ERDC/EL SR-04-3

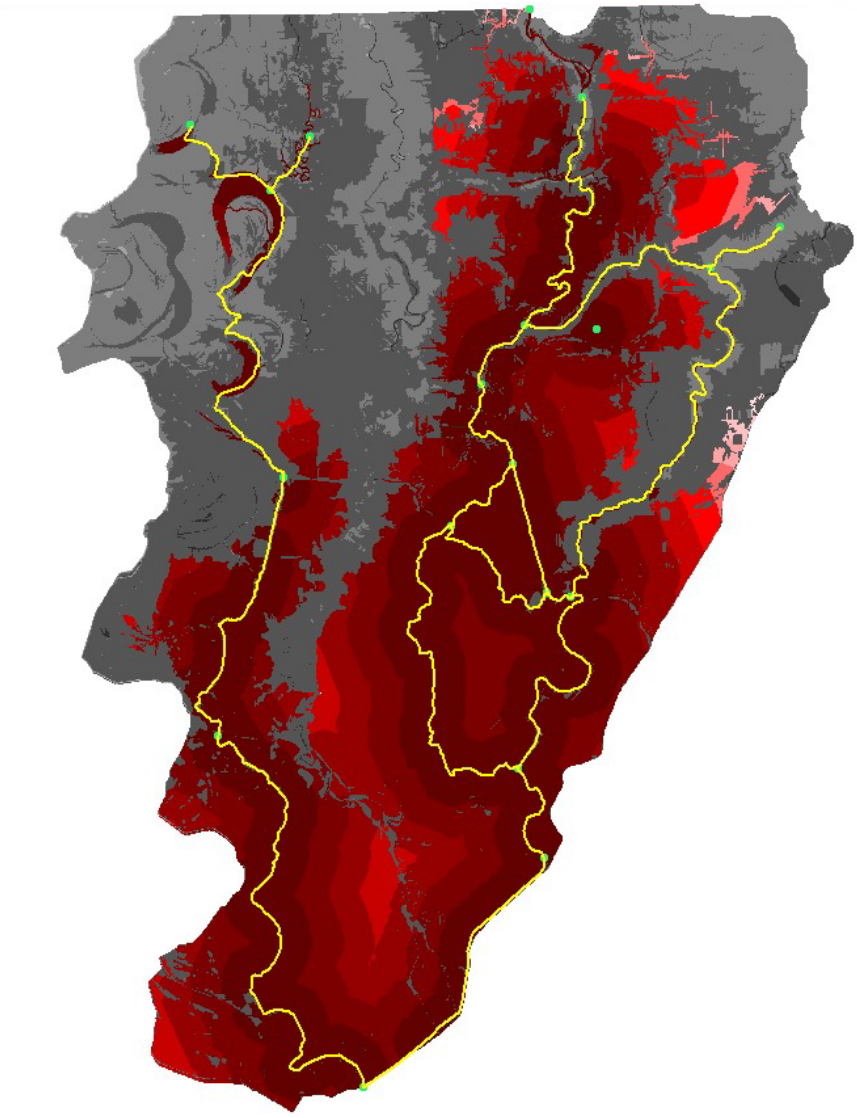
Environmental Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

Flood Event Assessment Tool (FEAT) User's Manual and Technical Documentation

Jerrell R. Ballard, Jr., and Margaret Rose Kress

July 2004



ERDC/EL SR-04-3
July 2004

Flood Event Assessment Tool (FEAT) User's Manual and Technical Documentation

Jerrell R. Ballard, Jr., and Margaret Rose Kress

Environmental Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final report

Prepared for U.S. Army Engineer District, Vicksburg
Vicksburg, MS 39183

ABSTRACT:
The Flood Event Assessment Tool (FEAT) is a prototype geospatial modeling tool that uses river and stream gage data, landscape digital elevation models (DEMs), main and secondary channel centerlines or cross sections to generate a geospatial-based flood surface. The primary objective in developing FEAT was to improve the accuracy and decrease the time and effort required to determine dynamic flood surfaces and provide information for decision-making. This guide provides an overview of FEAT, information about the algorithms, software and hardware requirements, installation procedures and operation, user input requirements, and examples of model output products.

Contents

List of Figures

Preface

The work described herein was conducted by personnel of the U.S. Army Engineer Research and Development Center (ERDC), Waterways Experiment Station (WES), Vicksburg, MS, from October 1997 to April 1999. The study was funded by the U.S. Army Engineer District, Vicksburg. Technical Monitor was Mr. Ron Goldman, Hydrologist, Hydraulics and Hydrology Division, Vicksburg District.

The report was prepared by Mr. Jerrell R. Ballard, Jr., and Dr. Margaret Rose Kress of the Environmental Characterization Branch (ECB), Natural Resources Division, Environmental Laboratory (EL), WES.

The study was conducted under the general supervision of Dr. John W. Keeley, Acting Director, EL, and Dr. David Tazik, Chief, NRD, and under direct supervision of Mr. Harold Wade West, Chief, ECB.

At the time of publication of the report, Commander and Executive Director of ERDC was COL James R. Rowan, EN. Director was Dr. James R. Houston.

1 Introduction

Existing methods for estimating the extent of flooding in the Lower Mississippi River Valley primarily consist of interpretation of satellite data. These methods are labor-intensive and often are restricted due to extensive cloud cover during a flood event. Another limitation includes the inability to "project" flood extent for future events or Corps project developments.

The Flood Event Assessment Tool (FEAT) is a prototype geospatial modeling tool that uses river and stream gage data, landscape digital elevation models (DEMs), and main and secondary channel centerlines or cross-sections to generate a geospatial-based flood surface. The primary objective for the development of FEAT was to improve the accuracy and decrease the time and effort to determine dynamic flood surfaces and provide information for decision making. The conceptual algorithm for the geospatial flooding was initially developed for the Flood Impact Support Tool[1] (FIST) system and was additionally documented by Ballard and Kress.[2]

This guide provides an overview of FEAT, information about the algorithms, software and hardware requirements; installation procedures and operation; user input requirements, and examples of model output products.

This modeling procedure was developed by the U.S. Army Engineer Research and Development Center, Waterways Experiment Station, Vicksburg, MS, for the Hydrography and Hydrology Division, Vicksburg District, U.S. Army Corps of Engineers.

---

[1] Ballard, J. R., Jr., and Kress, M. R. (1999). *Flood Impact Support (FIST) User's Manual and Technical Documentation.* Instruction Report EL-99-1, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

[2] Ballard, J. R., Jr., and Kress, M. R. (1998). "Cost-Distance Flood Surface Generation for Economic Impact Assessment." Proceedings for the 18th Annual ESRI User Conference, San Diego, California, July 27-31, 1998.

2 Components of FEAT

FEAT combines spatial analysis techniques and high-resolution geospatial data to determine land area inundated under specified flood conditions. FEAT has four components:

- A fully functional commercial off-the-shelf (COTS) geographic information system (GIS) (ArcView® 3.1.1).
- A high-resolution geospatial database for the area.
- The flood impact model.
- A custom-designed user interface.

The GIS serves as the database manager, provides access to a complete range of spatial analysis capabilities, and establishes the graphic display environment. ArcView® from Environmental Systems Research Institute (ESRI) is the GIS used for FEAT.

The high-resolution databases depict quantitative information on the physical terrain surface, water elevations at stream recording gages, and land-cover (vegetation types) conditions in selected study areas. Required database components are documented in Chapter 4, "Database Requirements."

The flood impact model has been designed and developed to run in conjunction with and utilize the capabilities of ArcView®. It includes programs written in Avenue, the programming language within ArcView®. These programs are listed in Appendix A.

The user interface is written using the Avenue language. The interface helps the user select appropriate data types for the model and establishes user set parameters.

3 Hardware, Software, and Training Requirements

FEAT requires specialized computer hardware and software for proper system operation. Simulating flood events is computationally intensive and requires a high performance personal computer platform for optimum performance. To achieve this level of performance, the minimum hardware requirements are a computer running Windows 98 with at least a 400-Mhz Pentium II CPU, CD-ROM drive, 6 gigabytes of hard disk space, and a 16-bit, 17-in. color graphics monitor.

The GIS used in FEAT is ESRI ArcView®, with the Spatial Analyst Extension. ESRI Avenue code was utilized for the algorithms in FEAT. Because of this, the minimum software requirements are ESRI ArcView® Version 3.1.1, and ESRI Spatial Analyst Extension Version 1.1.

It is recommended that the end user of FEAT be trained and well acquainted with the operation of ArcView® and have a technical understanding of geographic information systems concepts (GIS) and hydrology. ESRI provides several training courses during each year at locations nationwide.

4 Database Requirements

Specific data types are required by FEAT. Each type of data are briefly described below.

FEAT is designed to use data contained in the GIS database and other usersupplied parameters. All geospatial data used in FEAT must be projected to the Universal Transverse Mercator (UTM) projection, North American Datum (NAD83). The required GIS data layers are as follows:

Table 1
Required GIS Data for FEAT

Name	Description
Elevation	Basin ground surface elevation model (Required)
Water Gages	Location of gages, and attributes of water surface elevations (Required)
Cross Sections	Channel cross sections and attributes of water surface elevations (Optional)
River Center Lines	River/Channel center lines, attributed with a from and to gage (Required)
Secondary Channels	Secondary channel center lines (Optional)
Landcover	Landcover (Optional)

Elevation

The “elevation” file is a grid file containing the digital elevation model (DEM) of the basin. This DEM must be able to accurately represent the topography of the basin. This implies that all channels, streams, and levees must be represented with an adequate grid cell resolution.

The DEM can be processed from digitized from 1:24,000 U.S. Geological Survey (USGS) topographic maps and supplemental field-surveyed elevations. The DEM can also be obtained from the USGS as DEM level II products. As a general rule, for any project, the cell resolution must be at least four times smaller than the width of the smallest river channel in the data set. If the grid cell resolution is too large, small channel geometry may not be represented sufficiently in the DEM and thereby result in inaccurate modeling of the flooded area, especially backwater flooding. The DEM vertical units must be represented as meters above mean sea level.

Water Gages

The “water gages” file is a point feature file containing the geographic location and the unique identification number of each river gage. Also, at each river gage there must be at least one water surface elevation associated with each river gage location. Table 2 provides an example of possible water gages point file attributes where the water surface elevations are provided in feet above mean sea level. These water surface elevations can be obtained from either direct readings or model outputs.

