buffer
/*          INSID:  A variable indicating whether or not current buffer is valid
/*    INDIVBUF:  A copy of the buffered stream cover w/only the current buffer
/*  WORKRLARC:  A copy of the stream file with only the current stream
/*       LENGTH:  The overall stream length
/* WORKMSNOD:  A copy of the point cover w/only points from the current stream
/*       SEGLEN:  The length of a section of a stream
/*  WORKRLBIG:  A unsplit copy of the stream cover
/*        BIGPNT:  The points taken from WORKRLBIG
/*       BIGLIST:  An ASCII file containing the # of arcs and a list of all nodes
/*  THATMANY:  An ASCII file containing each node & the # of times it is a
/*                  from or to node in the stream AAT
/*      MANY:  An INFO file version of THATMANY
/*     PNT-ID:  The ID of the point in THATMANY
/*   NOLINKS:  The number of arcs which a node connects
/*BIGPNT.BUF:  A buffered copy of BIGPNT
/*  WHICHBUF:  An indicator of which buffer goes around each point
/*      IDDPNT:  A copy of BIGPNT with a specification of which buffer
/*                  is around each point
/*  IDDMSNOD:  A copy of WORKMSNOD with the # of links to each node
/*TEMPORARY:  A temporary copy of lenstore used as an input to ORGANIZE
/*  LENSTORE:  The ASCII file used to store the river nodes and their
/*                 associated links until each river has been considered
/*       NODID:  The user ID for msnod
/*       RULER:  LENSTORE, complete with path name
/*      CLOVIS:  The INFO file version of LENSTORE
/*    WICHNOD:  The node number in CLOVIS
/*    STRMLEN:  The stream segment length in CLOVIS

&echo &off
&args dtype rlarc snad msnod

/*  -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/*  -Test to see if all arguments are present as expected-O
&if [type %rlarc%] ne 1 &then &goto badentry
&if [type %snad%] gt 0 &then &goto badentry
&if [type %msnod%] ne 1 &then &goto badentry
&if [length %msnod%] eq 0 &then &goto badentry

/*  -Prepare the error-indication file-M
```

```
&s i [delete coderr]
&setvar filunit [open coderr openstatus -w]
&setvar i [write %filunit% 9]
&setvar i [close %filunit%]

/* -Define variables to be used-E
&setvar bufd %snad% * 0.6
&setvar rlbuf [translate %rlarc%].BUF
&setvar rlbufpat %rlbuf%.PAT
&setvar rlbufaat %rlbuf%.AAT
&setvar infofil [pathname RCINFODAT]
&setvar noninfofil rcinfodat
&setvar msnodint [translate %msnod%]#
&setvar msnodid [translate %msnod%]-ID
&setvar mscaps [translate %msnod%]
&setvar mspat [translate %msnod%].PAT
&s i [delete nolen]
&s i [delete lenstore]

/* -Create the INFO program to be run-D
&r makeaddl

/* -Buffer the original input cover and determine the number of study areas-Y
&if [exists %rlbuf% -coverage] &then &goto nobuff
 &severity &error &ignore
 kill arccopy all
 &severity &error &fail
 copy %rlarc% arccopy
 &r spleen %dtype% %bufd% 0 0 arccopy
 buffer arccopy %rlbuf% # # %bufd% 40 line
 kill arccopy all
&label nobuff
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %rlbufpat%
OUTPUT %infofil% INIT
PRINT $NOREC
OUTPUT XXXNSP
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %rlbufpat%
OUTPUT %infofil% INIT
PRINT $NOREC
OUTPUT XXXNSP
Q STOP
QUIT
&end
```

```
&end
&setvar filunit [open %noninfofil% openstatus -r]
&setvar numbufs [trim [read %filunit% rdstat]]
&setvar i [close %filunit%]
&s i [delete %noninfofil%]

/* -Prepare to progress through each buffer in the buffered cover-
&setvar currbuf 1
build %rlbuf% line
&severity &error &ignore
additem %rlbufpat% %rlbufpat% currmkr 4 4 i
&severity &error &fail
&s i [delete nolen]

&label begloop
remepf -prg -na -nq -nvfy

/* -Find the next buffered area-H
&setvar currbuf %currbuf% + 1
&if %currbuf% GT %numbufs% &then &goto endloop
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %rlbufpat%
CALC CURRMKR = 0
RESEL FOR $RECNO = %currbuf%
CALC CURRMKR = 1
OUTPUT %infofil% INIT
PRINT INSIDE
OUTPUT XXXNSP
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %rlbufpat%
CALC CURRMKR = 0
RESEL FOR $RECNO = %currbuf%
CALC CURRMKR = 1
OUTPUT %infofil% INIT
PRINT INSIDE
OUTPUT XXXNSP
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy

/* -Determine the validity of the selected area-O
&setvar filunit [open %noninfofil% openstatus -r]
&setvar insid [trim [read %filunit% rdstat]]
```

```
&setvar i [close %filunit%]
&s i [delete %noninfofil%]
&if %insid% ne 100 &then &goto begloop

/* -Isolate the selected area-U
&severity &error &ignore
kill indivbuf all
&severity &error &fail
dissolve %rlbuf% indivbuf currmkr poly

/* -Isolate the appropriate stream-R
&severity &error &ignore
kill workrlarc all
&severity &error &fail
intersect %rlarc% indivbuf workrlarc line 40
remepf -prg -na -nq -nvfy

/* -Find the stream segment lengths-
&s i [delete %noninfofil%]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
CALC $COMMA-SWITCH = -1
SELECT WORKRLARC.AAT
CALCULATE $NUM1 = 1
CALCULATE $NUM2 = 0
RUN ADD.LENGTHS
OUTPUT %infofil% INIT
PRINT $NUM2
OUTPUT XXXNSP
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
CALC $COMMA-SWITCH = -1
SELECT WORKRLARC.AAT
CALCULATE $NUM1 = 1
CALCULATE $NUM2 = 0
RUN ADD.LENGTHS
OUTPUT %infofil% INIT
PRINT $NUM2
OUTPUT XXXNSP
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy
&setvar filunit [open %noninfofil% openstatus -r]
&setvar length [trim [read %filunit% rdstat]]
&setvar i [close %filunit%]
```

```
&s i [delete %noninfofil%]

/* -Isolate the appropriate mesh nodes-S
&severity &error &ignore
kill workmsnod all
&severity &error &fail
intersect %msnod% indivbuf workmsnod point 40

/* -Find the number of mesh arcs in the area-O
&s i [delete %noninfofil%]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT WORKMSNOD.PAT
OUTPUT %infofil% INIT
PRINT $NOREC
OUTPUT XXXNSP
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT WORKMSNOD.PAT
OUTPUT %infofil% INIT
PRINT $NOREC
OUTPUT XXXNSP
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy
&setvar filunit [open %noninfofil% openstatus -r]
&setvar numarcs [trim [read %filunit% rdstat]] - 1
&setvar i [close %filunit%]
&s i [delete %noninfofil%]

/* -Find the appropriate segment length-F
&setvar seglen %length% / 2
&setvar seglen %seglen% / %numarcs%

/* -Make an unsplit copy of the stream-T
&severity &error &ignore
kill workrlbig all
kill bigpnt all
&severity &error &fail
copy workrlarc workrlbig
ae
disp %dtype%
mape workrlbig
editc workrlbig
editf arc
```

```
drawe arc node
draw
select screen
&severity &error &ignore
unsplit none
&severity &error &fail
save
q
remepf -prg -na -nq -nvfy
build workrlbig line

/* -Create a derivative point cover and FORTRAN file-W
nodepoint workrlbig bigpnt
build bigpnt point
&s i [delete biglist]
&setvar biglist [pathname BIGLIST]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT WORKRLBIG.AAT
OUTPUT %biglist% INIT
PRINT $NOREC
PRINT FNODE#
PRINT TNODE#
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT WORKRLBIG.AAT
OUTPUT %biglist% INIT
PRINT $NOREC
PRINT FNODE#
PRINT TNODE#
Q STOP
Q
&end
&end
remepf -prg -na -nq -nvfy

/* Use FORTRAN to create a list of nodes and number of occurances-A
&s i [delete thatmany]
&setvar dumarg DummY
&if .computer_type = 'prime' &then &sys r howmany
&else &sys howmany.out

/* Relate the FORTRAN output file to the point cover-R
&setvar thatmany [pathname THATMANY]
&severity &error &ignore
additem bigpnt.pat bigpnt.pat PNT-ID 5 5 i
additem bigpnt.pat bigpnt.pat NOLINKS 5 5 i
```

```
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT MANY
PURGE
Y
ERASE MANY
Y
DEFINE MANY
PNT-ID,5,5,I
NOLINKS,5,5,I

GET %thatmany% COPY
SELECT BIGPNT.PAT
CALC PNT-ID = BIGPNT-ID
SELECT MANY
RELATE BIGPNT.PAT by PNT-ID
CALC $1NOLINKS = NOLINKS
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT MANY
PURGE
Y
ERASE MANY
Y
DEFINE MANY
PNT-ID,5,5,I
NOLINKS,5,5,I

GET %thatmany% COPY ASCII
SELECT BIGPNT.PAT
CALC PNT-ID = BIGPNT-ID
SELECT MANY
RELATE BIGPNT.PAT by PNT-ID
CALC $1NOLINKS = NOLINKS
Q STOP
QUIT
&end
&end

remepf -prg -na -nq -nvfy
dropitem bigpnt.pat bigpnt.pat PNT-ID

/* -Buffer the node file with only the endpoints & intersections as nodes-R
&severity &error &ignore
kill bigpnt.buf all
&severity &error &fail
```

```
buffer bigpnt bigpnt.buf # # %bufd% 40 point
build bigpnt.buf poly

/* -Identify each buffer in the buffered cover-E
additem bigpnt.buf.pat bigpnt.buf.pat WICHBUF 5 5 i
additem bigpnt.buf.pat bigpnt.buf.pat NOLINKS 5 5 i
tables
SELECT BIGPNT.BUF.PAT
CALC WICHBUF = $RECNO - 1
Q STOP
remepf -prg -na -nq -nvfy

/* -Identify each point by which buffer is around it-
&severity &error &ignore
kill iddpnt all
&severity &error &fail
identity bigpnt bigpnt.buf iddpnt point 40
dropitem iddpnt.pat iddpnt.pat BIGPNT-ID
dropitem iddpnt.pat iddpnt.pat BIGPNT#
dropitem iddpnt.pat iddpnt.pat BIGPNT.BUF-ID
dropitem iddpnt.pat iddpnt.pat BIGPNT.BUF#
dropitem iddpnt.pat iddpnt.pat INSIDE

/* -Associate the number of links with the buffers-
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT IDDPNT.PAT
RELATE BIGPNT.BUF.PAT by WICHBUF
CALC $1NOLINKS = NOLINKS
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT IDDPNT.PAT
RELATE BIGPNT.BUF.PAT by WICHBUF
CALC $1NOLINKS = NOLINKS
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy

/* -Identity the link-numbered buffer cover with the all stream point cover-
&severity &error &ignore
kill iddmsnod all
&severity &error &fail
identity workmsnod bigpnt.buf iddmsnod point 40
dropitem iddmsnod.pat iddmsnod.pat BIGPNT.BUF-ID
dropitem iddmsnod.pat iddmsnod.pat BIGPNT.BUF#
```

```
dropitem iddmsnod.pat iddmsnod.pat WORKMSNOD-ID
dropitem iddmsnod.pat iddmsnod.pat WORKMSNOD#
dropitem iddmsnod.pat iddmsnod.pat %msnodint%
dropitem iddmsnod.pat iddmsnod.pat INDIVBUF-ID
dropitem iddmsnod.pat iddmsnod.pat INDIVBUF#
dropitem iddmsnod.pat iddmsnod.pat INSIDE
dropitem iddmsnod.pat iddmsnod.pat CURRMKR
dropitem iddmsnod.pat iddmsnod.pat WICHBUF

/*  -Make the node-river length-storage file that FORTRAN can read-
&s i [delete %noninfofil%]
&severity &error &ignore
additem iddmsnod.pat iddmsnod.pat SEGLEN 4 4 i
&severity &error &fail
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
CALC $COMMA-SWITCH = -1
SELECT IDDMSNOD.PAT
RESEL FOR $RECNO = 1
CALC SEGLEN = %seglen%
OUTPUT %infofil% INIT
PRINT SEGLEN
SELECT IDDMSNOD.PAT
PRINT $NOREC
PRINT %msnodid%,NOLINKS
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
CALC $COMMA-SWITCH = -1
SELECT IDDMSNOD.PAT
RESEL FOR $RECNO = 1
CALC SEGLEN = %seglen%
OUTPUT %infofil% INIT
PRINT SEGLEN
SELECT IDDMSNOD.PAT
PRINT $NOREC
PRINT %msnodid%,NOLINKS
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy
dropitem iddmsnod.pat iddmsnod.pat SEGLEN

/*  -Assign lengths to the nodes-
&s i [delete temporary]
&severity &error &ignore
&if %.computer_type% = 'prime' &then &sys copy lenstore temporary
```

```
&else &sys cp lenstore temporary
&severity &error &fail
&s i [delete lenstore]
&if .computer_type = 'prime' &then &sys r organize %dumarg%]
&else &sys organize.out
&s i [delete temporary]
remepf -prg -na -nq -nvfy
&goto begloop