Table 2
Water Gage Required Items (Example)

Gage ID	Name	Two-Year (ft)	Ten_Year (ft)
1	Anguilla	98.0	100.2
2	Holly Bluff	95.3	98.2

Cross Sections

The “cross sections” file is a line feature file containing the channel cross sections with attached water surface elevation attributes. These cross sections can be obtained typically from Hydrologic Engineering Center (HEC) model outputs. This file can be used optionally with FEAT.

River Center Lines

The “river center lines” file is a line feature file containing river and channel center lines. Each line is attributed with an upstream gage identification number and a downstream gage identification number. These attributes allow FEAT to identify the proper up- and downstream water elevations at run-time. Table 3 provides an example of the attributes for one center line. In this example the river center line runs from gage 1 (Anguilla) to gage 2 (Holly Bluff) according to Table 2.

Table 3
River Center Line Required Items (Example)

Center-line_ID	From	To
1004	1	2

Secondary Channels

The “secondary channels” file is a line feature file that represents important secondary channels in the flood event. These are unattributed line segments that are directly attached to a river center-line vertex. During execution, FEAT determines the water surface elevation at the center-line vertex and associates it with the entire secondary channel. A schematic example of this is shown in

Figure 1. The secondary channel file is useful in obtaining proper backwater flooding in small areas with inadequate delineated channels.

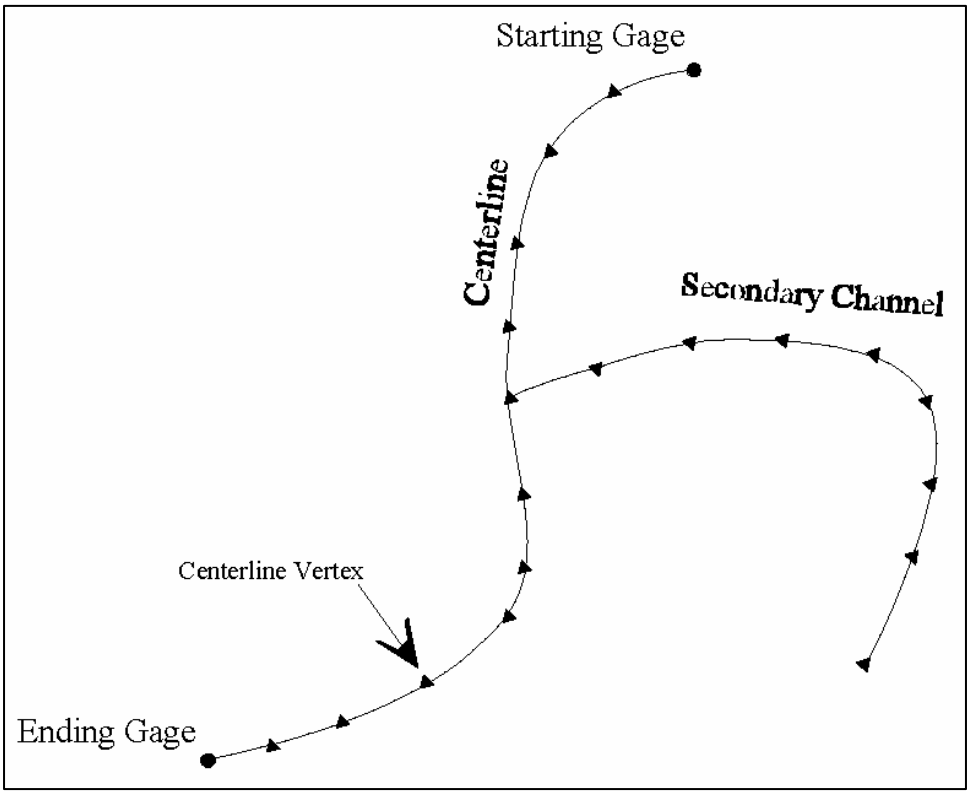


Figure 1. Main channel center line and secondary channel example

Landcover

The "landcover" file is a grid file describing cleared, wooded, and urban areas in the basin. This grid file is used after a flood surface is generated to determine which types of landcover are impacted by the flood event.

5 FEAT Operation

The FEAT extension in ArcView® contains a simple graphic interface that requires inputs from the user. The following text describes the operation of the extension, interface, user inputs, and examples of its operation.

Activating the FEAT extension

Load the extension using the extensions dialog under the **File** pull-down menu. Select and load the FEAT extension. The user should see a dialog box that looks similar to Figure 2. Find where the extension is listed and place a check in the box. Then press the **OK** button on the Extensions dialog to load the extension. The extension will load itself and the Spatial Analyst extension into ArcView and is now ready for use.

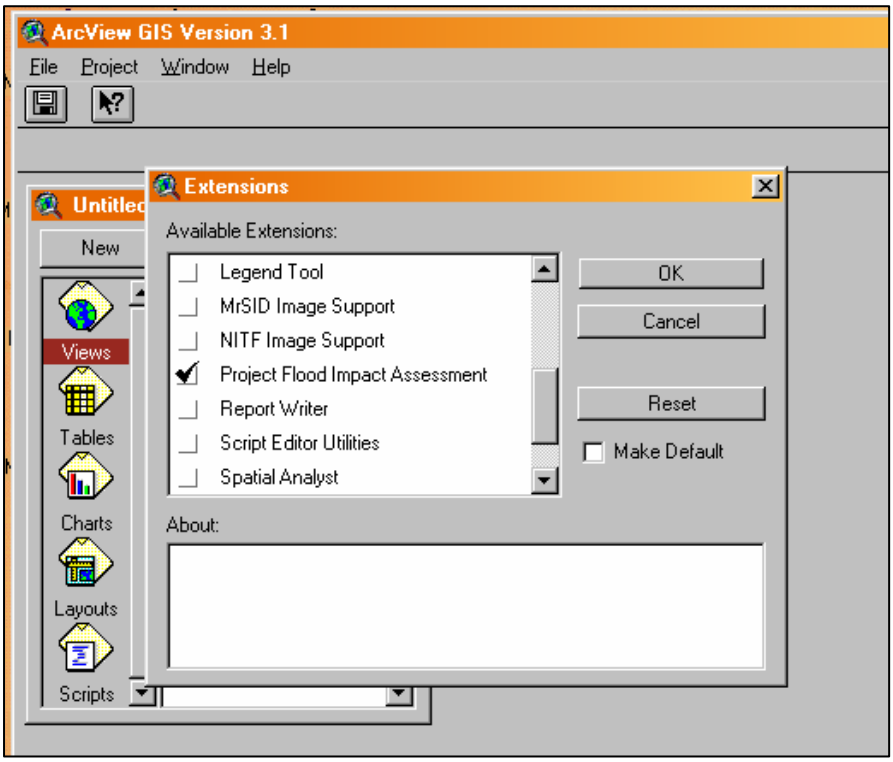


Figure 2. Enabling the FEAT extension within ArcView

Starting the Flood Event Simulation

Once the required data layers are installed into a view, (the ArcView® users manual explains how to do this), the display should look similar to Figure 3. The following steps explain and user inputs and actions.

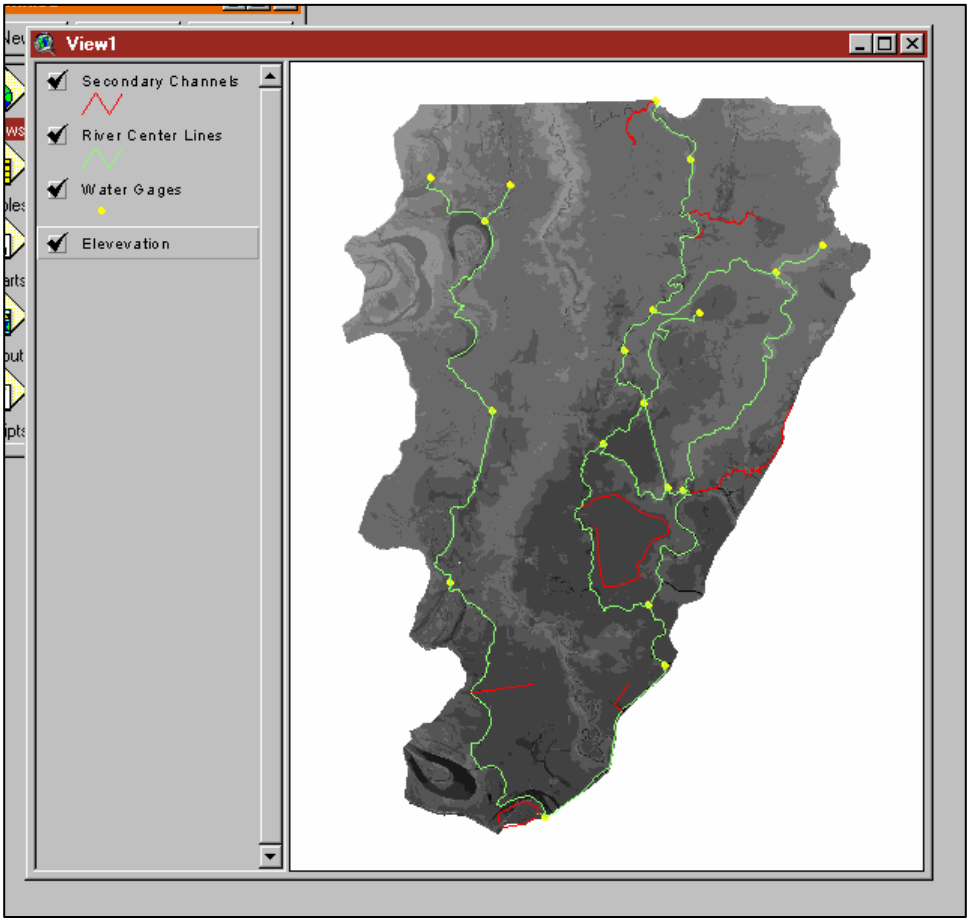


Figure 3. Initial setup of required data layers

STEP 1: The pull down menu is titled 'Flood Impact Assessments' and select the one 'Determine Flood Extent' (Figure 4). The extension will prompt the user to indicate the elevation grid theme to be used for calculations (Figure 5). If no grid themes are available, the simulation will terminate. There will be only one elevation grid theme used during the simulation.

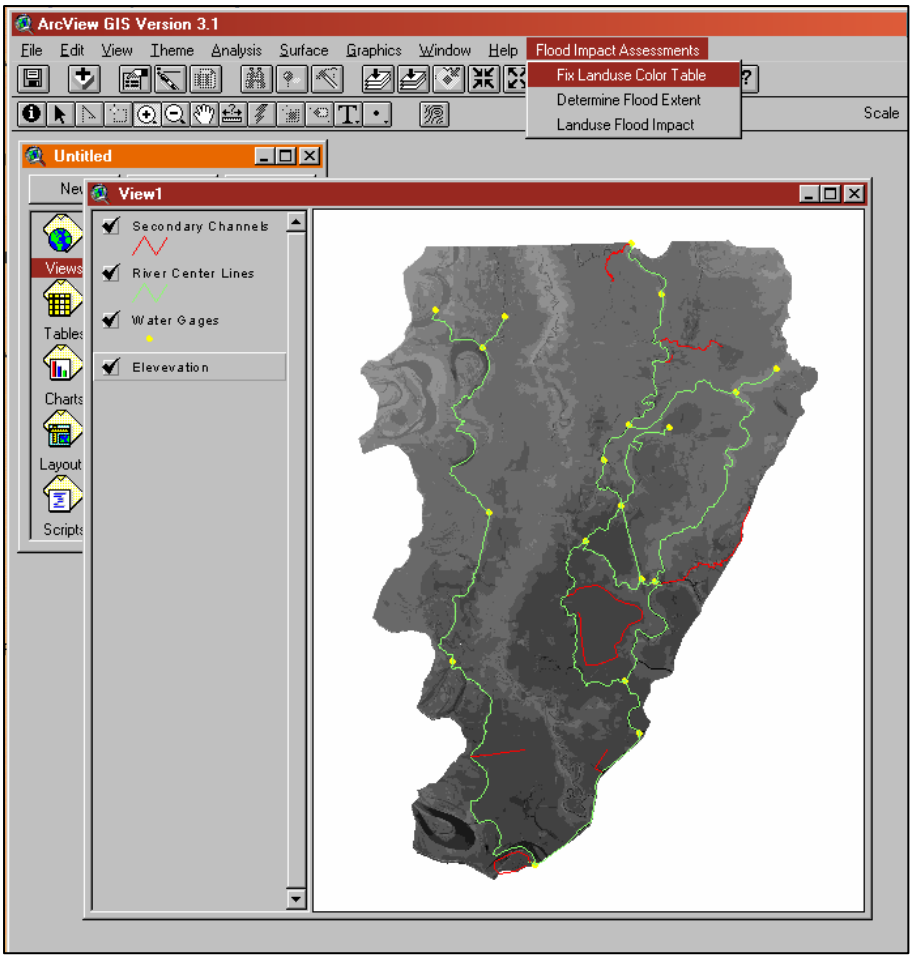


Figure 4. Selecting the option in the pull-down menu

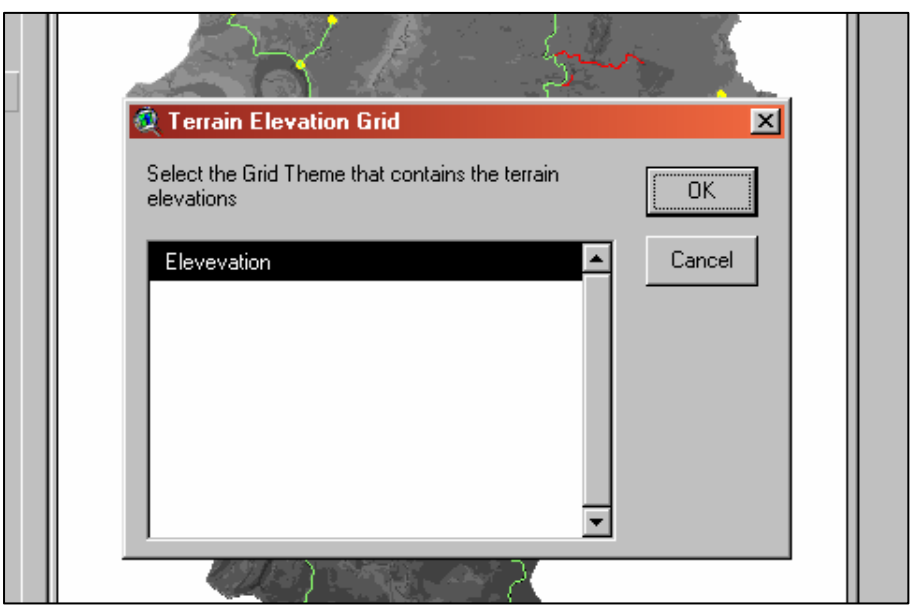


Figure 5. Elevation grid theme prompt

STEP 2: After the user has selected the appropriate elevation grid theme, the next prompt will be the point theme that will be used for the river gages (Figure 6).

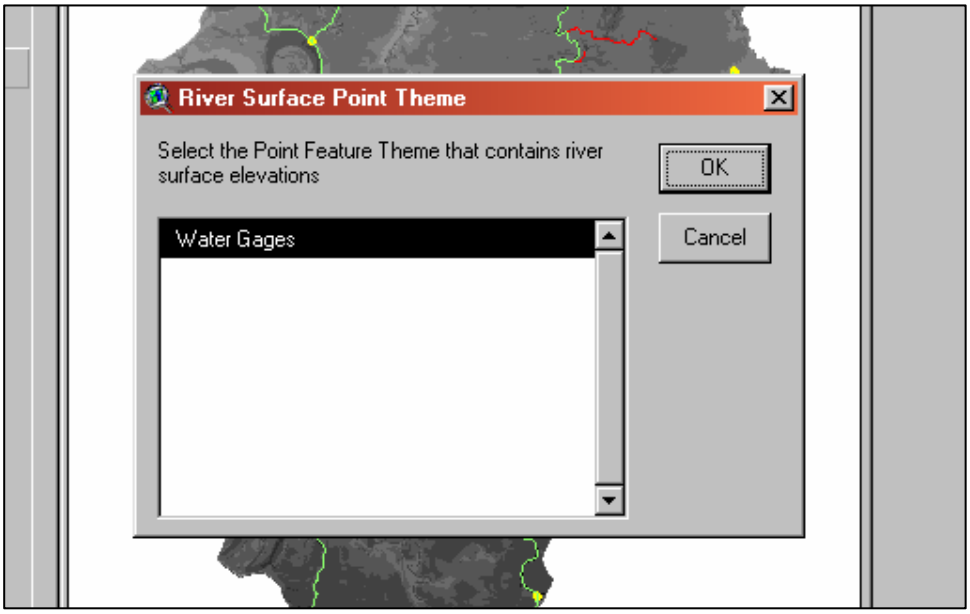


Figure 6. Prompt for the Water Gage theme

STEP 3: Once selected, an examination of the attributes (or fields) attached to the points will be made to indicate the water surface elevations.

STEP 4: The last prompt associated with the gages, is to check the units of the water surface elevations (Figure 7). If the units are not in feet, then the units are assumed to be in meters.

Figure 7. Vertical units prompt for the water surface gages

STEP 5: The next step is to select a centerlines theme. All themes are examined in the view, and only the line themes are presented. It is the user's

responsibility to select the correct line theme. The prompt for this is shown in Figure 8. Select the correct line theme and then press the **OK** button.

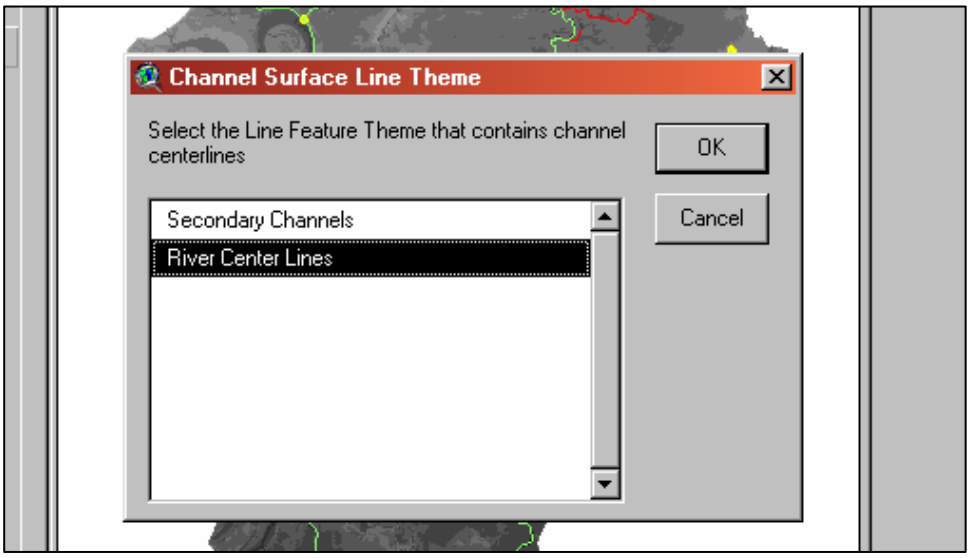


Figure 8. River Center Lines prompt

STEP 6: This step examines the attributes for the selected point theme and prompts for item names where the beginning and ending water gage identification numbers are stored (Figures 9 and 10).