&label endloop

/*  -Alter the mesh node file to add the stream segment lengths-
remepf -prg -na -nq -nvfy
&setvar nodid [translate %msnod%]-ID
&setvar ruler [pathname LENSTORE]
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT CLOVIS
PURGE
Y
ERASE CLOVIS
Y
DEFINE CLOVIS
WICHNOD,7,7,I
STRMLEN,15,15,N,3

GET %ruler% COPY
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT CLOVIS
PURGE
Y
ERASE CLOVIS
Y
DEFINE CLOVIS
WICHNOD,7,7,I
STRMLEN,15,15,N,3

GET %ruler% COPY ASCII
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy
&severity &error &ignore
dropitem %mspat% %mspat% STRMLEN
additem %mspat% %mspat% STRMLEN 15,15,N,3
```

```
additem %mspat% %mspat% WICHNOD 7,7,I
&severity &error &fail
remepf -prg -na -nq -nvfy
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
SELECT %mspat%
CALC WICHNOD = %nodid%
SELECT CLOVIS
RELATE %mspat% BY WICHNOD
CALC $1STRMLEN = STRMLEN
Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
SELECT %mspat%
CALC WICHNOD = %nodid%
SELECT CLOVIS
RELATE %mspat% BY WICHNOD
CALC $1STRMLEN = STRMLEN
Q STOP
QUIT
&end
&end
remepf -prg -na -nq -nvfy

/* -Clean up the files-
&severity &error &ignore
dropitem %mspat% %mspat% WICHNOD
&s i [delete lenstore]
&s i [delete temporary]
&s i [delete nolen]
&s i [delete biglist]
&s i [delete rcinfodat]
&s i [delete thatmany]
kill iddmsnod all
kill bigpnt all
kill bigpnt.buf all
kill iddpnt all
kill indivbuf all
kill %rlbuf% all
kill workmsnod all
kill workrlarc all
kill workrlbig all
&severity &error &fail

/* -Prepare the error-indication file-
&s i [delete coderr]
&setvar filunit [open coderr openstatus -w]
&setvar i [write %filunit% noerr]
```

```
&setvar i [close %filunit%]

&goto quitit
&label badentry

/* -Print the error message-
&type Usage:  REALENGTH <display type><original length stream cover(existing)>
&type                  <minimum distance between derivative stream cover
&type                   points> <derivative stream point cover (existing)>

&label quitit
&type End of REALENGTH
```

MAKEADDL.AML

Description

This macro creates the INFO program "ADD.LENGTHS." Because it is cumbersome to copy INFO programs whenever the mesh generation programs are copied, MAKEADDL.AML rebuilds ADD.LENGTHS whenever REALENGTH.AML is run. Its flowchart is in figure 45.

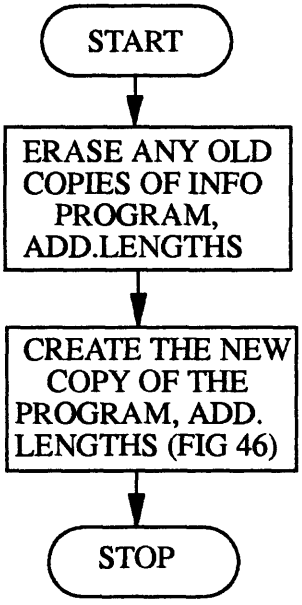


Figure 45.--Flowchart for MAKEADDL.AML.

Program listing

```
/* MACRO: Create the INFO program, ADD.LENGTHS to total all arc lengths
/*              in a cover
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky

/* VARIABLE LIST
/*    $NUM1: The counter indicating the current arc
/*   $NOREC: The total number of arcs
/*    $NUM2: The total of all arc lengths
/*   LENGTH: The length of a particular arc

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Delete any old copy of the program and create a new copy-H
&if %.computer_type% = 'prime' &then
&do
&data ARC INFO
 &severity &error &ignore
 ERASE ADD.LENGTHS
 Y
 &severity &error &fail
 PROGRAM ADD.LENGTHS
 DO WHILE $NUM1 LE $NOREC
 SELECT WORKRLARC.AAT
 RESEL FOR $RECNO = $NUM1
 CALCULATE $NUM2 = $NUM2 + LENGTH
 CALCULATE $NUM1 = $NUM1 + 1
 DOEND

 Q STOP
&end
&end
&else
&do
&data ARC
INFO
ARC
&severity &error &ignore
ERASE ADD.LENGTHS
Y
```

```
&severity &error &fail
PROGRAM ADD.LENGTHS
DO WHILE $NUM1 LE $NOREC
SELECT WORKRLARC.AAT
RESEL FOR $RECNO = $NUM1
CALCULATE $NUM2 = $NUM2 + LENGTH
CALCULATE $NUM1 = $NUM1 + 1
DOEND

Q STOP
QUIT
&end
&end
```

Info Program ADD.LENGTHS

Description

This INFO program totals the lengths of all arcs that compose a given stream (fig. 46). This total, the total length of the stream, is used as an input to REALENGTH.

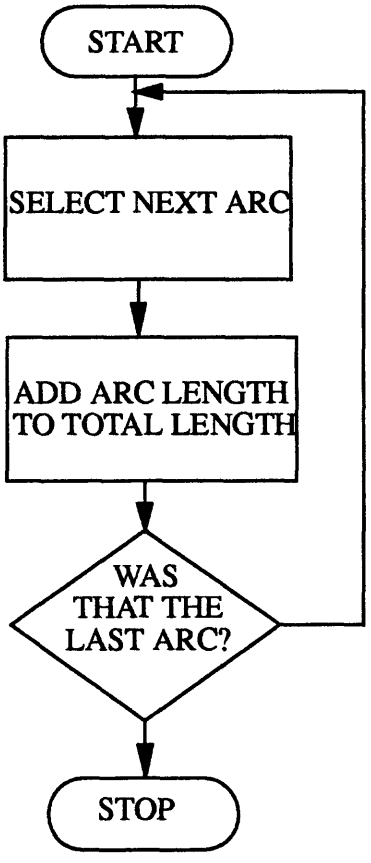


Figure 46.--Flowchart for ADD.LENGTHS.

Program listing

```
PROGRAM ADD.LENGTHS
DO WHILE $NUM1 LE $NOREC
SELECT WORKRLARC.AAT
RESEL FOR $RECNO = $NUM1
CALCULATE $NUM2 = $NUM2 + LENGTH
CALCULATE $NUM1 = $NUM1 + 1
DOEND
```

Fortran Program HOWMANY.F77

Description

The length assigned to a particular node of a stream is equivalent to the sum of stream lengths between a node and each of the midpoints between that node and the two nearest nodes. Each of these lengths is a segment length. While nodes in the middle of a stream will all be assigned the same length (two times the aforementioned segment length), nodes at the ends of streams and nodes at stream intersections will be assigned different lengths. The length assigned is an integer multiple of the basic segment length, dependent on the number of arc connections to the node in question. Ends of streams will only represent one segment length, and nodes at the intersection of three stream branches will represent three segment lengths.

HOWMANY.F77 determines the number of arcs connected to each node (fig. 47). It takes the arc cover input and creates a list, organized by node, of the arc connections. This list is written to an ASCII file which REALENGTH.AML can read.

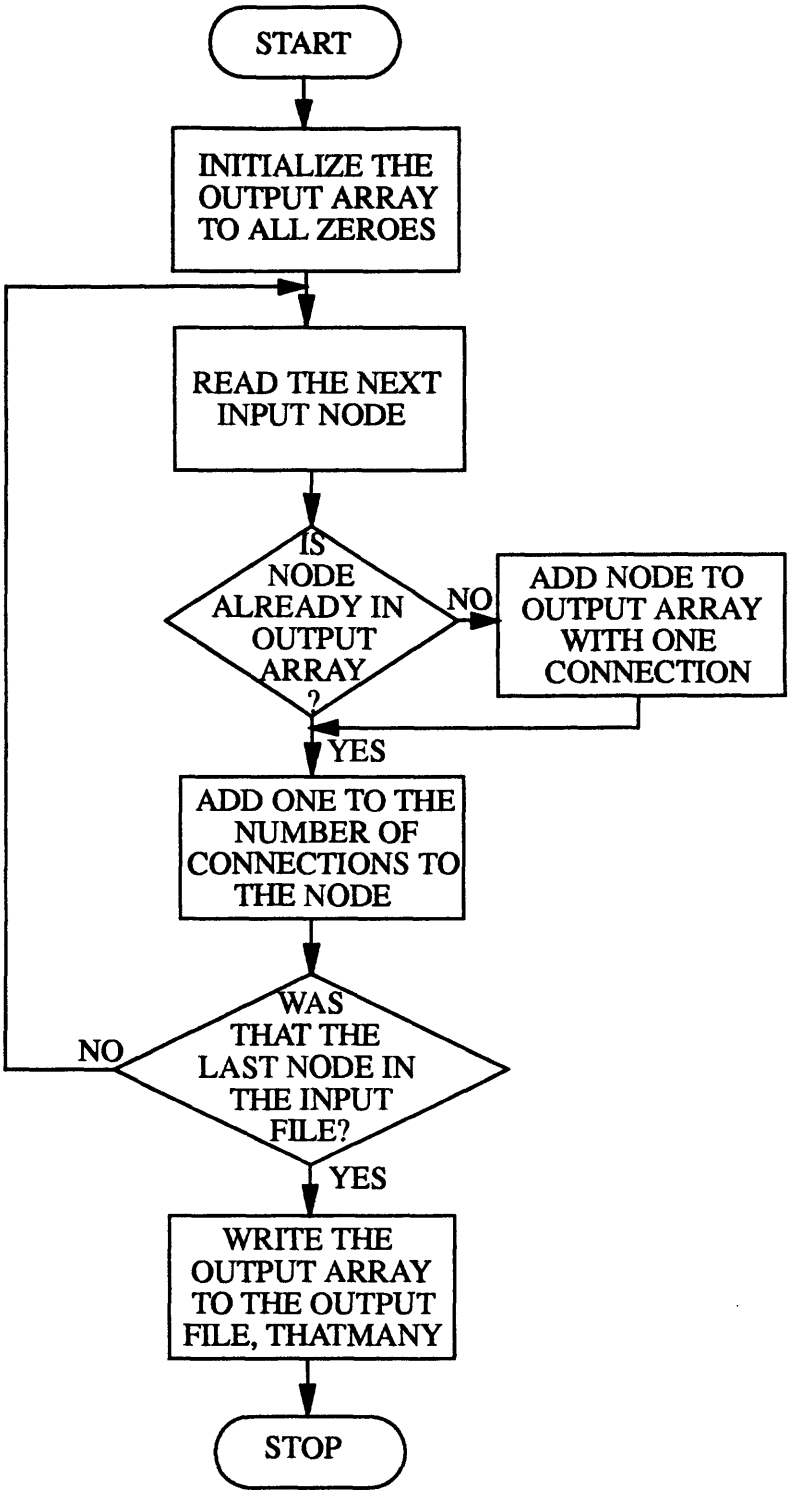


Figure 47.--Flowchart for HOWMANY.F77.

Program listing

```
C*************************************************************************C
C   PROGRAM: Determine the number of arc connections for each node in the    C
C                    input stream                                            C
C   CODED BY: Robert Lowther                                                 C
C   SUPERVISED BY: Eve L. Kuniansky                                          C
C*************************************************************************C

    PROGRAM HOWMANY

C*************************************************************************C
C   VARIABLE LIST:                                                         C
C                                                                          C
C      I,J: Counter variables                                              C
C     OLIST: Array containing the output list of nodes and # ofconnections  H
C      NUMO: The total number of nodes in the output file                  S
C       NOD: A given node number                                           C
C      OFND: Indicates that the given node has been found in theoutputfile C
C*************************************************************************C

    COMMON/C1/OLIST(999,2)

    INTEGER I,J,OLIST,NUMO,NOD,OFND


100  FORMAT (2I5)

C-----Open the input and output files-------------C
    OPEN (7,FILE= 'biglist')
    OPEN (8,FILE= 'thatmany')

    READ (7,*) NUMI

C-----Initialize the output array-----------------C
    DO 10, I=1,999
     OLIST(I,1) = 0
     OLIST(I,2) = 0
10   CONTINUE

C-----Read a node from the input file---------------C
    NUMO = 0
    DO 20, I=1,NUMI*2
     OFND = 0
     READ (7,*) NOD

C-----Check the output file:  if found, add one to the node's total connectionsC
     DO 30, J=1,NUMO
      IF (OLIST(J,1) .NE. NOD) GO TO 30
       OFND = 1
       OLIST(J,2) = OLIST(J,2) + 1
30    CONTINUE

C-----If node not in output array, write it & increase the array's node totalC
```

```
      IF (OFND .EQ. 1) GO TO 20
       NUMO = NUMO + 1
       OLIST(NUMO,1) = NOD
       OLIST(NUMO,2) = 1
20    CONTINUE

C-----Write the output array to the output ASCII file-----C
      DO 40, I=1,NUMO
       WRITE (8,100) OLIST(I,1),OLIST(I,2)
40    CONTINUE