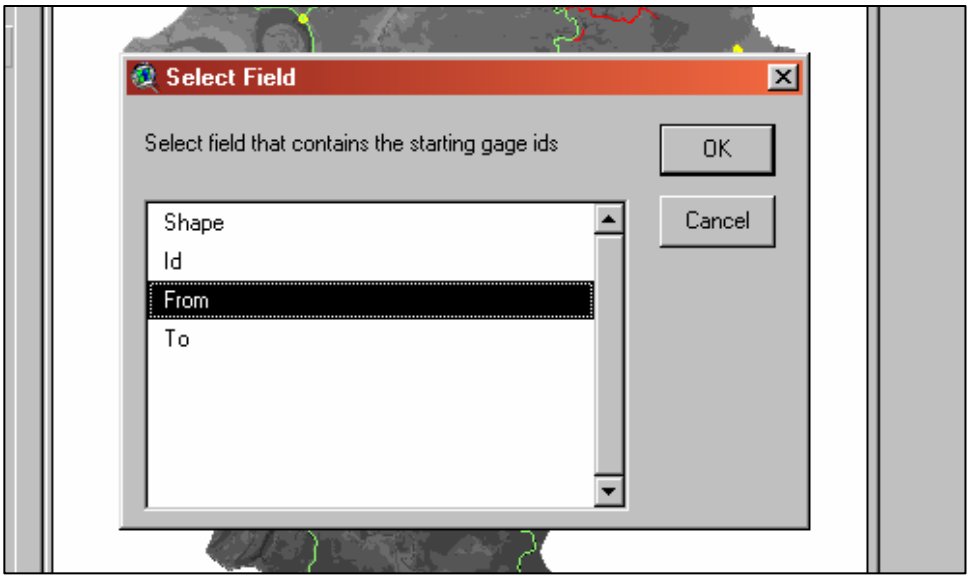


Figure 9. Center Line Water Gage Upstream selection

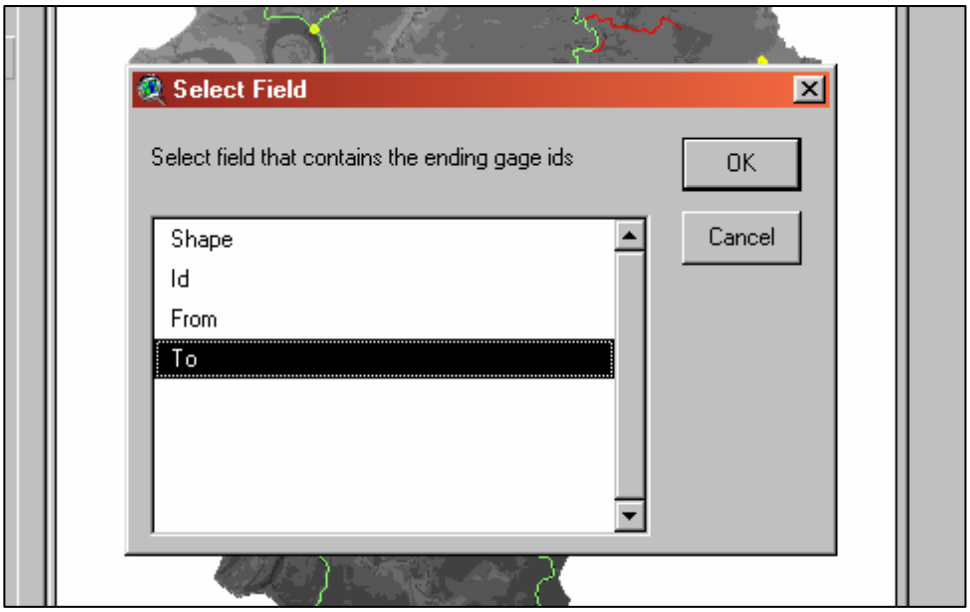


Figure 10. Center Line Water Gage Downstream selection

STEP 7: To enhance the water surface interpolation, additional vertices may be added to the centerlines by increasing the density of all the lines. A prompt asks for the density, but if a density increase is not required then the density should be set to a number greater than the average density of the centerline's vertices (Figure 11).

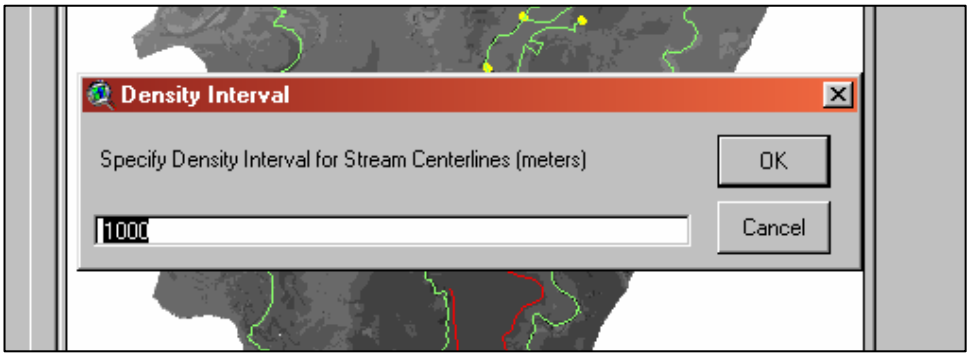


Figure 11. Vertex density selection for the stream center lines

STEP 8: If a center-line theme is selected for use in the water surface interpolation, the next prompt determines if any secondary channels are to be used (Figure 12). There are no attributes associated with this line theme, and the simulation will determine which end point of the line connects to a center line and will assigns a water surface elevation to the entire secondary channel.

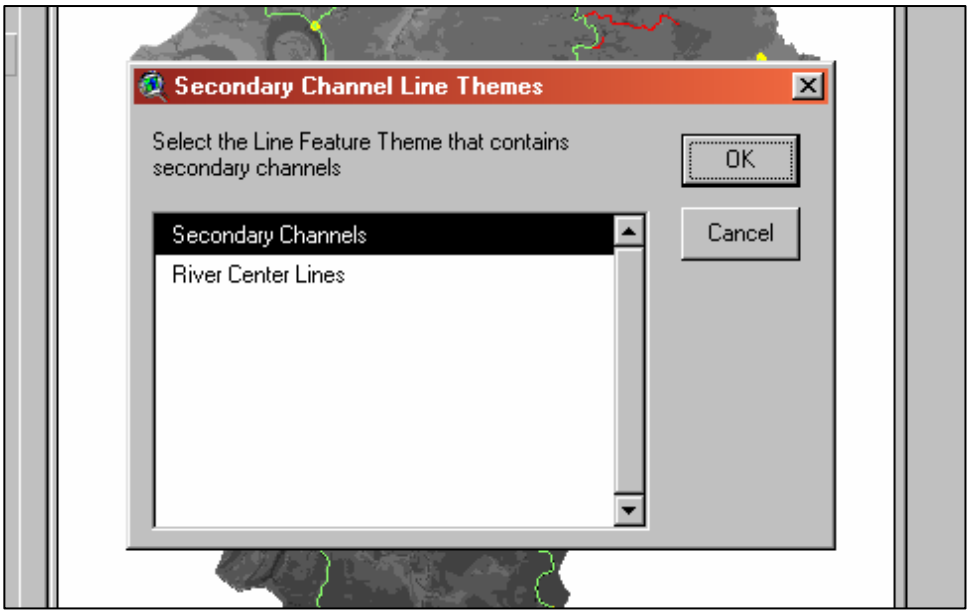


Figure 12. Secondary channel selection

STEP 9: The next prompt is for river cross sections. This is similar to the river center lines selection. Once the line theme has been selected, an examination of the attributes is made and prompts the user again for the attribute (item) that contains the water surface elevations.

STEP 10: In step 10, the user is prompted for the number of steps to use in the water flow calculations. The default is 25 steps (Figure 13). This may be incremented to slightly increase the accuracy of the water flow calculations, but dramatically increases the run time of the simulation. An explanation of how this parameter works and how it affects accuracy of the water flow calculations is provided in Chapter 7.

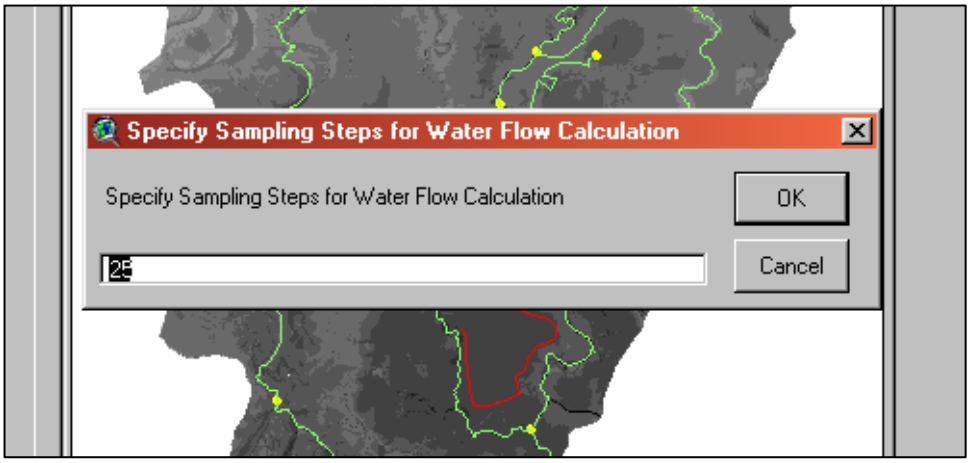


Figure 13. Sampling steps for the water flow calculations

After this last prompt, modeling of the flood event is initiated.

Output Files

Processing time is dependent on elevation grid cell size, water flow step sampling, and flood event complexity. A resulting flooded area map is displayed (Figure 14). The flood event map is named 'Flood Surface: xxxx', where the 'xxxx' is the name of the attribute (item) from the gage point theme that the water surface elevations were utilized.

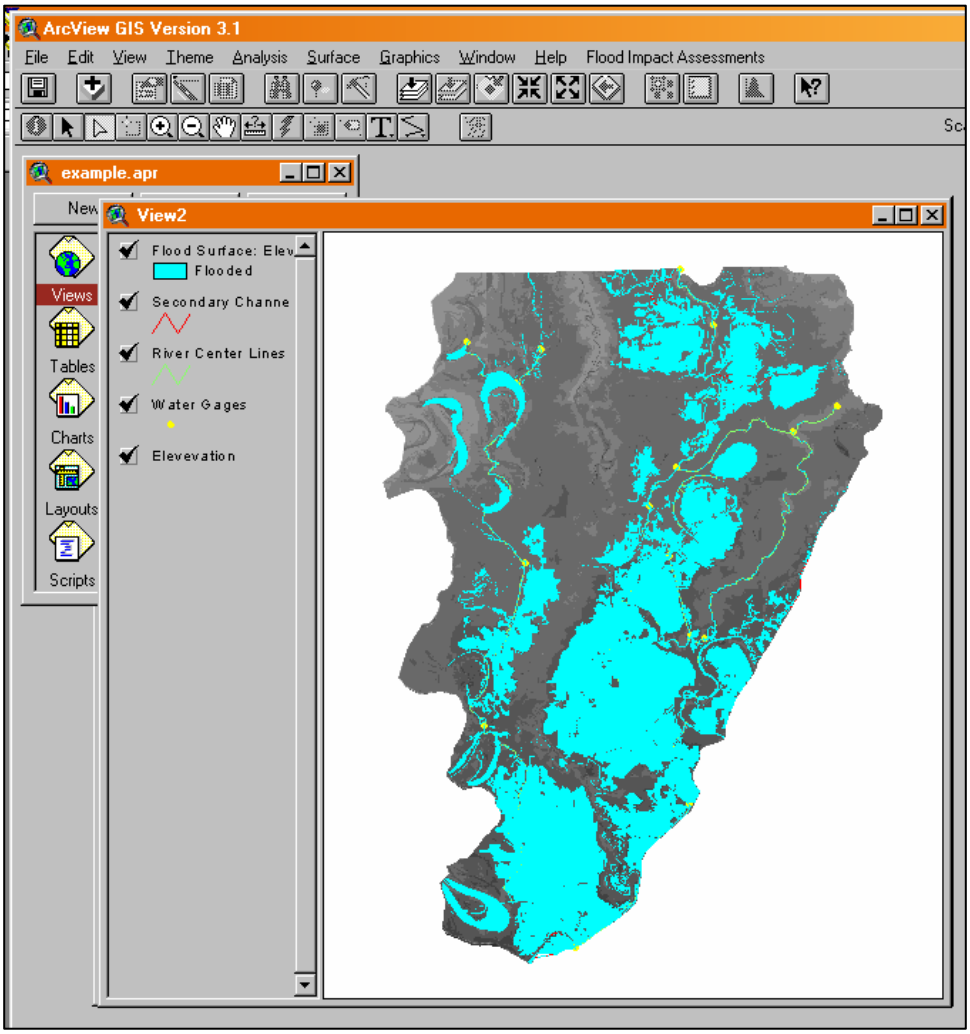


Figure 14. Final results (flooded area map) of flood event simulation

Calculating Flooded Area by Landuse Category

The flooded surface may now be compared to a landcover grid theme to obtain number of acres inundated for each landcover category. Under the pull-down menu '**Flood Impact Assessments'** there is a selection named, '**Landuse Flood Impact'** (Figure 4). This feature will prompt for a landcover grid and a flood surface grid and generate a landcover grid that indicates which areas are inundated and determines the combined acres flooded.

Operational Guidance

During testing and developing of data sets for testing the simulation model, several operational issues were noted. The following is a general list of operational guidance hints:

a. All river gage locations (x,y coordinates) must be located in the center of the river channels.

b. River centerlines must be located in the center of the river channels.

c. All gage readings must have a valid water elevation (no zeros or missing values) and set to an elevation above mean sea level.

d. Increasing sampling size/rate of the water flow calculations (25-50) will provide better results. Increasing this rate will roughly double simulation times.

e. Simulation times will increase with the flooded area extent.

f. Increasing the density of vertices of river center lines will improve results. Increasing the density smaller than twice the elevation cell size will not improve results.

g. Adding secondary channels in small backwater flood areas will improve results and minimize external water channel influences.

h. In ArcView® set the project properties working directory to a directory other than /windows/temp. All grids and final flood extent surfaces will be saved to the working directory specified.

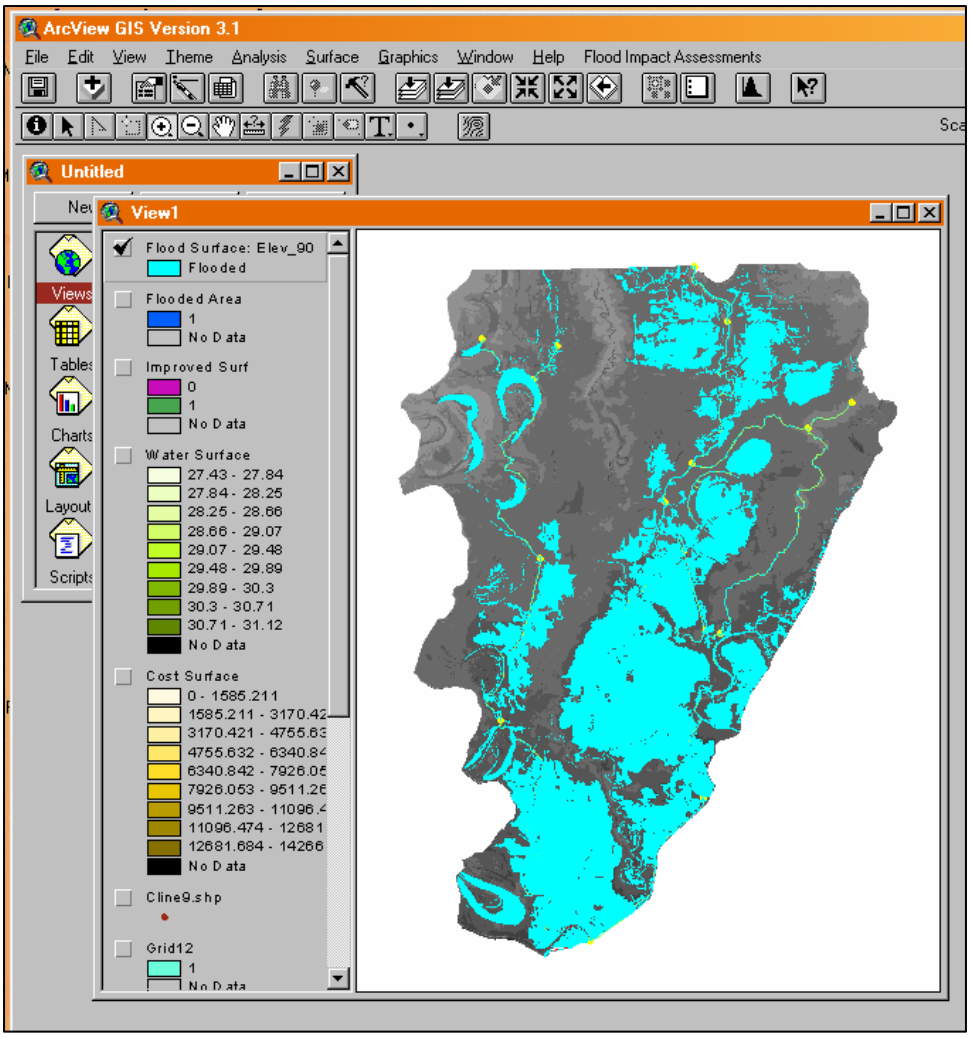


Figure 15. Final results (flooded area map) of flood event simulation using diagnostic mode

6 Diagnostic Mode

Two versions of the FEAT extension are available in ArcView®. Both of these extensions provide the same end results, but one extension has been modified to provide a diagnostic mode and saves all intermediate geospatial data generated during the flood event simulation. These intermediate data are useful in diagnosing and verifying simulation input data and understanding how the iterative simulation determines the flood extent within the basin or region.

In addition to the final flooded surface, in the diagnostic version of FEAT, there are also seven other intermediate diagnostic grid and point themes that can be used for understanding how the simulation is determining which land areas are flooded and how water is routed over the land.

The first data theme produced in the diagnostic mode is a grid theme named 'InitSurf' (Figure 16). This grid theme is a grid of water surface elevations generated by using all the gage, centerline, and secondary channel water surface elevations.

The second data theme produced in the diagnostic mode is a temporary grid theme with the name 'Gridxx,' where 'xx' is a number generated during the simulation (Figure 17). This grid is the result of the first flood event approximation where the simulated water surface elevation is compared to the digital elevation surface. The resulting grid only shows the extent of the flooded area. Only the grid cells that could be flooded under these conditions are indicated. Gridxx also does not consider the effect of manmade or natural levees nor the actual flow of water across the complex landscape.

Data are also provided for the results of the cost distance grid using the previous intermediate flood extent (Figure 18). This grid theme shows the shortest path distance from each cell back to the closest water elevation source point.

In Figure 19, a second water surface grid theme is generated in the diagnostic mode and is illustrated. This second surface is the result of all the gage, centerline channel, secondary channel, and flow path water surface elevations. This water surface grid is then compared to the DEM, which results in the theme 'Improved Surf' as shown in Figure 20.

Using the second flood event approximation, 'Improved Surf', water connectivity is evaluated and a final flood surface is generated (Figure 21).