C-----Close all files----------C
      CLOSE (7)
      CLOSE (8)
      END
```

Fortran Program ORGANIZE.F77

Description

This program updates the ASCII file containing the node and associated length data (fig. 48). It reads the data generated from streams in previous iterations from the file "TEMPORARY." It writes the input data and the current stream data to the storage file, "LENSTORE." TEMPORARY is merely a copy of LENSTORE created as a convenience just before ORGANIZE.F77 is called.

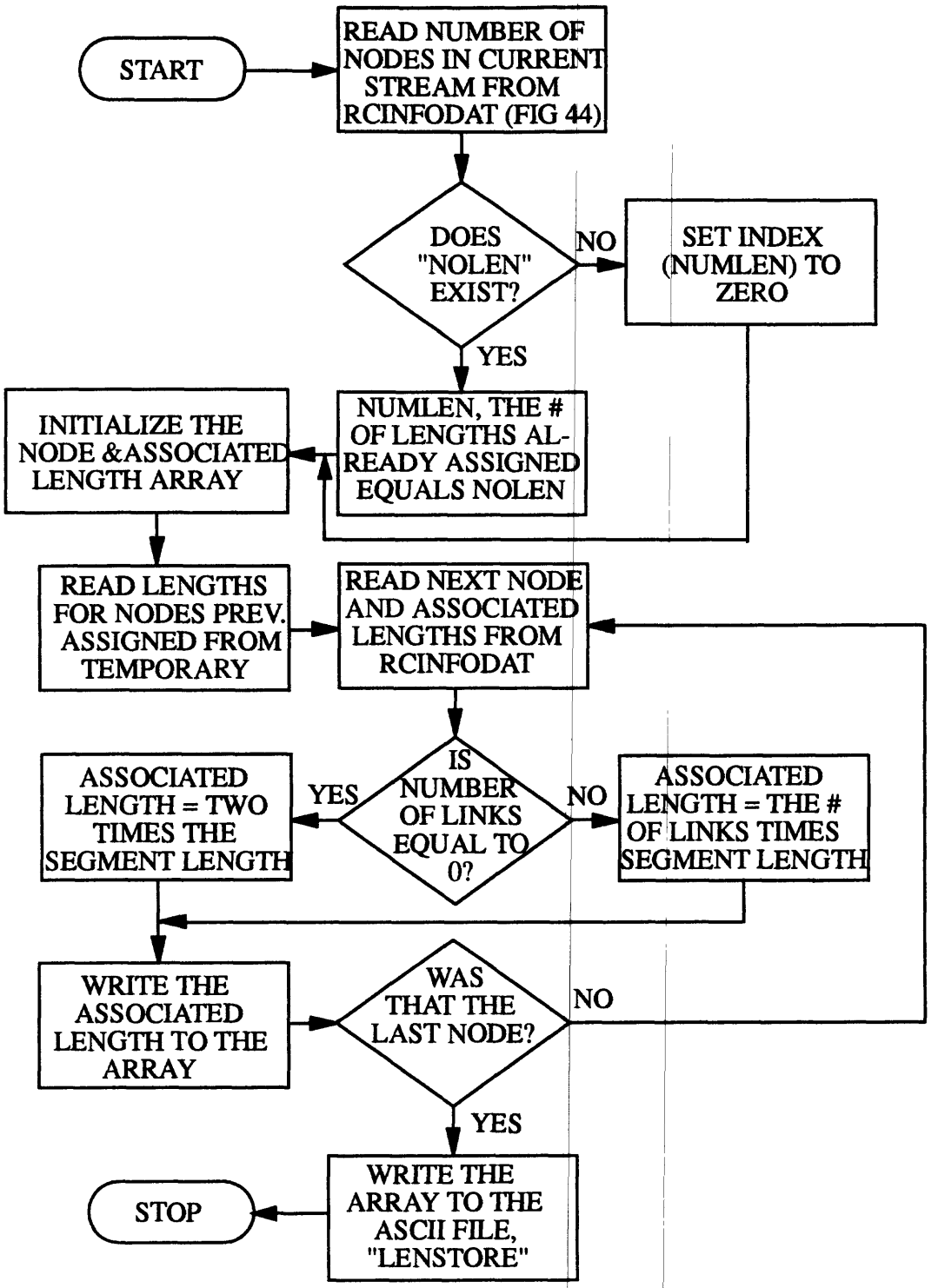


Figure 48.--Flowchart for ORGANIZE.F77.

Program listing

```
C*************************************************************************C
C   PROGRAM: Update the node-associated length file with currentstreaminfo    C
C   CODED BY: Robert Lowther                                                  C
C   SUPERVISED BY: Eve L. Kuniansky                                           C
C*************************************************************************C

    PROGRAM ORGANIZE

C*************************************************************************C
C   VARIABLE LIST:                                                          H
C                                                                           S
C    NUMARC: The number of arcs in the current stream                       C
C    NUMNOD: The number of nodes in the current stream                      C
C         I: A counter variable                                             C
C       NOD: The current node number, and pointer for LENNOD                C
C   NOLINKS: The number of connections to a given node                      C
C    SEGLEN: The length of a particular section of stream                   C
C    LENNOD: The array containing the lengths associated w/ allstreamnodes  C
C    NODLEN: The input length, from TEMPORARY, associated with a node       C
C    DUMARG: A dummy argument                                               C
C     DOESIT: Indicates whether or not the file w/ the number of nodes w/   C
C             associated lengths already exists                             H
C*************************************************************************S

    COMMON/C1/LENNOD(25000)

    INTEGER NUMARC,NUMNOD,I,NOD,NOLINKS,NUMLEN,NUM
    DOUBLE PRECISION SEGLEN,LENNOD,NODLEN
    CHARACTER DUMARG
    LOGICAL*4 DOESIT

C=====Open the information, input and output files========C
    OPEN (8,FILE= 'rcinfodat', RECL= 999)
    OPEN (9,FILE= 'lenstore', RECL= 999)
    OPEN (7,FILE= 'temporary', RECL = 999)

101  FORMAT (F15.3)
102  FORMAT (I7)
103  FORMAT (I7,F15.3)

C=====Read the current stream section length & number of nodes=======C
    READ (8,*) SEGLEN
    PRINT *,'CURRENT STREAM SECTION LENGTH:',SEGLEN
    READ (8,*) NUMNOD
    PRINT *,'NUMBER OF NODES IN THE CURRENT STREAM:',NUMNOD

C=====Determine the number of nodes in the input file================C
    INQUIRE (FILE = 'nolen',EXIST= DOESIT)
    IF (DOESIT .EQV. .FALSE.) GO TO 12
     OPEN (10,FILE= 'nolen')
     READ(10,102) NUMLEN
```

```
    GO TO 13
12  NUMLEN = 0

C=====Initialize the node/associated length array============C
13  DO 10, I=1,25000
    LENNOD(I) = 0
10  CONTINUE

    PRINT *,'NUMBER OF NODES ALREADY ASSIGNED IN THIS COVER:',NUMLEN

C=====Read all previously assigned nodes into the array=========C
    DO 11, I=1,NUMLEN
    READ (7,*) NOD,NODLEN
    LENNOD(NOD) = NODLEN
11  CONTINUE

C=====Read the current stream and write lengths to the array=====C
21  DO 20, I=1,NUMNOD
    READ (8,*) NOD,NOLINKS
    NUMLEN = NUMLEN + 1
    IF (NOLINKS .NE. 0) LENNOD(NOD) = NOLINKS*SEGLEN
    IF (NOLINKS .EQ. 0) LENNOD(NOD) = 2*SEGLEN
20  CONTINUE

C=====Write to the output files=========================C
    PRINT *,'TOTAL NUMBER OF NODES ASSIGNED:',NUMLEN
    IF (DOESIT .EQV. .FALSE.) OPEN (10,FILE= 'nolen')
    REWIND (10)
    WRITE (10,102) NUMLEN
    DO 30, I=1,25000
    IF (LENNOD(I) .NE. 0) WRITE (9,103) I,LENNOD(I)
30  CONTINUE

C=====Close all files============C
    CLOSE (8)
    CLOSE (9)
    CLOSE (7)
    CLOSE (10)
    END
```

REMODEL.AML

Description

This macro is nearly identical to MODEL.AML (fig. 49). It writes necessary data to an ASCII file that can be read by its Fortran program, REOPT.F77. It re-optimizes the numbering system of a mesh once FILENCD and FILEECD already exist. It creates the same output coverages as MODEL.AML. It does not force the input file to be optimized, as MODEL.AML does, but makes this optional. REMODEL.AML is therefore useful if modifications have been made to the FILENCD and FILEECD files and new output coverages are needed. The macro is also useful if optimization must not be performed so that the new files can in some way be linked to the old files. This occurs when a one-layer model is expanded to multiple layers and the relations between the nodes of each element need to be maintained.

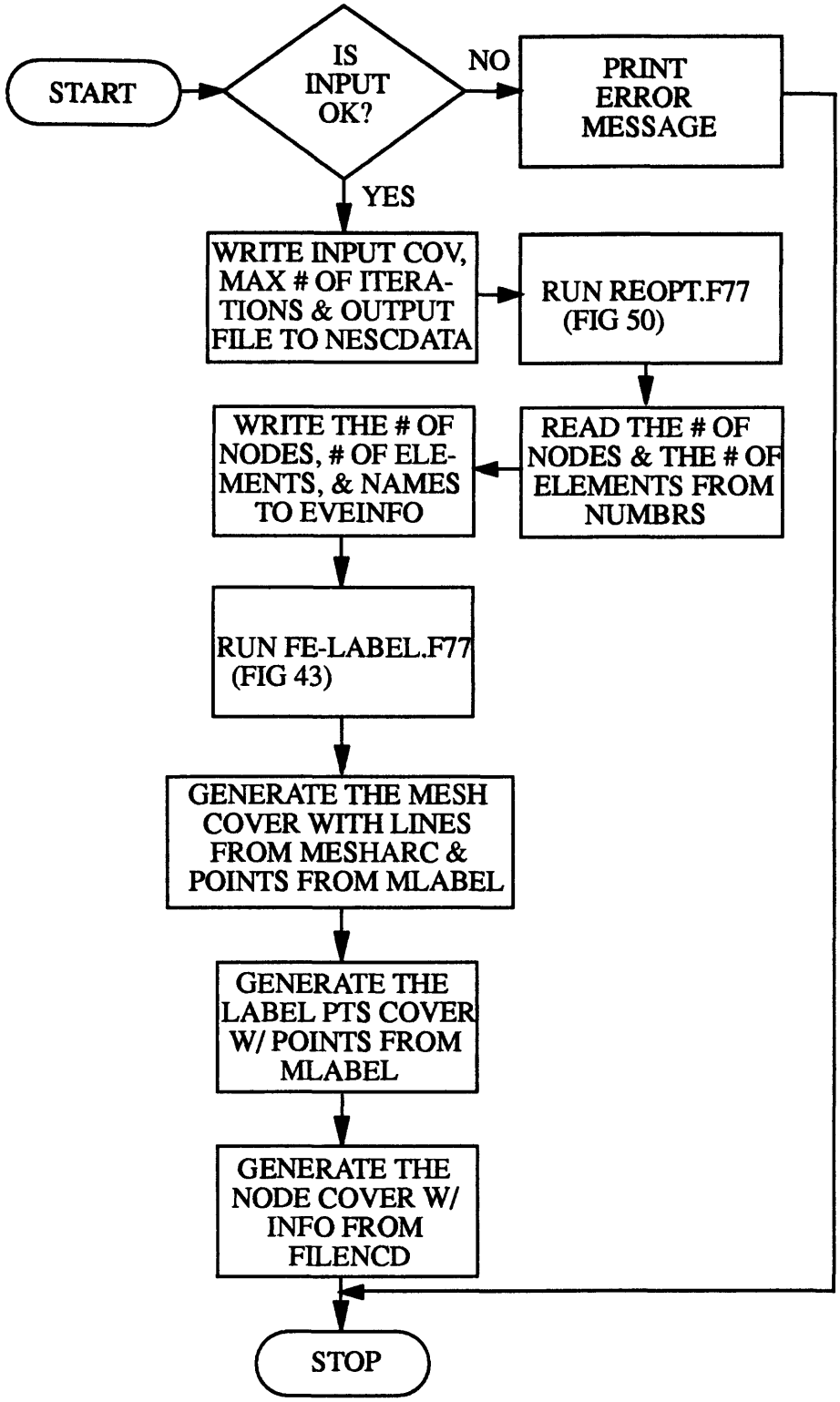


Figure 49.--Flowchart for REMODEL.AML.

```
/* MACRO: Optimize the Node Coordinate Data file and the Element Connection
/*          Data file needed for modelling.  Also, build a printable file
/*          detailing the optimization of the nodes performed, and build ARC
/*          files based on the re-labeled arc file, re-labeled label file,
/*          and the re-labeled node coordinate data file.
/* CODED BY: Robert Lowther
/* SUPERVISED BY: Eve L. Kuniansky modified 11-09-92

/* VARIABLE LIST
/*  MAXITR: The maximum number of optimizing iterations
/*   FPRINT: The name of the printable file to be created
/*   NNODE: The number of nodes in the input cover
/*   NELEM: The number of elements in the input cover
/*     MESH: The name of the mesh polygon cover with labels equal
/*           to the element number
/* MESHLAB: The name of the point cover with point ID at the
/*           center of each element labeled with the element number
/* NODECRD: The name of the point cover whose ID equals the
/*           node number
/*   FILUNIT: The unit number of the data file, EVEINFO
/*   EVEINFO: The data file used to pass data to FE-LABEL.F77
/*  MESHARC: The output element arc file from FE-LABEL.F77
/*   MLABEL: The output element node file from FE-LABEL.F77
/*  FE-LABEL: The Fortran77 program which creates the basis for the outputs

&echo &off
&args maxitr nnode nelem fprint opt mesh meshlab nodecrd

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
 &end
&else
 &do
  &s .computer_type unix
 &end

/* -Test input to see if all arguments are present as expected-O
&if [type %maxitr%] ne -1 &then &goto badentry
&if [type %fprint%] ne 1 &then &goto badentry
&if [type %opt%] ne 1 &then &goto badentry
&if [type %mesh%] ne 1 &then &goto badentry
&if [type %meshlab%] ne 1 &then &goto badentry
&if [type %nodecrd%] ne 1 &then &goto badentry
&if [length %nodecrd%] eq 0 &then &goto badentry

/* -Delete any old occurances of the reopt output files-M
```

```
&label beginit
&s i [delete vertices]


/* -Build the file which REOPT needs-E
&setvar cover = 'dummy'
&setvar filunit [open nescdata openstatus -write]
&setvar i [write %filunit% %cover%]
&setvar i [write %filunit% %maxitr%]
&setvar i [write %filunit% %fprint%]
&setvar i [write %filunit% %nnode%]
&setvar i [write %filunit% %nelem%]

&if [close %filunit%] = 0 &then &type File written successfully.

&if %opt% = 'Y' &then
 &do
  &if %.computer_type% = 'prime' &then &sys r reopt
  &else &sys reopt.out
 &end
&if %opt% = 'y' &then
 &do
  &if %.computer_type% = 'prime' &then &sys r reopt
  &else &sys reopt.out
 &end
&if %opt% = 'YES' &then
 &do
  &if %.computer_type% = 'prime' &then &sys r reopt
  &else &sys reopt.out
 &end
&if %opt% = 'yes' &then
 &do
  &if %.computer_type% = 'prime' &then &sys r reopt
  &else &sys reopt.out
 &end

&label go_on
/*  -Read the file created by Optimize which is necessary to create the
/*   output coverages-D
&setvar filunit [open numbrs openstatus -r]
&if %filunit% ne 0 &then &setvar nnode [read %filunit% readstatus]
&if %filunit% eq 0 &then &s nnode [response 'How many nodes are there?']
&if %filunit% ne 0 &then &setvar nelem [read %filunit% readstatus]
&if %filunit% eq 0 &then &s nelem [response 'How many elements are there?']
&setvar i [close %filunit%]

/*  -Create the data file which FE-LABEL can read as input-Y
&setvar filunit [open eveinfo openstatus -w]
&setvar i [write %filunit% %nnode%]
&setvar i [write %filunit% %nelem%]
&setvar i [write %filunit% mesharc]
&setvar i [write %filunit% mlabel]
&if [close %filunit%] eq 0 &then &type File created successfully.
```

```
/* -Create the ASCII output files-
&if %.computer_type% = 'prime' &then &sys r fe-label
&else &sys fe-label.out

/* -Delete the FE-LABEL input data file-H
&s i [delete eveinfo]

/* -Delete any old occurances of the output coverages-O
&severity &error &ignore
kill %mesh% all
kill %meshlab% all
kill %nodecrd% all
&severity &error &fail

/* -Create the mesh polygon output cover-U
&if %.computer_type% = 'prime' &then
&do
generate %mesh%
input mesharc
line
input mlabel
point
q
&end
&else
&do
&data arc generate %mesh%
input mesharc
line
input mlabel
point
q
&end
&end
clean %mesh%
build %mesh% poly
build %mesh% line

/* -Create the label point output cover-R
&if %.computer_type% = 'prime' &then
&do
generate %meshlab%
input mlabel
point
q
&end
&else
&do
&data arc generate %meshlab%
input mlabel
point
q
&end
&end
```

```
build %meshlab% point

/* -Delete the ASCII output files-
&s i [delete mesharc]
&s i [delete mlabel]

/* -Create the node point output file-
&if %.computer_type% = 'prime' &then
&do
generate %nodecrd%
input filencd
&severity &error &ignore
point
&severity &error &fail
q
&end
&else
&do
&data arc generate %nodecrd%
input filencd
&severity &error &ignore
point
&severity &error &fail
q
&end
&end
build %nodecrd% point

&severity &error &ignore
&s i [delete nescdata]
&s i [delete numbrs]
&severity &error &fail
&goto endit

&label badentry
&type Usage:  REMODEL <max # of optimizing iterations> <# of nodes> <# of elements>
&type              <name of output printedfiles(created)><should the FILENCD and FILEECD files be
&type              optimized? (y/n)> <output mesh polygon cover (created)>
&type             <output label point cover (created)> <output node point
&type              cover (created)>
&label endit

&type End of REMODEL
```

Fortran Program REOPT.F77

Description

This program is a shell used to call OPTIMIZE.F77 (fig. 50). It simply reads the ASCII file input from REMODEL.AML and then calls OPTIMIZE.F77. When OPTIMIZE.F77 is finished running, REOPT.F77 returns control to REMODEL.AML for the cover creation. It is run only from within REMODEL.AML.

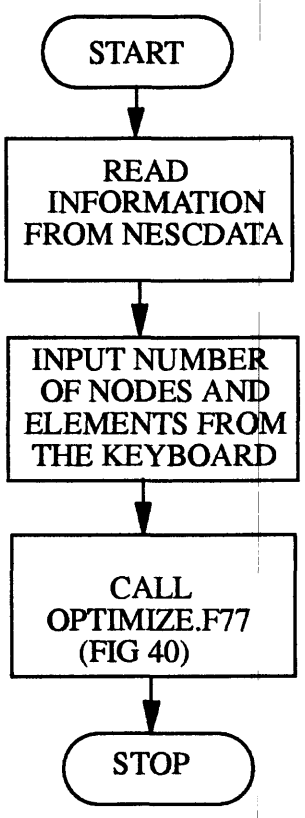


Figure 50.--Flowchart for REOPT.F77.

Program listing

```
C*************************************************************C
C   PROGRAM: Re-optimize the NCD and ECD files                C
C   CODED BY: Robert Lowther                                  C
C   SUPERVISED BY: Eve Kuniansky modified 11-09-92            C
C*************************************************************C

    PROGRAM REOPT

C*****************************************************************C
C   VARIABLE LIST                                                 C
C                                                                 C
C       ARGS: THE DUMMY ARGUMENT PASSED INTO AND OUT OF REOPT     C
C      COVER: THE NAME OF THE COVER TO BE OPTIMIZED               H
C     MAXITR: THE MAXIMUM NUMBER OF OPTIMIZING ITERATIONS         S
C      FPRINT: THE NAME OF THE INFORMATION PRINTOUT FILE          C
C     NUMNOD: THE NUMBER OF NODES IN THE NODE COVER               C
C       NUMEL: THE NUMBER OF ELEMENTS IN THE ARC COVER            C
C*****************************************************************C

C   VARIABLES contains the common statements and corresponding
```

```
C    variable definitions which are used by the subroutines.

     INCLUDE 'variables'

     character*128 dum,nnstring,nestring

C=====INPUT INFORMATION FROM NESCDATA==================C
     OPEN (7,file ='nescdata',status ='OLD',recl = 60)
98   format (I5)
100  FORMAT (A10)
     READ (7,100) COVER
     READ (7,*) MAXITR
     READ (7,100) FPRINT
     READ(7,*) numnod
     READ(7,*) numel
     CLOSE (7)

C=====OPTIMIZE THE NODE NUMBERING SCHEME================C
     PRINT *,'Optimizing the node numbering scheme...'

C    OPTIMIZE uses the previous two routines' output files as its
C    input. They are read into input arrays, where calculations
C    are performed on them. The output array is then written to
C    a user-specified output file.

     CALL OPTIMIZE

C=====EXIT THE PROGRAM==========C
     BAD = 'OK'
     END
```

SNAPPY.AML

Description

This macro corrects a problem that exists in the ARC command *SNAP*. It deals only with point coverages, otherwise operating in the same manner as *SNAP* (fig. 51).

SNAP, using the *CLOSEST* option, will *SNAP* to any point the point that lies closest, so long as that point is within the *SNAP* distance. The problem with this command arises when two or more points are within the *SNAP* distance of a specified point. The closest point that the routine encounters will be *SNAP*ped to the specified point first. The point is moved to the specified point's location rather than deleted. This means that after the operation, two points lie at the same location.

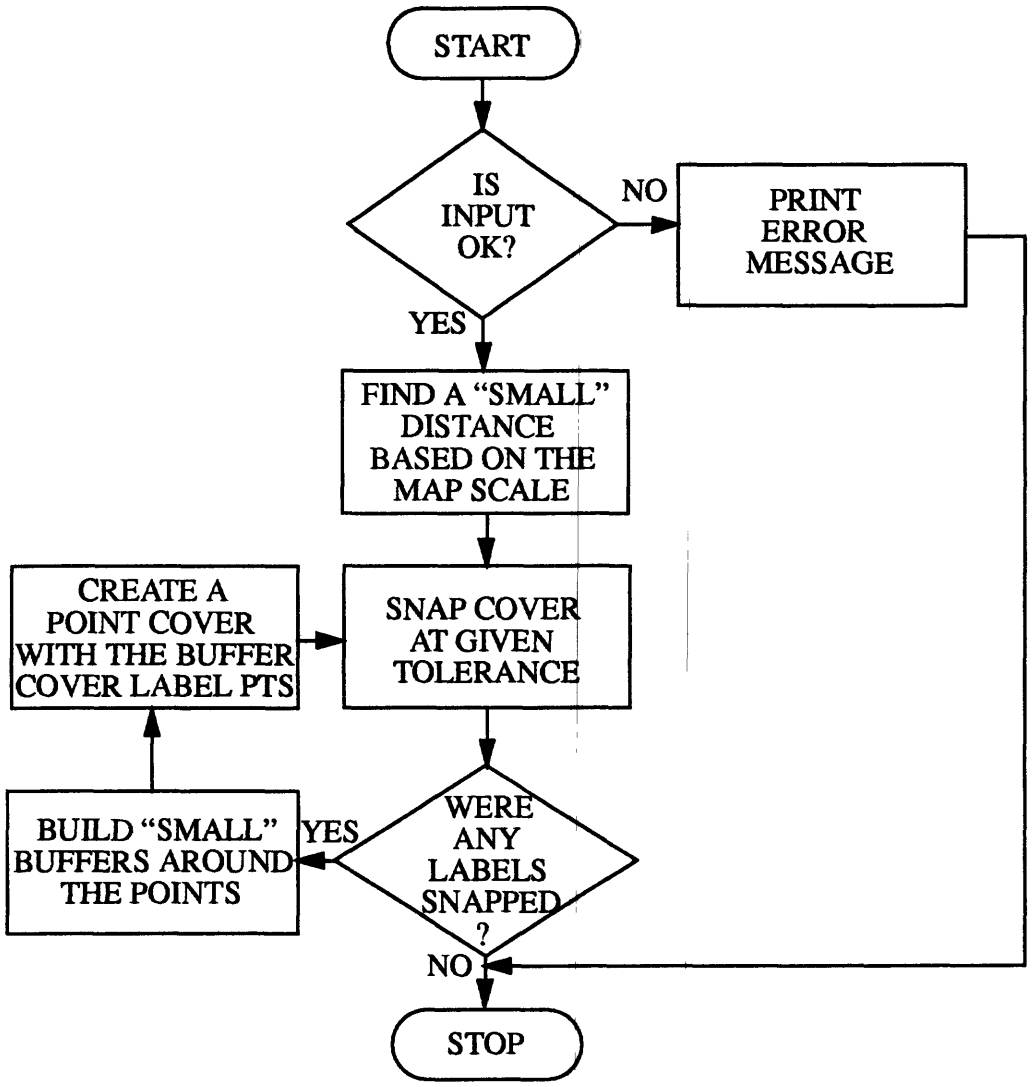
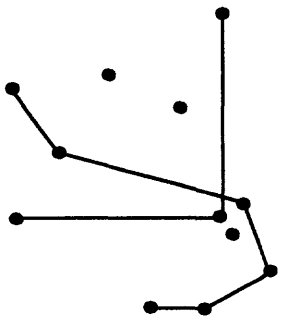


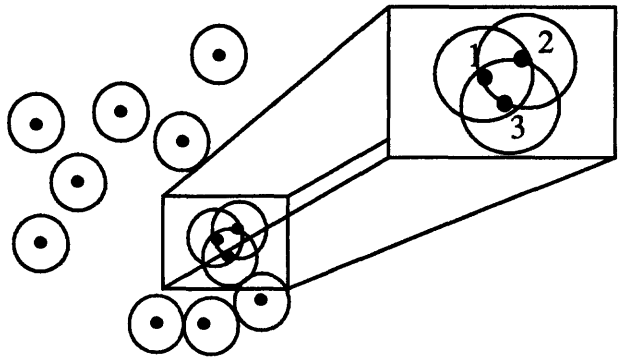
Figure 51.--Flowchart for SNAPPY.AML.

It is this overlaying of points that causes the problem. The second point lying within the *SNAP* distance of the specified point will not be *SNAP*ped because the point that, due to *SNAP*ping, shares a location with the specified point, is unquestionably the closest point. Hence, the second, further point will not *SNAP* to the specified point. The *SNAP* process will, therefore, leave all points except one that lie within the *SNAP* distance of the specified point.

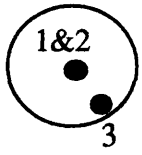
SNAPPY.AML corrects this by a recursive process. After performing a *SNAP* on the input cover using a normal, large *SNAP* distance, SNAPPY.AML creates very small (a relative term based on the cover's map scale) *BUFFER*s around each of the points. Because the *BUFFER*ing operation merges overlapping polygons, only one *BUFFER* polygon is created wherever two or more points lie at the same location. SNAPPY.AML then creates a point cover based on the polygon cover's polygon labels, labels that lie at the center of the *BUFFER* polygons and hence the locations of the original points. SNAPPY.AML performs this entire process iteratively until no point is within the *SNAP* distance of other points. This process is shown in figure 52.



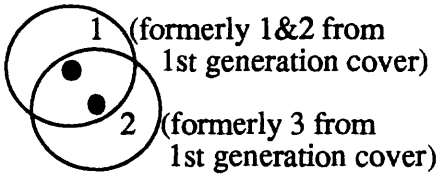
a: Original cover



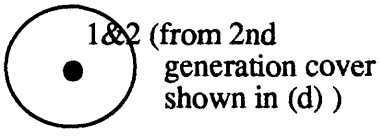
b: Point cover with SNAP distances shown



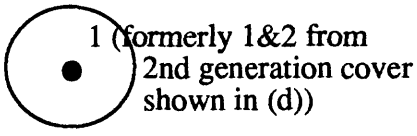
c: Cover after one SNAP, with multiple points and an unSNAPped third point



d: The new, renumbered point cover with buffers shown and duplicate points eliminated



e: The new cover after SNAPping, with SNAP distance (buffer) and new duplicate points shown



f: The final cover, with no duplicate points remaining

Figure 52.--Diagram of the recursive process in SNAPPY.AML.

One further note is necessary concerning SNAPPY.AML. SNAPPY.AML, unlike the rest of the AML programs, requires monitoring while it runs. It requires user input concerning the effects of the *SNAP* command during each iteration. The necessary information is printed by the system as a normal function of *SNAP*. This inconvenience is necessary because we have not discovered how to access the output data from the *SNAP* command directly.

SNAPPY.AML *KILL*s the cover called "WEEBUF" at the beginning of its run. One common error in using SNAPPY.AML is to use a map scale that causes the *BUFFER*s that SNAPPY.AML creates to be too big and hence to overlap each other. This will cause SNAPPY.AML to eliminate too many points in the cover.

If the input cover contains approximately 10,000 or more points, then SNAPPY.AML may not run correctly. Trying to create that many circles will overload the *BUFFER* command. One solution to this problem is to use SNAPPY.AML to *SNAP* the points, then *UNGENERATE* the point cover and use a statistical or database software package to remove all duplicate points. *GENERATE* can be used to return this file to an ARC/INFO cover.

Program Listing