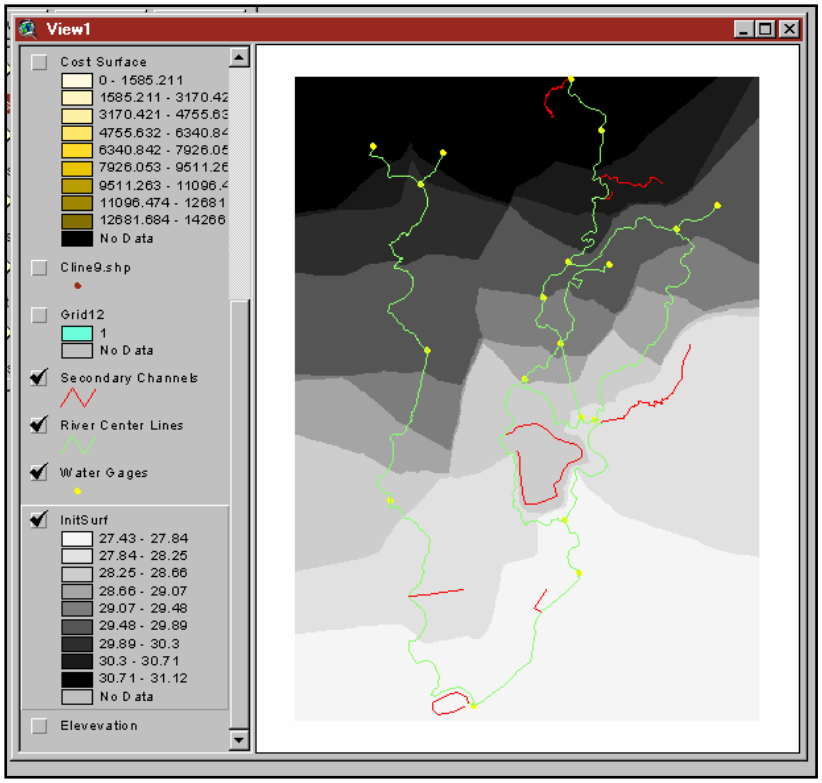


Figure 16. Initial water surface interpolation

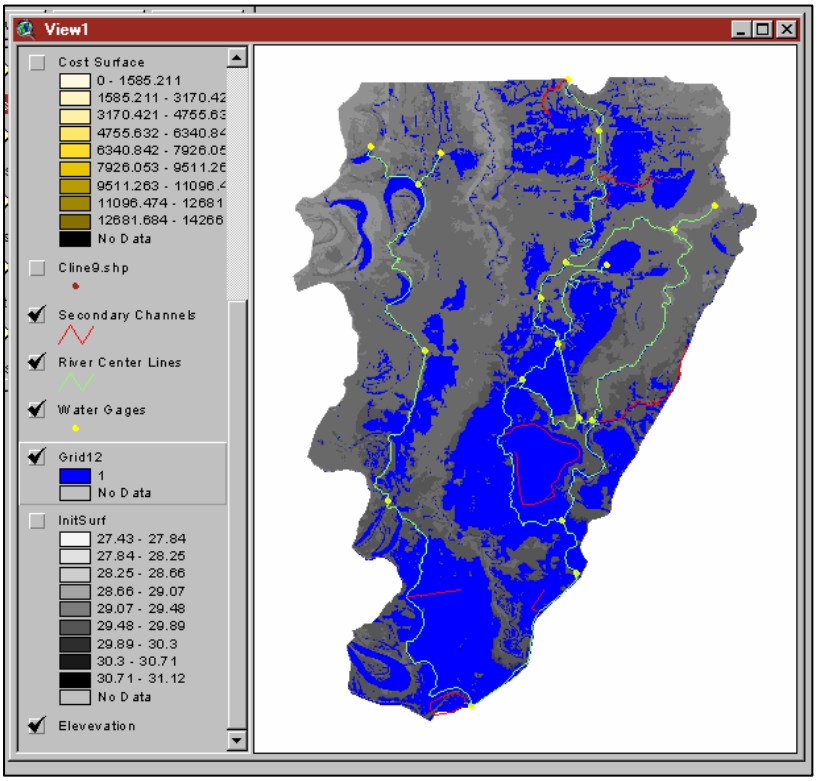


Figure 17. First flood extent approximation

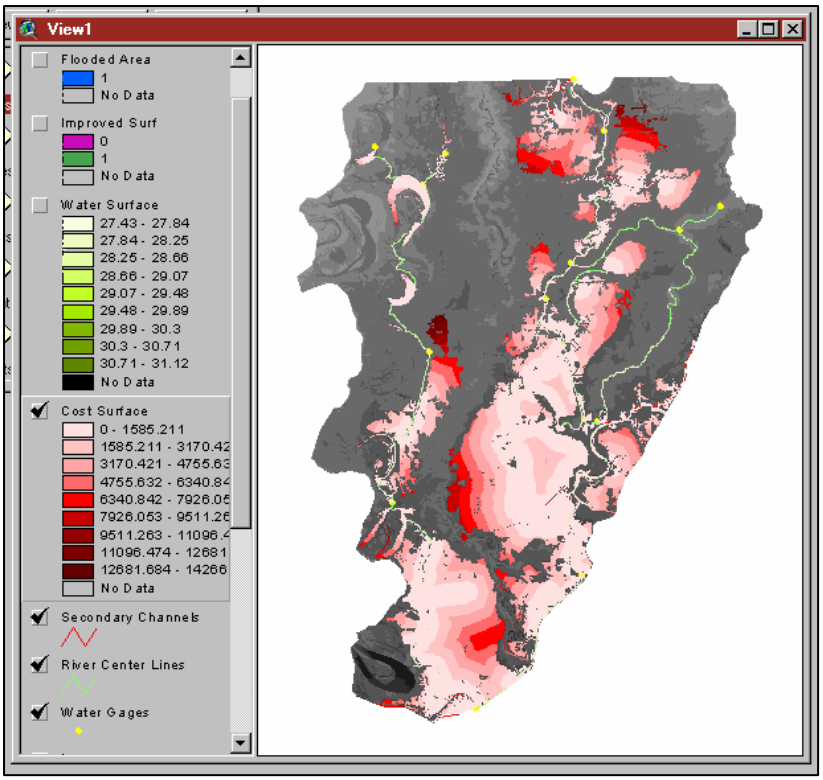


Figure 18. Cost distance grid for determining water flow paths

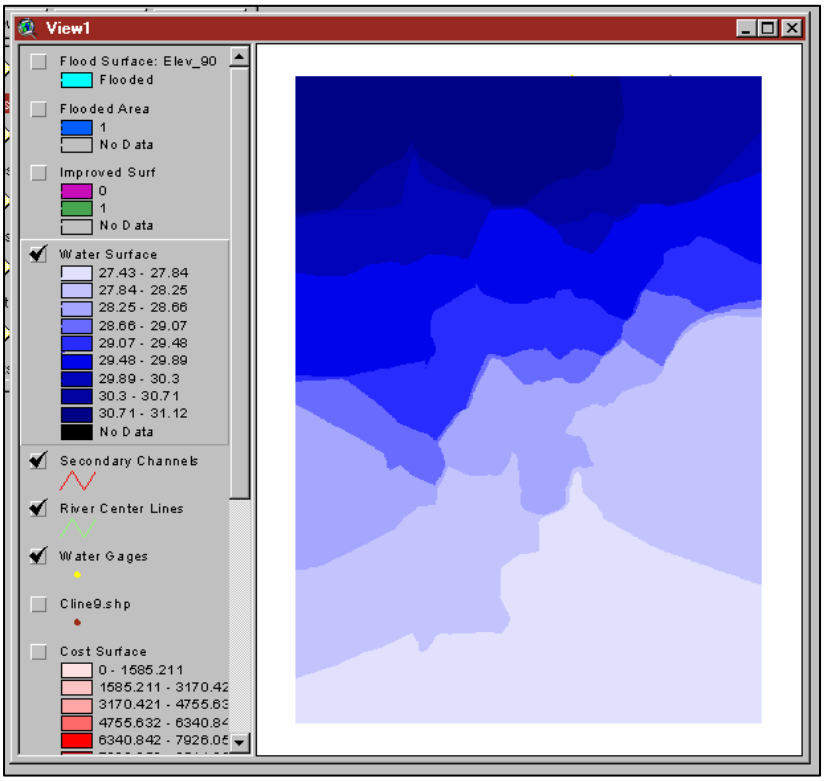


Figure 19. Second water surface interpolation based on water flow paths

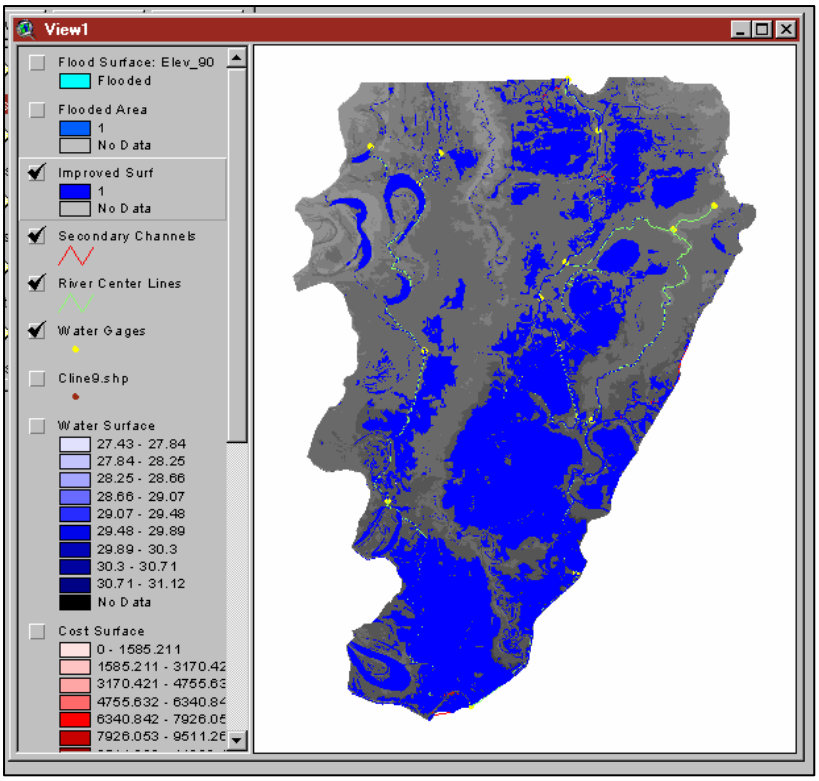


Figure 20. Second flood extent approximation designated as “Improved Surf”

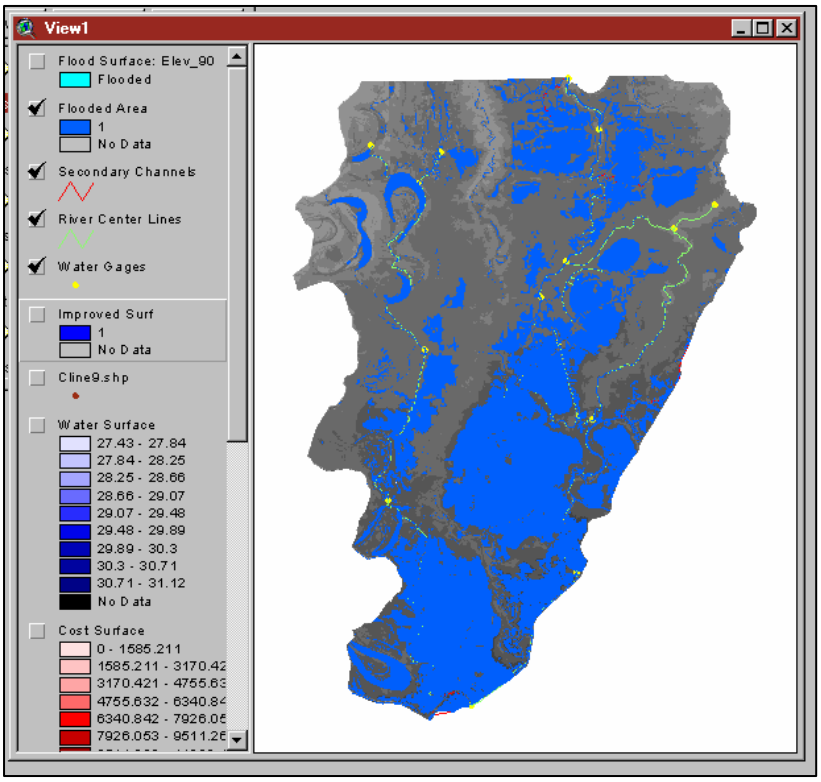


Figure 21. Second flood extent surface

7 FEAT Algorithms

The following is a step-by-step description of the algorithms and processes used to generate a flood surface. Much of the terminology used in this chapter is compatible with the ArcView Avenue programming language.

The first part of the simulation collects and evaluates all the source data from the user. Typically these are river gage water elevations, river center lines, secondary channels, and river channel cross sections. Water elevations from the gage theme are associated with the river center-line polyline endpoints. Once the endpoints have elevation values, each of the vertices of the polyline is assigned an elevation based on the beginning and endpoint elevation slope and distance from the beginning of the polyline. The distance calculated is the distance along the polyline, not a straight line distance. The vertices attributed to the elevation are combined with the gage water elevations and produce a temporary point theme. This process is repeated similarly for the secondary channels, except that the elevation for all the vertices is assigned by finding the closest center-line vertex to one of the secondary channel endpoints. The elevation from this vertex is assigned to all the vertices of the secondary channel and the vertices are appended to the temporary point theme.

Once the point theme has been generated with all the necessary water surface elevations, a water stage surface elevation grid is generated. This water surface grid is generated at a large cell resolution (100 m) using the inverse distance weighted average algorithm using a minimum of 12 sample points. A large resolution was selected at this time to speed up the interpolation process and because this approximated surface only provides a rough estimate of the extent of the flood event.

The approximated water surface is now compared to the ground elevation model to evaluate a first approximated flood area. If the elevation of the water surface is greater than or equal to the ground surface, then the grid cell is flooded; otherwise, the grid is not flooded. The resulting flood area does not contain water surface elevations. It only indicates all grids that could be flooded under these stage conditions. It does not consider effects of man-made or natural levees or the actual flow of water across the flood plain.

The next step determines the effects of topography on overland flow. The initial flood surface is modified to eliminate all flooded grid cells that have no direct connectivity to any of the input stage data locations. This is accomplished in ArcView® Spatial Analyst Extension by using the function CostDistance(). The

function CostDistance() calculates the least- accumulative-cost distance over a surface to a set of source points and also produces a direction surface for each cell to its closest source point. In this application the source points are the stage locations in the main river channel, vertices of the center lines, and vertices of secondary channels. The direction surface shows the overland flow path from each cell to the closest flood water source.

Next the first approximate flood surface and the direction surface are used to determine the flood surface elevation in each flooded cell. The direction surface is sampled at user-defined regular intervals and analyzed to define the shortest water flow path from a cell back to the source point across the flood surface. The water surface elevation of the source point is assigned to the cell and all cells are traversed. A default value of 25 allows a 25 by 25 sampling grid over the area. Since this is a uniform sampling, increasing this sampling can provide for improved results. If the sampling interval is too large, many small backwater features (streams, channels) may not be sampled and thereby not depict flooding of backwater areas.

Using the water surface elevation of the sample cells, all the cells in the flow path, the original gage elevations, center line, and secondary channel vertices, a second approximate flood surface is interpolated. This surface is generated at the same cell size as the ground elevation model using the inverse distance weighted interpolation method. This surface is generated without regard to topography.

This second approximated flood surface is then modified to eliminate all flooded grid cells that do not have direct connectivity to any of the input water elevation source point locations. This is the same process as was applied to the first approximated flood surface using the same functions. The resulting distance surface is converted to the final flood area surface.

The final flood area surface is added to the current view in ArcView and the process is terminated.