```
/* MACRO:  Take an input cover and use snapping as many times as necessary
/*              to eliminate all points which lie too close to other points.
/* CODED BY:  Robert Lowther
/* SUPERVISED BY:  Eve L. Kuniansky

/* VARIABLE LIST:
/*     DTYPE:  The graphic display terminal type code
/*       COV:  The cover to be snapped
/*       SKALE:  The map scale, in feet, of the cover to be snapped
/* TOLERANCE:  The maximum distance across which snapping may occur
/*          DONE:  The user response indicating whether or not further
/*                 snapping is necessary
/*      WEEBUF:  The cover containing very small buffered areas surrounding
/*                 each of the point locations from cov

&echo &off
&args dtype cov skale tolerance

/* -Prepare the error-indication file-C
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% 7]
&setvar i [close %filun%]

/* -Test the input to see if all arguments are present as expected-O
&if [type %dtype%] ne 1 &then &goto proceed
&if [len %dtype%] eq 0 &then &goto badentry
&label proceed
&if [type %cov%] ne 1 &then &goto badentry
&if [length %cov%] eq 0 &then &goto badentry
&if [type %skale%] ne -1 &then &goto badentry
&if [type %tolerance%] ne -1 &then goto badentry

/* -Eliminate unecessary coverages-M
&severity &error &ignore
kill weebuf all
```

```
&severity &error &fail

/* -Establish a relatively "small" distance based on the map scale-E
&setvar bufdis %skale% * 0.001

/* -Snap the point cover and check to see if further snapping is necessary-D
&label ohlamour
ae
mape %cov%
disp %dtype%
editc %cov%
editf label
drawe label
draw
select screen
snapc %cov%
snapf label label
snapping closest %tolerance%
snap
&setvar done [response 'Does the above line say:  0 label(s) snapped?']
save
q
&if %done% eq 'Y' &then &goto endit
&if %done% eq 'y' &then &goto endit
&if %done% eq 'YES' &then &goto endit
&if %done% eq 'yes' &then &goto endit

/*  -Use buffers in order to remove multiple points with the same
/*   location, a condition which prevents the SNAP command from
/*   working properly-Y
remepf -prg -na -nq -nvfy
build %cov% point
&SEVERITY &ERROR &IGNORE
buffer %cov% weebuf # # %bufdis% 40 point
&SEVERITY &ERROR &FAIL
build weebuf point
kill %cov% all
copy weebuf %cov%
kill weebuf all
&goto ohlamour

/*  -Print the error message-
&label badentry
&type Usage:  SNAPPY <display type> <point cover to be snapped (existing)><map
&type              scale as if on a 24" plotter (ft)> <snapping tolerance>
&goto enderr

&label endit
/*  -Prepare the error-indication file-
&s i [delete coderr]
&setvar filun [open coderr openstatus -w]
&setvar i [write %filun% noerr]
&setvar i [close %filun%]
```

```
&label enderr
&type End of SNAPPY
```

SPLEEN.AML

Description

This macro is an extension of the ARC command *SPLINE*. In fact, its name represents a whimsical plural of *SPLINE*. SPLEEN.AML will *SPLINE* up to 10 coverages in a single run (fig. 53). Also, it allows up to three different *SPLINE* distances. Multiple *SPLINE* distances require subareas of the cover to be specified with polygon outlines. These outlines should have identifying items. The general *SPLINE* distance applies to all areas not within either subarea, while the two priority *SPLINE* distances apply to features that lie in the respective subareas. The highest priority *SPLINE* area may lie within the higher priority *SPLINE* area.

The most common error in using SPLEEN.AML is to not define the polygon outlines properly, either by not using polygon coverages or by not having an identifying item in the cover's PAT.

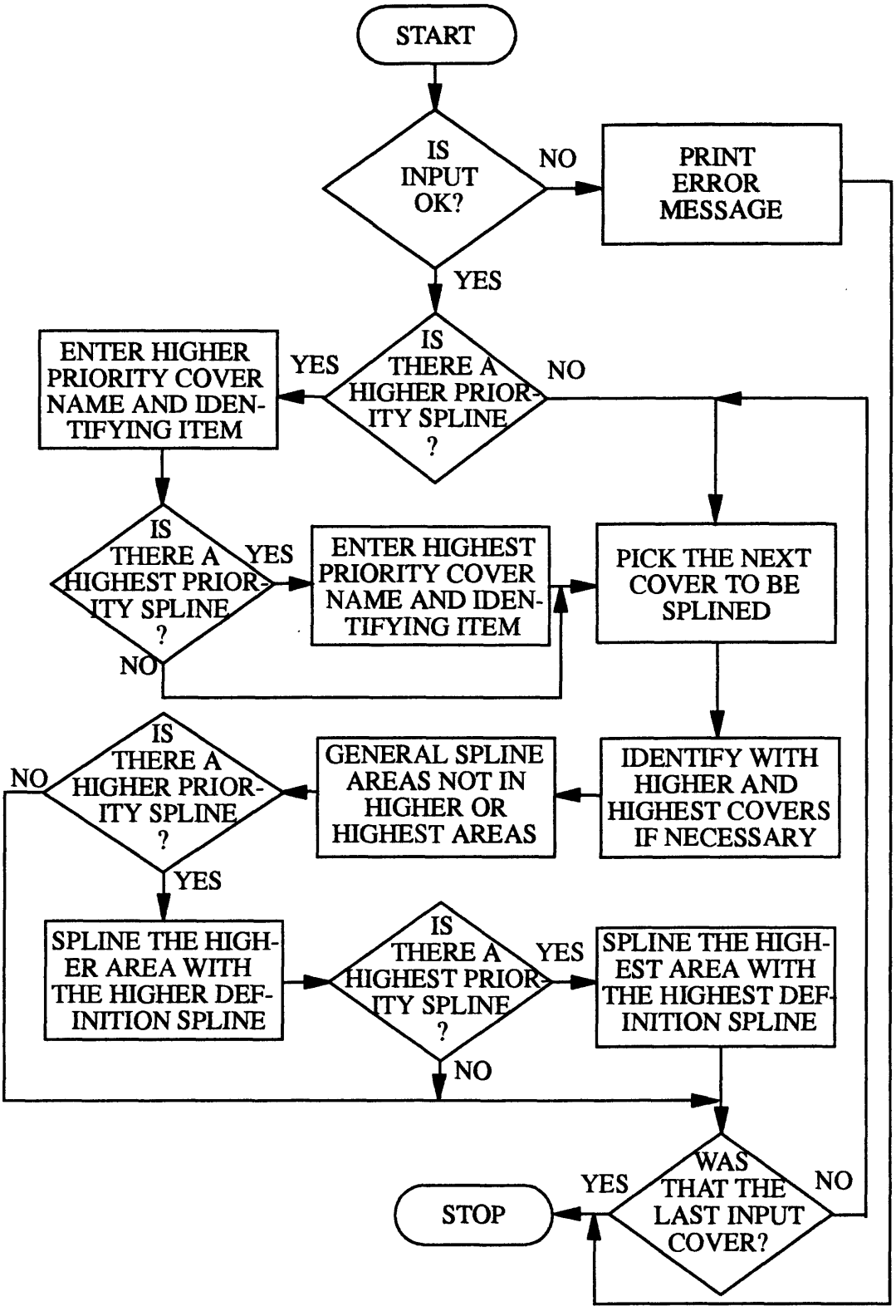


Figure 53.--Flowchart for SPLEEN.AML.

```
/*   MACRO:  SPLINE up to ten map coverages at up two three different
/*                distances, in areas defined by polygonal outlines.
/*   CODED BY:  Robert Lowther
/*   SUPERVISED BY:  Eve L. Kuniansky

/*   VARIABLE LIST:
/*        DTYPE:  The graphic terminal display type code
/*        DIST1:  The lowest priority (largest) splining distance
/*        DIST2:  The higher priority (smaller) splining distance
/*        DIST3:  The highest priority (smallest) splining distance
/*  COV1-COV10:  The names of the line coverages to be splined
/*         MANY:  Logical variable indicating that there is more than
/*                one splining distance
/*        THREE:  Logical variable indicating that there are three
/*                splining distances
/*      OUTLINE:  Name of the cover containing the outline of the
/*                higher priority spline area
/*         ITEM:  Name in the outline PAT which indicates whether or
/*                not an area is within the outline
/*     OUTLINE3:  Same as outline for the highest priority area
/*        ITEM3:  Same as item for outline3
/*          COV:  The name of the cover currently being considered
/*         COVD:  The name of the intersection of a splining outline and cov

&echo &off
&args dtype dist1 dist2 dist3 cov1 cov2 cov3 cov4 cov5 cov6 cov7 cov8 cov9 ~
    cov10

/* -Test input to see if all arguments are present as expected-C
&if [type %dtype%] ne 1 &then &goto proceed
&if [length %dtype%] eq 0 &then &goto badentry
&label proceed
&if [type %dist1%] ne -1 &then &goto badentry
&if [type %dist2%] ne -1 &then &goto badentry
&if [type %dist3%] ne -1 &then &goto badentry
&if [type %cov1%] ne 1 &then &goto badentry
&if [length %cov1%] eq 0 &then &goto badentry

&setvar many .FALSE.
&setvar three .FALSE.

/* -If needed, obtain input concerning areas to be splined differently-O
&if %dist3% eq 0 &then &goto checktwo
&setvar three .TRUE.
&setvar outline3 [response 'Enter name of polygon around smallest splined area']
&setvar item3 [response 'Enter item name in PAT which designates this area']
&label checktwo
&if %dist2% eq 0 &then &goto spline
&setvar outline [response 'Enter name of polygon around smaller splined area']
&setvar item [response 'Enter item name in PAT which designates this area']

/* -Prepare the map coverages for splining at different lengths by
```

```
/*  overlaying the necessary polygonal outlines-M
&label setcoverages
&setvar many .TRUE.
&do cov &list %cov1% %cov2% %cov3% %cov4% %cov5% %cov6% ~
  %cov7% %cov8% %cov9% %cov10%
&if [length %cov%] eq 0 &then &goto endl1
&setvar covd %cov%d
&severity &error &ignore
ae
 disp %dtype%
 editc %cov%
 editf arc
 drawe arc node
 draw
 select screen
 unsplit none
 save
q
identity %cov% %outline% %covd% line 40
kill %cov% all
&if %three% eq .TRUE. &then identity %covd% %outline3% %cov% line 40
&severity &error &fail
&if %three% eq .FALSE. &then copy %covd% %cov%
kill %covd% all
&label endl1
&end

/*  -Spline the coverages-E
&label spline

&do cov &list %cov1% %cov2% %cov3% %cov4% %cov5% %cov6% ~
  %cov7% %cov8% %cov9% %cov10%
&if [length %cov%] eq 0 &then &goto realendloop
&setvar covd %cov%d
ae
mape %cov%
disp %dtype%
editc %cov%
editf arc
drawe arc
draw

/*  -Do the general spline-D
select screen
&if %many% eq .TRUE. &then resel for %item% eq 0
&if %three% eq .TRUE. &then resel for %item3% eq 0
grain %dist1%
&severity &error &ignore
spline
&severity &error &fail
&if %many% eq .FALSE. &then &goto endloop

/*  -Do the higher priority spline-Y
select screen
```

```
resel for %item% eq 1
&if %three% eq .TRUE. &then resel for %item3% eq 0
grain %dist2%
&severity &error &ignore
spline
&severity &error &fail
&if %three% eq .FALSE. &then &goto endloop

/* -Do the highest priority spline-H
select screen
resel for %item3% eq 1
grain %dist3%
&severity &error &ignore
spline
&severity &error &fail
&label endloop
save
q

/* -Remove unecessary, added items from the coverages' AAT's-O
&setvar covpat %cov%.aat
&if %many% eq .FALSE. &then &goto realendloop
dropitem %covpat% %covpat% AREA
dropitem %covpat% %covpat% PERIMETER
&setvar dummy1 %outline%#
&setvar dummy2 %outline%-ID
dropitem %covpat% %covpat% %dummy1%
dropitem %covpat% %covpat% %dummy2%
dropitem %covpat% %covpat% %item%
&if %three% eq .FALSE. &then &goto realendloop
&setvar dummy1 %outline3%#
&setvar dummy2 %outline3%-ID
dropitem %covpat% %covpat% %dummy1%
dropitem %covpat% %covpat% %dummy2%
&setvar dummy1 %covd%#
&setvar dummy2 %covd%-ID
dropitem %covpat% %covpat% %dummy1%
dropitem %covpat% %covpat% %dummy2%
dropitem %covpat% %covpat% %item3%
&label realendloop
&end
&goto endit

/* -Error handling routine-U
&label badentry
&type Usage:  SPLEEN <display terminal type> <general spline distance>
&type       <smaller spline distance (or 0 if not using a smaller distance)>
&type       <smallest spline dist (or 0 if not using a smallest distance)>
&type       <cover #1 name (existing)>{{cover #2 name (existing)} ... {cover
&type        #10 name (existing)}

/* -End message-R
&label endit
&type End of SPLEEN
```

TRIANGLE.AML

Description

TRIANGLE.AML creates an ARC point cover based on coordinates calculated by the Fortran77 program, TRIANGRID.F77 (fig. 54). It allows the user to graphically enter the extent of the point cover by specifying points in relation to any given background cover. The distance between points may be indicated graphically against the same background or given numerically as part of the macro arguments. This information is stored in a file that can be read by TRIANGRID.F77.

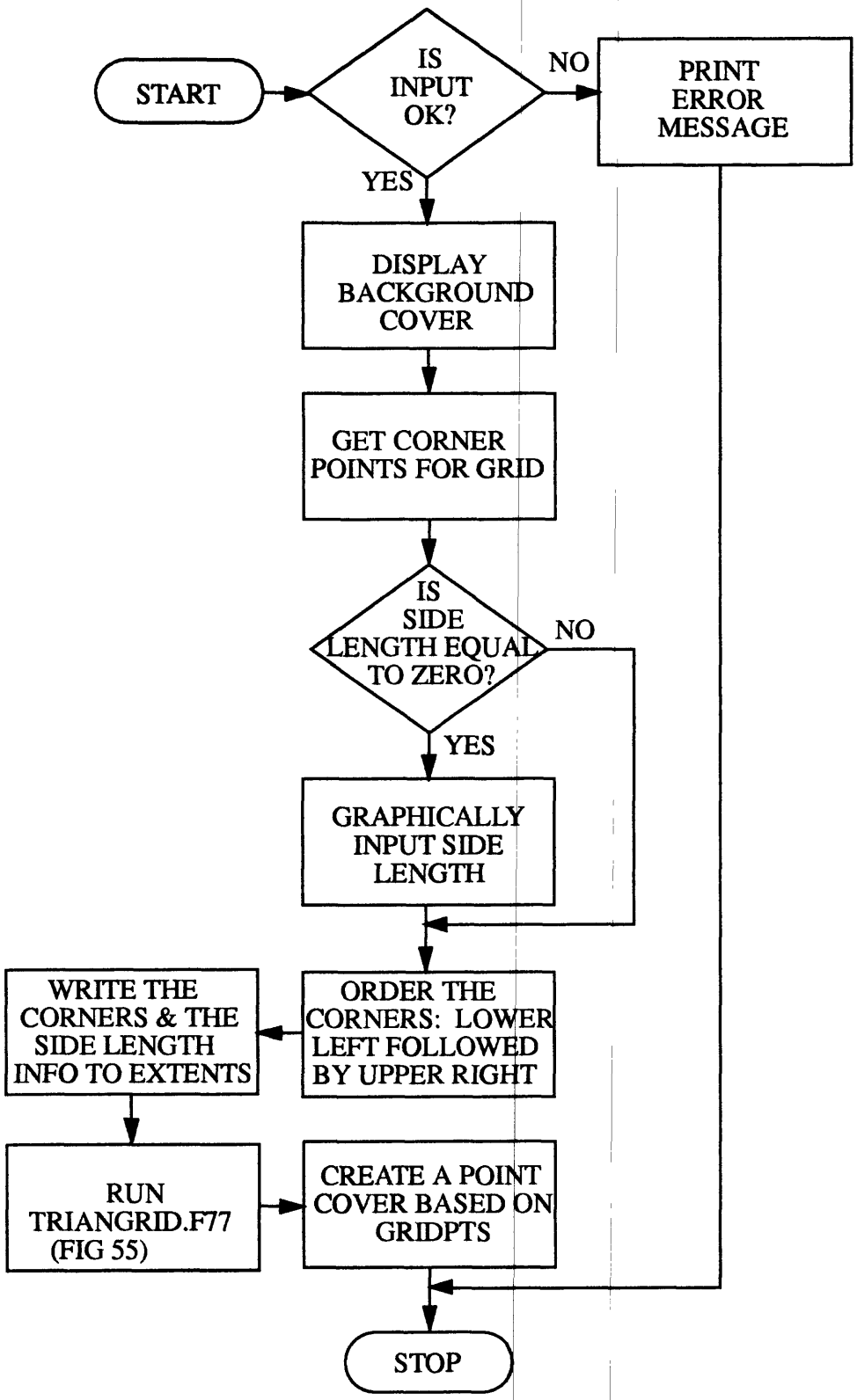


Figure 54.--Flowchart for TRIANGLE.AML.

```
/*    MACRO: Graphically input data for, and then run TRIANGRID, followed
/*                by a conversion of TRIANGRID's output into a triangular mesh
/*    CODED BY: Robert Lowther
/*    SUPERVISED BY: Eve L. Kuniansky
/*    SECOND MODIFICATION BY: ELK 8-21-90
/*    modified to run triangrid.out as &sys command not functiontaskELK031392

/*    VARIABLE LIST:
/*        DTYPE: The graphic display terminal type code
/*        COVER: The name of the cover to be used as a background when
/*                indicating the location of the grid and its point spacing
/*          POIN: The name of the polygon cover into which the grid
/*          SIDE: The length of a triangle side (put in decimal point).If side
/*                is left blank (null), you can put it in from the screen
/*          X1,Y1: The lower left rectangular boundary of the grid
/*          X2,Y2: The upper right rectangular boundary of the grid
/*X3,Y3 & X4,Y4: The points defining the length of the triangle sides
/*           LEN: The desired size of the triangle sides
/*      FILUNIT: The file reference number of the TRIANGRID file

&echo &off
&args dtype cover side poin

/* -Check the computer type (by Leonard L. Orzol)-C
&s .path [show &workspace]
&s .slash /
&s computer_flag [index %.path% %.slash%]
&if %computer_flag% <= 0 &then
 &do
  &s .slash >
  &s .computer_type prime
  &s program triangrid.run
 &end
&else
 &do
  &s .computer_type unix
  &s program triangrid.out
 &end

/* -Test input to see if all arguments are present as expected-O
&if [length %cover%] eq 0 &then &goto badentry
&if [length %poin%] eq 0 &then &goto badentry
&if [type %dtype%] ne 1 &then &goto proceed
&if [type %side%] gt 0 &then &goto badentry
&if [length %poin%] eq 0 &then &goto badentry

&label proceed
&severity &error &ignore
kill %poin% all
&severity &error &fail
&s i [delete extents]
&s i [delete gridpts]
```

```
&if [exists %poin% -COVER] &then kill %poin% all

/* -Use Arcplot to graphically input the extent and triangle
/* size length of the grid-M

arcplot
disp %dtype%
mape %cover%
arcs %cover%
&type Enter a corner coordinate for the desired grid area
&getpoint &map &cursor
&setvar x1 %pnt$x%
&setvar y1 %pnt$y%
&type Enter the opposite corner coordinate for the desired grid area
&getpoint &map &cursor
&setvar x2 %pnt$x%
&setvar y2 %pnt$y%
&if %side% ne 0 &then &goto skipthis
&type Enter two points defining the desired length of the triangle sides
&getpoint &map &cursor
&setvar x3 %pnt$x%
&setvar y3 %pnt$y%
&getpoint &map &cursor
&setvar x4 %pnt$x%
&setvar y4 %pnt$y%
&label skipthis
q
&if %side% ne 0 &then &goto skipthistoo
&setvar len [invdistance %x3% %y3% %x4% %y4%]
&label skipthistoo
&if %side% ne 0 &then &setvar len %side%
&if %x1% < %x2% &then &goto checky
&setvar temp %x1%
&setvar x1 %x2%
&setvar x2 %temp%
&label checky
&if %y1% < %y2% &then &goto getonwithyourlife
&setvar temp %y1%
&setvar y1 %y2%
&setvar y2 %temp%
&label getonwithyourlife
&label enterlength

/* -Write the information to a file which can be read by TRIANGRID-E
&setvar filunit [open extents openstatus -w]
&setvar i [write %filunit% %x1%]
&setvar i [write %filunit% %y1%]
&setvar i [write %filunit% %x2%]
&setvar i [write %filunit% %y2%]
&setvar i [write %filunit% %len%]
&if [close %filunit%] = 0 ~
  &then &type File created successfully.

/* -Create the equilateral triangular grid points-D
```

```
&if %.computer_type% = 'prime' &then &sys r triangrid
&else &sys triangrid.out

&if %.computer_type% = 'prime' &then
 &do
  generate %poin%
  input gridpts
  points
  quit
 &end
&else
 &do
  &data arc generate %poin%
   input gridpts
   points
   quit
  &end
 &end
build %poin% point
&goto endit

/* -Print error message-Y
&label badentry
&type Usage: TRIANGLE <display type> <background cover name(existing)><length
&type              of side (0 to enter graphically)> <output point cover
&type              name (created)>
&label endit
&type End of TRIANGLE
```

Fortran Program TRIANGRID.F77

Description

This program is an alternative to the *GENERATE GRID* command in ARC. Instead of generating a rectangular grid, TRIANGRID.F77 creates points representing an equilateral triangular grid (fig. 55). It reads the grid extent and the triangle side length from the file created by TRIANGLE.AML. Because equilateral triangles do not form a shape that will exactly overlay a rectangular extent, a new extent must be used. TRIANGRID.F77 calculates the smallest extent into which equilateral triangles will fit precisely, and which contains the extent entered in TRIANGLE.AML. It writes the output points to a file called "gridpts." This file can then be read by TRIANGLE.AML program in order to create an ARC point cover.

Program listing

```
C*************************************************************C
C    PROGRAM: Create equilateral triangular pattern of points      C
C    CODED BY: Robert Lowther                                      C
C    SUPERVISED BY: Eve L. Kuniansky                               C
C    modified to run as a system call from the aml ELK 031392      C
C*************************************************************C
```

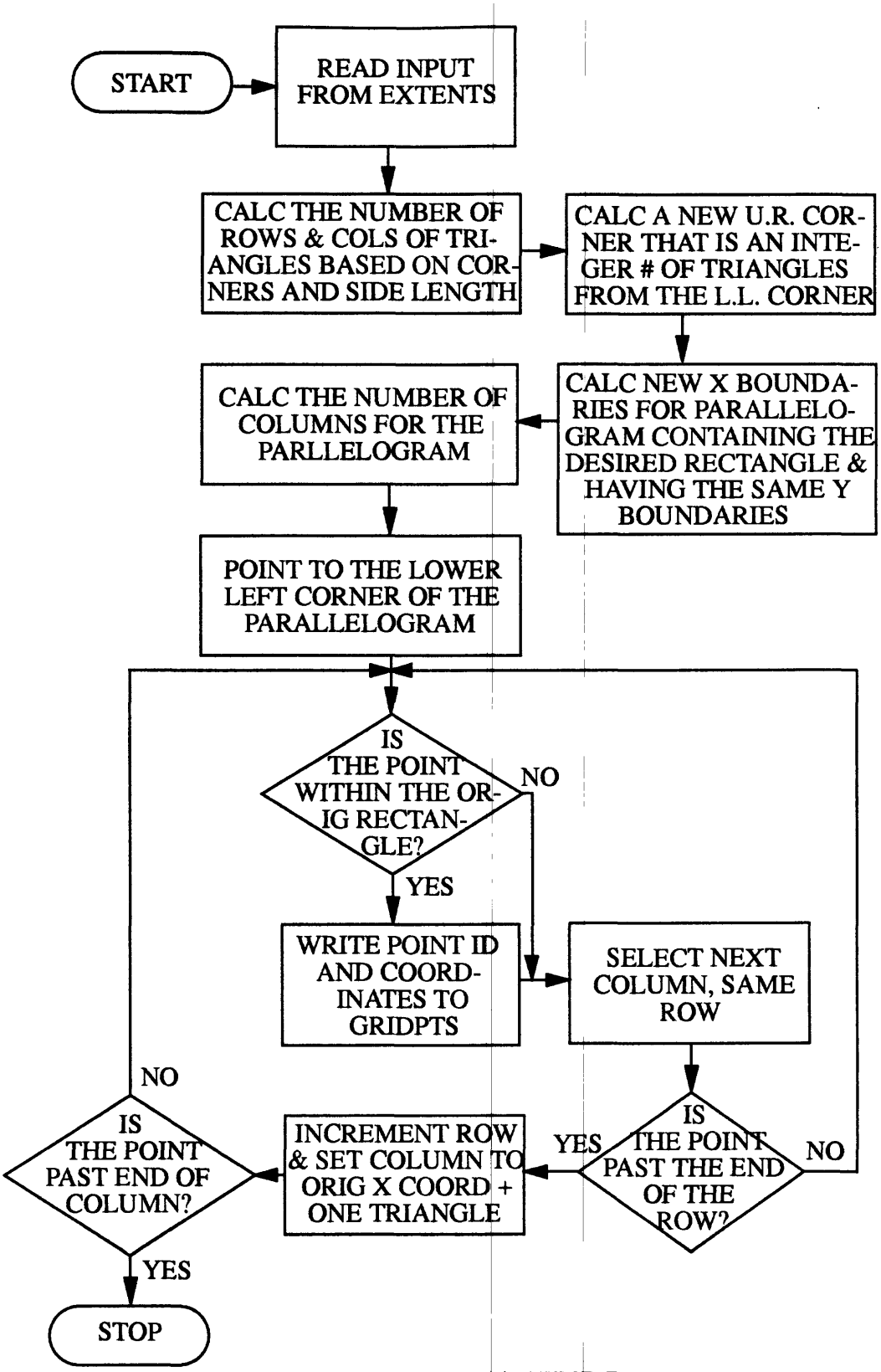


Figure 55.--Flowchart for TRIANGRID.F77.

```
      PROGRAM TRIANGRID

C*************************************************************C
C    VARIABLE LIST:                                           H
C                                                             S
C         NUMX:  The number of points in the parallelogram to be created   C
C                which lie along the X axis                   C
C         NUMY:  The number of points along the Y axis         C
C            I,J: counters                                    C
C           PTID:  The identifying point number for each point C
C           LLX:  The lower left x-coordinate for the desired map area  C
C           LLY:  The lower left y-coordinate                 H
C           URX:  The upper right x-coordinate                 S
C           URY:  The upper right y-coordinate                 C
C           SIDE:  The desired length of each triangle side    C
C          CURX:  The x-coordinate of the point currently being checked  C
C          CURY:  The y-coordinate of the point currently being checked  C
C          OLLX:  The x-coordinate just outside of the lower left  C
C          OLLY:  The y-coordinate just outside of the lower left  C
C OURX & OURY:  Similar to above for the upper right         H
C*************************************************************S

      INTEGER NUMX,NUMY,I,J,PTID
      REAL*8 LLX,LLY,URX,URY,SIDE,CURX,CURY,OLLX,OLLY,OURX,OURY

C======Open the output file=========================================C

      OPEN (5,FILE= 'gridpts', STATUS= 'NEW', ACCESS= 'SEQUENTIAL',
     C  FORM= 'FORMATTED', RECL= 377,ERR= 300)
      open (6,file = 'extents', status='OLD',recl= 60)

C======Format statements============================================C

  104   FORMAT (I7,2F15.3)
  105   FORMAT ('ERROR OPENING GRIDPTS.  PERHAPS IT ALREADY EXISTS.')
  106   FORMAT (F15.3)
  107   FORMAT ('END')

C======Input the rectangular area boundaries=======================C

      PTID = 1
      READ (6,106) LLX
      READ (6,106) LLY
      READ (6,106) URX
      READ (6,106) URY
      READ (6,*) SIDE
      NUMX = NINT((URX-LLX)/SIDE) + 1
      NUMY = NINT((URY-LLY)/(SIDE*SIN(1.047197551))) + 1

C======Create a slightly larger rectangle with which to compare point coords=C

      OLLX = LLX - 1
      OLLY = LLY - 1
      OURX = URX + 1
```

```
      OURY = URY + 1

C=====Recalculate the upper right corner based upon the triangle side length=C

      URX = LLX + (NUMX-1)*SIDE
      URY = LLY + (NUMY-1)*(SIDE*SIN(1.047197551))

C=====Calculate X boundaries for a parallelogram containing the rectangle===C

      LLX = LLX - (URY-LLY)/TAN(1.047197551)
      URX = URX + (URY-LLY)/TAN(1.047197551)

C=====Recalculate the number of points in the X direction=================C

      NUMX = NUMX + NINT(((URY-LLY)/TAN(1.047197551))/SIDE)

C=====Initialize and begin the point-creating loop=======================C

      CURX = LLX
      CURY = LLY
      DO 10, I=1,NUMY
       DO 20, J=1,NUMX

C=======Check to see if the boundaries have been exceeded================C

       IF ((CURX .LT. OLLX) .OR. (CURX .GT. OURX)
     C   .OR. (CURY .LT. OLLY) .OR. (CURY .GT. OURY)) GO TO 301

C=======Write the point to the output file==============================C

       WRITE (5,104) PTID,CURX,CURY
       PTID = PTID + 1
301      CURX = CURX + SIDE
20     CONTINUE

C=======Increment the y-coordinate and reset the x-coordinate===========C

      CURY = CURY + SIDE*SIN(1.047197551)
      CURX = CURX - NUMX*SIDE + SIDE*COS(1.047197551)
10     CONTINUE

C=====Close the output file and return to the TRIANGLE AML==============C

      WRITE (5,107)
      go to 11
300   print 105
11    stop
      END
```

SELECTED REFERENCES

American National Standards Institute, 1978, Programming language FORTRAN: American National Standards Institute, X3.9-1978, 18 ch.

Collins, R.J., 1973, Bandwidth reduction by automatic renumbering: International Journal for Numerical Methods in Engineering, vol. 6, no. 3, p. 345-356.

Environmental Systems Research Institute, 1987, User guide ARC/INFO volume 1; the geographic information system software: Environmental Systems Research Institute, Redlands, California, 438 p.

Kuniansky, E.L., 1990, A finite-element model for simulation of two-dimensional steady-state ground-water flow in confined aquifers: U.S. Geological Survey Open-File Report 90-187, 77 p.

Strang, Gilbert, and Fix, G.J., 1973, An analysis of the finite-element method: Prentice-Hall, Inc., Englewood-Cliffs, N.J., 306 p.

SUPPLEMENTAL DATA

PART ONE, SELECT IMPORTANT FEATURES

STEP 1.1: Decide what features are important to the model and create the necessary coverages.

STEP 1.2: Decide which features define areas in need of more detail and choose a model boundary.

PART TWO, GENERALIZE FEATURES

STEP 2.1: (a) Create buffers around point and/or line features or
(b) Define areas of greater detail with polygon outlines

(a) If the area is defined to be all space within some distance of certain features (the basis):

COPY (name) (name).TEMP and all others	Make working copy of coverages defining the basis of the area Use *ARCEDIT* to remove unwanted parts of all the .TEMP coverages
APPEND DTAILAREA	*APPEND* all parts of the area basis together into one cover
(name).TEMP and all others	coverages to be *APPEND*ed
BUILD DTAILAREA LINE	*BUILD* the basis as a line
&R BUFFNSHINE	*BUFFER* to create the desired polygon outline (Repeat to create as many outlines from this basis as desired)
KILL (name).TEMP ALL & all others	*KILL* all temporary coverages

(b) If the areas are defined by polygons already present in feature coverages:

COPY (name) (name).TEMP and all others	Make working copy of coverages defining the basis of the area Use *ARCEDIT* to remove unwanted parts of all the .TEMP coverages
APPEND DTAILAREA	*APPEND* all parts of the area outline together into one cover
(name).TEMP and all others	coverages to be *APPEND*ed
BUILD DTAILAREA LINE	*BUILD* the outline as a line

&R MAKOUTLIN	Make a polygon outline from the appended coverages
KILL (name).TEMP ALL & all others	*KILL* all temporary coverages

STEP 2.2: Define the study area with a model boundary

Use the same process as in step three, using MAKOUTLIN.AML rather than BUFFNSHINE.AML to create a polygon outline.

STEP 2.3: Generalize the line data by splining (*SPLINE*)

COPY (stream cover name) (stream cover name).LEN	Make a full-detail copy of any stream cover
&R SPLEEN	Use SPLEEN.AML to *SPLINE* all line coverages. Use the polygon outlines defined in step three

STEP 2.4: Create archive copies

COPY (name) (1st three letters of name).A and all others	Use *COPY* to make an archive copy of all with the naming convention shown

STEP 2.5: *CLIP* all coverages with the model boundary

&R CLIPIT line coverages &R CLIPIT point coverages	Use CLIPIT.AML, once for lines and once for points, to *CLIP* all with the model boundary

STEP 2.6: Interactively edit each of the input model coverages

Use *ARCEDIT* to eliminate any short, meaningless line segments created by the *CLIP*ping process of step 2.5

STEP 2.7: Create a grid of regularly spaced points

&R TRIANGLE as many times as necessary	Run once for each size of grid desired
&R CLIPIT as many times as necessary	*CLIP* each grid with the model boundary and *CLIP* within or without each of the polygon outlines created in step three
APPEND and *BUILD*	Combine the component grid coverages

PART THREE, GENERATE A MESH

STEP 3.1: Run KITSINK.AML to finish the process

&R KITSINK	Run KITSINK.AML, which requires user input at the beginning of its run and a few minutes into its run

STEP 3.2: Run ELEVATE.AML to assign elevations to any features (optional)

&R ELEVATE	Run ELEVATE.AML, which assigns elevations to any points with the specified identifying item. Requires a cover of topographic map points

ARCPOTIN.AML

This macro automatically adds and then deletes the requisite items to coverages for use of the *ARCPOINT* or the *ARCTIN* commands. Optionally, if the *ARCTIN* command is used, it can create a line cover based on the command's output TIN.

Usage: ARCPOTIN <TIN command> <in_cover (existing)> <out_cover or
out_tin (created)> <type of input cover (point,line)>
{output polygon cover name if ARCTIN is used and a poly
cover is to be created from the out_tin (created)}

BUFFNSHINE.AML

BUFFNSHINE.AML *BUFFER*s coverages and adds an identifying item to the output cover. It optionally will remove features from a second cover that fall within the *BUFFER*ed areas of the output cover.

Usage: BUFFNSHINE <name of cover to be BUFFERed (existing)> <type of
cover (line,point,poly)> <BUFFER distance> <name of
item to be added to output poly cover to indicate
areas inside the polygon (created)> {cover whose
features lying within the output polygon are to be
removed (existing)} {cover type (line,point,poly)}
{display type}

CHICPOX.AML

This macro creates an output point cover based upon the vertices of an input point cover. It ensures a perfect match between the input and output coverages by *SNAP*ping the input arcs to the output points.

Usage: CHICPOX <display type> <input line cover(existing)> <maximum
distance output points might deviate from input arcs>
<output point cover (created)>

CLIPIT.AML

This macro will *CLIP* up to 10 coverages of the same type at a time. It will perform either a normal *CLIP* (IN) or an inverse *CLIP* (OUT). Optionally, it will *BUFFER* the *CLIP*ping cover and remove all input cover features that fall inside of the *BUFFER* region.

Usage: CLIPIT <display type> <CLIPping cover (existing)> <item denoting
area inside the CLIPping cover (existing)> <distance from
the edge within which features should be removed (or 0 if
no removal desired)> <type of coverages to be CLIPped (line,
point)> <cover #1> <in/out #1>....{cover #10} {in/out #10}

CLIPIT2.AML

This macro is used only within KITSINK.AML. It has no user applications.

ELEVATE.AML

This macro assigns elevations to points within a point cover. The points to which elevations are to be assigned should be relatively near points taken from a topographic map that have known elevations.

Usage: ELEVATE <display type> <output cover to be elevated (existing)>
<output item indicating points to be elevated (existing)>
<output cover item to contain elevation (created)> <input
cover w/ elevation points near the points to be elevated
(existing)> <input item containing elevation (existing)>

FIXSNAP.AML

This macro eliminates points that lie extremely close to other points in large point cover. These points typically lie so close to the other points that they will not be affected by the *SNAP* command or by SNAPPY.AML.

Usage: FIXSNAP <display type> <point cover (existing)> <SNAPping tolerance>

FREUD.AML

This macro simplifies the relation between an arc cover and the associated point cover. It causes the from- and to-node numbers in the arc cover to correspond to the node numbers in the point cover.

Usage: FREUD <point cover with correct node numbers (existing)> <arc
cover whose from and to nodes are to be changed (existing)>

IDENTIFY.AML

This macro adds user-specified identifying items to up to 10 coverages. Optionally, if the coverages are polygon coverages, it will *IDENTITY* the 10 coverages with any specified cover, thereby identifying the regions of that cover.

Usage: IDENTIFY <type (point,line,poly) of coverages to have identifying
items added> <cover to have item added #1 (existing)>
<identifying item #1 (created)> ... {cover to have
item added #10 (existing)} {item #10 (created)}

IDENTI2.AML

This macro is used only within KITSINK.AML. It has no user applications.

IDENTILOTS.AML

This macro *BUFFER*s up to 10 coverages, adds user-specified identifying items to each, and then optionally intersects the coverages with a target cover.

Usage: IDENTILOTS <point cover to be identified by multiple others
(existing)> <minimum expected distance between points>

KITSINK.AML

This macro takes input coverages of all types, a model boundary, and a regularly spaced point grid, and creates an output mesh with triangular elements.

Usage: KITSINK <display type> <master feature pnt cover (created)>
<mapscale as if to plot on a 24" plotter (ft)>
<min distance between grid & feature pts> <master grid (existing)> <cover w/ all points (created)>
<study area model boundary poly cover name (existing)>
<item name denoting area inside model boundary (existing)>
<min dist between interior and edge points>
<master point cover (created)> <root name for output mesh and node coverages (created)> <max # of optimizing iterations>

MAKOUTLIN.AML

This macro creates a *CLIP*ping cover, complete with an identifying item, from any polygon cover. Optionally, it will remove any interior arcs so that a multi-polygon input cover generates a single-polygon output cover.

Usage: MAKOUTLIN <input polygon cover (existing)> <output polygon outline name (or '#' if not removing internal lines> <Is the in_cover a BUFFER of another cover? (y/n)> <name to designate the interior of the output cover (created)>

MODEL.AML

This macro takes an input polygon cover and creates the node coordinate data (NCD) and element connection data (ECD) files, as well as three output coverages based on those files. MODEL.AML optimizes the node numbering of the nodes in each element after it creates the NCD and ECD files.

Usage: MODEL <name of point cover on which to base model (existing)> <max # of optimizing iterations> <name of printed output file (created)> <output mesh polygon cover (created)> <output label point cover (created)> <output node point cover (created)>

REALENGTH.AML

This macro takes the total length of a given, non-*SPLINE*d stream and divides that length by the number of nodes in the *SPLINE*d stream. Each node is given a "length" and the sum of the nodal lengths equals the length of the original stream.

Usage: REALENGTH <display type> <original length stream cover (existing)>
<minimum distance between derivative stream cover points> <derivative stream point cover (existing)>

REMODEL.AML

This macro creates the output model coverages based on existing copies of the NCD and ECD files. It does not mandate optimization, as MODEL.AML does.

Usage: REMODEL <max # of optimizing iterations> <name of output printed files (created)> <should the FILENCD and FILEECD files be optimized? (y/n)> <output mesh polygon cover (created)> <output label point cover (created)> <output node point cover (created)>

SNAPPY.AML

This macro uses an iterative process to *SNAP* points in a point cover together. This process ensures that no points will lie within the *SNAP* distance of any other points.

Usage: SNAPPY <display type> <point cover to be SNAPped (existing)> <map scale as if on a 24" plotter (ft)> <SNAPping tolerance>

SPLEEN.AML

This macro SPLINEs up to 10 coverages at a time. Also, it allows the use of up to three different SPLINE distances if there already exist polygon outlines to separate the areas of differing distances.

Usage: SPLEEN <display terminal type> <general spline distance>
<smaller SPLINE distance (or 0 if not using a smaller distance)>
<smallest SPLINE distance (or 0 if not using a smallest distance)>
<cover #1 name (existing)> {{cover #2 name (existing)} ... {cover #10 name (existing)}

TRIANGLE.AML

This macro creates a regularly spaced triangular grid of points. The extent of the grid as well as the point spacing may be entered graphically.

Usage: TRIANGLE <display type> <background cover name (existing)> <length of side (0 to enter graphically)> <output point cover name (created)>

III. Compiling and Linking Fortran Programs on the Prime System

On a PRIME minicomputer, all programs can be compiled using the "F77" command. The format for this command is: "F77 (root name of '.F77' program)." This will generate a "(root name).BIN" output file. This binary file can then be linked using the BIND command. For all programs except SLIP.F77 and BLDMOD.F77 (and its associated subroutines), the BIND command is very simple. The only file which needs to be linked to the binary file is the standard FORTRAN library. This is done using the following format: "BIND -LO (root name) -LI."

For the SLIP.F77 and the BLDMOD.F77 programs, the PRIME system library, and the ARC/INFO system library also need to be linked. For SLIP.F77, the command is: "BIND -LO SLIP -LI VAPPLB -LI ARCLIB -LI." Note that if the ARC/INFO system library is not in the same area as the program then the path to it must be included in its name. For BLDMOD.F77, the format is very similar, but includes the subroutines. It is: "BIND -LO BLDMOD -LO BLDNCD -LO BLDECD -LO OPTIMIZE -LI VAPPLB -LI ARCLIB -LI."

IV. Compiling and Linking Fortran Programs on a Unix System

On Unix systems, like the Data General, the Fortran compiler usually requires that the source code have a ".f" suffix rather than the ".F77" suffix with which the programs are supplied. For this reason, the first step in compiling on the DG is to rename all of the ".F77" files to "(root name).f"

All programs except SLIP.F and BLDMOD.F and its subroutines are compiled and linked in the same step. This is accomplished with the following command: "ghf77 (full name, including '.f' suffix) -o (root name).out." For SLIP.F and BLDMOD.F, "make files" are used to compile and link. These are called "MAKESLIP" and "MAKEBLDMOD." Each will require a slight modification before it is run. Early in the make file, there are two lines that begin with "SOURCELOC" and "ESRILOC." These paths must be changed to reflect the current path in which the source programs and the ARC/INFO libraries reside. Once this is done, the command to compile and link each is "make -f make file name (either MAKESLIP or MAKEBLDMOD)."

MAKESLIP

```
# Makefile for compiling SLIP which is a fortran program for Freud.AML that
# calls subroutines from the ARC/INFO system libraries
# Modified by Kuniansky 4-6-92 from a makefile by Orzol and McGrath, USGS 1991
# for their modules that interface ARC/INFO with the
# MacDonald and Harbaugh ground-water flow model

# Define the libraries
# (modify the variables SOURCELOC and ESRILOC for your machine paths)

SOURCELOC = /arc_users/rlowther/meshtools/test/
ESRILOC = /usr/arc/esri/source/arc50/
PROGRAM = slip


#
# # Define all object files which make up your programs
#  (Put any of your own object codes here for your future programs)

OBJECTS = \
        $(SOURCELOC)slip.o


SOURCES=$(OBJECTS:.o=.f)

# Define the ARC and system libraries

ARCLIBS  = $(ESRILOC)lib/aclib.a $(ESRILOC)lib/arclib.a
SYSLIBS=  -lX11 -leditread -lcurses

APBD=\
        $(ESRILOC)ctxsys/aptxt.o \
        $(ESRILOC)clnsys/aplin.o \
        $(ESRILOC)cmksys/apmrk.o \
        $(ESRILOC)cshsys/apshd.o
ADBD=\
        $(ESRILOC)addsys/adcmin.o \
        $(ESRILOC)addsys/isint.o
```

```
ARCBD=\
        $(ESRILOC)arcio/*bd.o \
        $(ESRILOC)dgintf/dgbd.o \
        $(ESRILOC)dvisys/dvibd.o \
        $(ESRILOC)aexit/aexblk.o \
        $(ESRILOC)bitsys/bitblk.o \
        $(ESRILOC)ttsys/ttbd.o \
        $(ESRILOC)p23sys/p23bd.o \
        $(ESRILOC)csys/cmode.o \
        $(ESRILOC)isp/infxin.o \
        $(ESRILOC)wrksys/wrkbd.o
AMLBD=\
        $(ESRILOC)aml/amlbd.o \
        $(ESRILOC)aml/dirbd.o \
        $(ESRILOC)aml/funbd.o \
        $(ESRILOC)aml/kfilbd.o \
        $(ESRILOC)aml/pgtbd.o \
        $(ESRILOC)aml/srcbd.o \
        $(ESRILOC)aml/uflbd.o \
        $(ESRILOC)smgsys/curbd.o \
        $(ESRILOC)smgsys/mx1bd.o \
        $(ESRILOC)smgsys/sm1bd.o \
        $(ESRILOC)smgsys/sm1sysx.o \
        $(ESRILOC)smgsys/smgbd.o

# Define the Fortran compile flags

FFLAGS= -g

# Define Task Function Program bldmod

all: slip

# Define what slip is

slip: $(OBJECTS) \
        $(APBD) \
        $(ADBD) \
        $(ARCBD) \
        $(AMLBD) \
        $(ARCLIBS)
        -f77 $(FFLAGS) -o slip.out \
        $(OBJECTS) \
        $(APBD) \
        $(ADBD) \
        $(ARCBD) \
        $(AMLBD) \
        $(ARCLIBS) \
        $(SYSLIBS)

# slip object codes

slip.o: slip.f
        ghf77 $(FFLAGS) -c slip.f
```

```
# Define Module fortran

all install: $(OBJECTS)

object: $(OBJECTS)

# end
```

MAKEBLDMOD

```
# Makefile for compiling BLDMOD which is a fortran program that
# calls subroutines from the ARC/INFO system libraries
# Modified by Kuniansky 3-25-92 from a makefile by Orzol and McGrath, USGS
# 1991, for their modules that interface ARC/INFO with the
# MacDonald and Harbaugh ground-water flow model

# Define the libraries
# (modify the variables SOURCELOC and ESRILOC for your machine paths)

SOURCELOC = /arc_users/rlowther/meshtools/test/
ESRILOC = /usr/arc/esri/source/arc50/
PROGRAM = bldmod
#
# # Define all object files which make up your programs
#  (Put any of your own object codes here for your future programs)

OBJECTS = \
        $(SOURCELOC)bldmod.o\
        $(SOURCELOC)bldncd.o\
        $(SOURCELOC)bldecd.o\
        $(SOURCELOC)optimize.o

SOURCES=$(OBJECTS:.o=.f)

# Define the ARC and system libraries

ARCLIBS  = $(ESRILOC)lib/aclib.a $(ESRILOC)lib/arclib.a
SYSLIBS=  -lX11 -leditread -lcurses

APBD=\
        $(ESRILOC)ctxsys/aptxt.o \
        $(ESRILOC)clnsys/aplin.o \
        $(ESRILOC)cmksys/apmrk.o \
        $(ESRILOC)cshsys/apshd.o
ADBD=\
        $(ESRILOC)addsys/adcmin.o \
        $(ESRILOC)addsys/isint.o
ARCBD=\
        $(ESRILOC)arcio/*bd.o \
        $(ESRILOC)dgintf/dgbd.o \
        $(ESRILOC)dvisys/dvibd.o \
```

```
        $(ESRILOC)aexit/aexblk.o \
        $(ESRILOC)bitsys/bitblk.o \
        $(ESRILOC)ttsys/ttbd.o \
        $(ESRILOC)p23sys/p23bd.o \
        $(ESRILOC)csys/cmode.o \
        $(ESRILOC)isp/infxin.o \
        $(ESRILOC)wrksys/wrkbd.o
AMLBD=\
        $(ESRILOC)aml/amlbd.o \
        $(ESRILOC)aml/dirbd.o \
        $(ESRILOC)aml/funbd.o \
        $(ESRILOC)aml/kfilbd.o \
        $(ESRILOC)aml/pgtbd.o \
        $(ESRILOC)aml/srcbd.o \
        $(ESRILOC)aml/uflbd.o \
        $(ESRILOC)smgsys/curbd.o \
        $(ESRILOC)smgsys/mx1bd.o \
        $(ESRILOC)smgsys/sm1bd.o \
        $(ESRILOC)smgsys/sm1sysx.o \
        $(ESRILOC)smgsys/smgbd.o

# Define the Fortran compile flags

FFLAGS= -g

# Define Task Function Program bldmod

all: bldmod

# Define what bldmod is

bldmod: $(OBJECTS) \
        $(APBD) \
        $(ADBD) \
        $(ARCBD) \
        $(AMLBD) \
        $(ARCLIBS)
        -f77 $(FFLAGS) -o bldmod.out \
        $(OBJECTS) \
        $(APBD) \
        $(ADBD) \
        $(ARCBD) \
        $(AMLBD) \
        $(ARCLIBS) \
        $(SYSLIBS)

# bldmod object codes
bldmod.o: bldmod.f
        ghf77 $(FFLAGS) -c bldmod.f
# Define Module fortran
all install: $(OBJECTS)
object: $(OBJECTS)
#  end
```