Appendix A
Model Listing

DRIVER4.AVE

```
' Name: LMK.Driver4
' Name: Driver4
' ==========================================================
'
'
' Name: Driver4.ave
'
'
' Purpose: Test Avenue script driver for user input of line and point
' shape files
'
'
' Calls: LMK.line2dpts, LMK.GetPointTheme, LMK.Water-Flow
' LNK.streamline
'
' Comments: This is different from Driver3 in that it includes
' interpolated water surface values on channel centerlines
'
'
' ==========================================================
'
' Developer: USA ERDC-WES/ Mr. Jerry Ballard
' 3909 Halls Ferry Road
' Vicksburg, MS 39180
'
'
' Sponsor: CELMK-ED-H (Mr. Ron Goldman)
' 4155 Clay Street
' Vicksburg, MS 39180-3435
'
' ==========================================================
'
' Notes:
'
```

```
' 3/25/1998 - fixed problem with deleting the direction grid from first
' cost distance iteration. For some reason it can only be deleted
' before it is used. After it is generated, it cannot be deleted
' from the same script. -jrb
'
' 3/25/1998 - Added the item name to the final flood surface. Not
' really sure what will happen if there is a combined point and
' transect inputs. -jrb
'
' 10/15/1998 - Rats! ArcView3.1 upgrade undid my fix on the direction
' grid.
'
' 10/15/1998 - Added the LMK.steamline to interpolate water surface
' elevations along channel centerlines (just like FIST!).
'
' 10/21/1998 - Added the LMK.Secondary to add flat water surface
' elevations along secondary channels.
'
theView = av.GetActiveDoc
'
' ==========================================================
' Get a list of Grid Themes and have user select one
' ==========================================================
'
' Get a list of all the Themes
'
lstAllThemes = theView.GetThemes
lstGridThemes = list.Make
'
' Get a list of all available Grid Themes
'
for each thm in lstAllThemes
if ( thm.IS(GTheme)) then
lstGridThemes.Add(thm)
end
end
'
' Did we find any?
'
if ( lstGridThemes.Count = 0 ) then
MsgBox.Error("There are NO Grid Themes in this View.","Returning")
return(nil)
end
'
' Prompt the user to select one of the Grid Themes
'
thmElevTheme = MsgBox.List(lstGridThemes,
"Select the Grid Theme that contains the terrain elevations,"
"Terrain Elevation Grid")
'
```

```
' Check for cancelation of procedure
'
if ( thmElevTheme = nil ) then
return(nil)
end
' ==========================================================
' Get a list of Point Themes and let the user possibly
' select one for processing.
' ==========================================================
lstPointThemeInfo = av.Run("LMK.GetPointTheme",{})
' ==========================================================
' If a point theme of water gages was selected, then we can
' ask for the theme of polylines for the channel center-line
' interpolation.
' ==========================================================
if ( lstPointThemeInfo.Count > 1) then
lstInterpPointThemeInfo = av.Run("LMK.streamline",lstPointThemeInfo)
' thmInterpPointTheme = lstInterPointThemeInfo.Get(0)

lstSecondPointThemeInfo = av.Run("LMK.Secondary",lstInterpPointThemeInfo)

end
' ==========================================================
' Get a list of Line Themes (cross-sectional input) and let the
' user select one and also the item of the theme to use
' ==========================================================
lstLineThemeInfo = av.Run("LMK.line2dpts",{})

' ==========================================================
' Prompt for the sampling steps for use in LMK.Water-Flow
' Default is 25
' ==========================================================


strSampleSteps =
MsgBox.Input("Specify Sampling Steps for Water Flow Calculation,"
"Specify Sampling Steps for Water Flow Calculation,"
"25")

if ( (strSampleSteps.IsNumber) or (strSampleSteps.AsNumber > 0) ) then
numSampleSteps = strSampleSteps.AsNumber
else
MsgBox.Error("Sampling Steps Provided is not numeric. Returning,"
"Error in Input")
return{nil}
end

' ==========================================================
' Prompt for the additive constant for the initial watersurface
' Default is 0
' ==========================================================
```

```
strAddElev = MsgBox.Input(
"Specify Additive Constant for Initial Surface in Meters,"
"Specify Additive Constant for Initial Surface,"
"0")

if ( strAddElev.IsNumber ) then
numAddElev = strAddElev.AsNumber
else
MsgBox.Error("Additive Constant entered is not numeric. Returning,"
"Error in Input")
return {nil}
end

' ==========================================================
' Some convoluted logic here.....
'
' If neither a point nor line shape file was selected, then cancel the
' whole process.
'
' If just the point shapefile was selected, then continue as planned.
'
' If just the line shapefile was selected, then continue as planned.
'
' If *BOTH* were selected, then we have to merge the point and line
' shapefile into a single point coverage. UGH. But fortunately the
' line shapefile was just converted to a shapepoint file so we can just
' keep adding features to the shapefile and then continue.
'
' ==========================================================

strFieldName = "" 'Kludge to keep field name from point shapefile

if ( (lstPointThemeInfo.Count < 3) and (lstLineThemeInfo.Count < 2) ) then
MsgBox.Info("No Theme inputs selected. Ending Procedure,"
"Procedure Termination")
return nil
end

if ( (lstPointThemeInfo.Count > 1) and (lstLineThemeInfo.Count < 2) ), then
Av.ShowMsg("Using only the Point Theme")
'
' Copy/Clone the objects into required variables
'
thmPointTheme = lstPointThemeInfo.Get(0)
ftbPointTheme = thmPointTheme.GetFTab
fldPointThemeStage = lstPointThemeInfo.Get(1)
numPointThemeConv = lstPointThemeInfo.Get(2)
fldPointThemeShape = ftbPointTheme.FindField("Shape")
strFieldName = fldPointThemeStage.AsString
end
```

```
if ( (lstPointThemeInfo.Count < 3) and (lstLineThemeInfo.Count > 1) ), then
Av.ShowMsg("Using only the Line Theme")
'
' Copy the list objects into required variables
'
thmPointTheme = lstLineThemeInfo.Get(0)
fldPointThemeStage = lstLineThemeInfo.Get(1)
ftbPointTheme = thmPointTheme.GetFTab
numPointThemeConv = 1.0

end

if (( lstPointThemeInfo.Count > 1) and (lstLineThemeInfo.Count > 1) ), then
Av.ShowMsg("Merging line and point themes")

thmUserPointTheme = lstPointThemeInfo.Get(0)
ftbUserPointTheme = thmUserPointTheme.GetFTab
fldUserPointThemeStage = lstPointThemeInfo.Get(1)
numUserPointConv = lstPointThemeInfo.Get(2)
fldUserPointThemeShape = ftbUserPointTheme.FindField("Shape")

' remember actually this theme is a point theme generated from the
' selected line theme

thmUserLineTheme = lstLineThemeInfo.Get(0)
fldUserLineThemeStage = lstLineThemeInfo.Get(1)

ftbUserLineTheme = thmUserLineTheme.GetFTab
fldUserLineThemeShape = ftbUserLineTheme.FindField("Shape")

' hopefully we have all the info now to merge this stuff
'
' Read though the point shapefile and copy the records into
' the output file.

for each rec in ftbUserPointTheme
shpUserPointObject = ftbUserPointTheme.ReturnValue
( fldUserPointThemeShape, rec)
numUserPointStage = ftbUserPointTheme.ReturnValueNumber
( fldUserPointThemeStage, rec)
numUserPointStage = numUserPointStage * numUserPointConv

numPtRec = ftbUserLineTheme.AddRecord
ftbUserLineTheme.SetValueNumber( fldUserLineThemeStage, numPtRec,
numUserPointStage)
ftbUserLineTheme.SetValue( fldUserLineThemeShape, numPtRec,
shpUserPointObject)
end
'
```

```
' Copy/Clone the objects into required variables
'
thmPointTheme = thmUserLineTheme.Clone
ftbPointTheme = ftbUserLineTheme
fldPointThemeStage = fldUserLineThemeStage
fldPointThemeShape = fldUserLineThemeShape
numPointThemeConv = 1.0

end ' End of conditional using both line and point themes
'
' ******************************************************
' ******************************************************
' ******************************************************
' Additional logic for adding/merging line and point themes
' specifically for the center line interpolation
'
'
' (1) if a line theme was not selected, then bail.

if ( lstInterpPointThemeInfo.Get(0).Is( Theme ) ), then

' (2) From the returned list, get Theme, Ftab, and Field

thmInterpPointTheme = lstInterpPointThemeInfo.Get(0)
ftbInterpPointTheme = thmInterpPointTheme.GetFTab
ftbInterpPointTheme.Flush
fldInterpPointThemeStage = ftbInterpPointTheme.FindField("Stage")
fldInterpPointThemeShape = ftbInterpPointTheme.FindField("Shape")

' (3) The previous section of code provides a point theme
' ready-to-go. We really aren't sure what this point
' theme is, so we need to append it to ours.

' Important variables:
' thmPointTheme - Point Theme
' ftbPointTheme - FTab of point theme
' fldPointThemeStage - Field of data values
' fldPointThemeShape - Shape of the Theme (it really is important)
'
' Step through each record of the previous point theme
for each rec in ftbPointTheme
' Get the shape
shpPointTheme = ftbPointTheme.ReturnValue(fldPointThemeShape,rec)
' Get the value (number)
numPointTheme = ftbPointTheme.ReturnValueNumber(fldPointThemeStage,rec)
numPointTheme = numPointTheme * numPointThemeConv

' Append a record to the interpolated point file
numPtRec = ftbInterpPointTheme.AddRecord
```

```
' Set the shape
ftbInterpPointTheme.SetValue(fldInterpPointThemeShape, numPtRec,
shpPointTheme)
' Set the value
ftbInterpPointTheme.SetValueNumber(fldInterpPointThemeStage, numPtRec,
numPointTheme)
end

'
' Process the secondary channels if any
'
if ( lstSecondPointThemeInfo.Get(0).Is( Theme ) ) then
thmSecondPointTheme = lstSecondPointThemeInfo.Get(0)
ftbSecondPointTheme = thmSecondPointTheme.GetFTab
ftbSecondPointTheme.Flush
fldSecondPointThemeStage = ftbSecondPointTheme.FindField("Stage")
fldSecondPointThemeShape = ftbSecondPointTheme.FindField("Shape")
for each rec in ftbSecondPointTheme
' Get the shape
shpSecondPointTheme =
ftbSecondPointTheme.ReturnValue(fldSecondPointThemeShape,rec)
' Get the value (number)
numSecondPointTheme =
ftbSecondPointTheme.ReturnValueNumber(fldSecondPointThemeStage,rec)
' numSecondPointTheme = numSecondPointTheme * numPointThemeConv

' Append a record to the interpolated point file
numPtRec = ftbInterpPointTheme.AddRecord

' Set the shape
ftbInterpPointTheme.SetValue(fldInterpPointThemeShape, numPtRec,
shpSecondPointTheme)
' Set the value
ftbInterpPointTheme.SetValueNumber(fldInterpPointThemeStage, numPtRec,
numSecondPointTheme)

end
end

' (4) That should do it. Now we need to copy/clone the
' objects for consistency.
'
' Copy/Clone the objects into required variables
'
thmPointTheme = thmInterpPointTheme.Clone
ftbPointTheme = ftbInterpPointTheme
fldPointThemeStage = fldInterpPointThemeStage
fldPointThemeShape = fldInterpPointThemeShape
numPointThemeConv = 1.0
```

```
end ' of center of streamline interpolation points
' ******************************************************
' ******************************************************
' ******************************************************

' ==========================================================
'
' Ok. We managed to navigate though the mess of getting or merging all
' necessary shapefiles. Important variables up to this point:
' *IMPORTANT VARIABLES*
'
' thmPointTheme - the point shape file for determine water surface elevation
' ftbPointTheme - the FTab of that shapefile
' fldPointThemeStage - the item name or field containing the numeric values
' of the water elevation
' numPointThemeConv - conversion factor for the units of water elevation
'
' thmElevTheme - the Grid (GTheme) Theme containing the ground (terrain)
' elevation values in meters
'
' ==========================================================
'
' Make ArcView dump everything in memory about ftbPointTheme to disk
' this may or may not be necessary depending on how we got this point shapefile
'
ftbPointTheme.Flush

Av.ClearMsg

'
' ==========================================================
'
' Prepare for the initial interpolation
'
' ==========================================================


thePrj = theView.GetProjection

'
' Access the terrain elevation grid theme
'

grdTopoElev = thmElevTheme.GetGrid

'
' Do all initial interpolation at a large cell size to save time
' (we'll use the correct cell size at the final interpolation)
numCellSize = 100
'numCellSize = grdTopoElev.GetCellSize
```

```
'
' Set analysis area
'

Grid.SetAnalysisCellSize(#GRID_ENVTYPE_VALUE,numCellSize)
Grid.SetAnalysisExtent(#GRID_ENVTYPE_VALUE, theView.ReturnExtent)

'
' Convert the Water Surface Point theme to a grid
'
' This converted Grid theme will be used later for calculating cost and
' flow direction
'


fldIDPointTheme = ftbPointTheme.FindField("Id")
grdPointTheme = Grid.MakeFromFTab(ftbPointTheme, thePrj,
fldIDPointTheme, NIL)
gthmPointTheme = GTheme.Make(grdPointTheme)
gthmPointTheme.SetName("Gauge Grid")

'
' Set the radius and interpolation for surface generation
'

theRadius = Radius.MakeVariable(12,nil)
theInterp = Interp.MakeIDW(2,theRadius,nil)

'
'
' ==========================================================
'
' This provides a water surface elevation determined by least linear distance
'

'
' Interpolate the initial surface
'

grdInitialSurface = Grid.MakeByInterpolation(
ftbPointTheme, thePrj, fldPointThemeStage, theInterp, NIL)

if ( grdInitialSurface.HasError) then
MsgBox.Error("Initial Interpolation Failed.","Error")
return nil
end

'
' Access the terrain elevation grid theme
'
```

```
grdTopoElev = thmElevTheme.GetGrid

'
' Determine the initial flooded surface
'

' Note: Don't forget to convert vertical units here!!
'

grdRoughInitFlood =
grdTopoElev <= ((grdInitialSurface * numPointThemeConv.AsGrid) +
numAddElev.AsGrid)
if ( grdRoughInitFlood.HasError ) then
MsgBox.Error("Initial Flood Delination Failed.","Error")
return nil
end

'
' Get rid of the stupid zeros (sigh)
'

thmRoughInitFlood = GTheme.Make(grdRoughInitFlood)
grdQGrid = grdRoughInitFlood.ExtractByAttributes("([value] = 1)")

thmInitSurface = GTheme.Make(grdInitialSurface)
thmQGrid = GTheme.Make(grdQGrid)

thmInitSurface.SetName("InitSurf")
thmInitSurface.SetComments(fldPointThemeStage.AsString)

theView.AddTheme(thmInitSurface)
theView.AddTheme(thmQGrid)
theView.AddTheme(thmPointTheme)

'
' ==========================================================
'
'
' ==========================================================
'

'
' Determine all water areas connected to main line stream
' - this effectively gets rid of floods in protected areas
'
prjProject = av.GetProject
flnWorkDir = prjProject.GetWorkDir
flnDirGrid = FileName.Merge(flnWorkDir.AsString,"dirgrid")
```

```
if ( File.Exists(flnDirGrid) ) then
if ( Grid.DeleteDataSet(flnDirGrid) ) then
'MsgBox.Info("Grid Deleted!!!","")
else
'MsgBox.Info("Grid *not* deleted.","")
end
end

'
' ==========================================================
'
' Calculate an improved water surface by calculating water surface elevation
' by least path distance (not necessarily a linear distance)

'
'
' ******************************************************
'
' Don't be confused here: !!!!!!!!!!!!!!!!!!!!!!
'
' grdQGrid - has 1 where the water is flooded. This is
' needed for CostDistance()
'
' grdInitialSurface - has the water surface elevations for
' the entire area extent.
' ******************************************************

'
' Calculate cost surface and direction grid
'

grdCostSurface = grdPointTheme.CostDistance(grdQGrid,flnDirGrid,NIL,NIL)

thmCostSurface = GTheme.Make(grdCostSurface)
thmCostSurface.SetName("Cost Surface")

theView.AddTheme(thmCostSurface)

'
'
' ==========================================================
'

'
' Using the direction grid, calculate water flow paths
'
' Call LMK.Water-Flow with Direction Grid, Initial Surface, ftbPointTheme,
' and fldPointThemeStage
'
' This returns an FTheme with point water surface elevations.
```

```
'

thmSurfPts = av.Run("LMK.Water-Flow",{flnDirGrid,grdInitialSurface,
ftbPointTheme, fldPointThemeStage,
numSampleSteps} )

'
' Try to delete the nasty DirGrid
'
'if ( File.Exists(flnDirGrid) ) then

' if ( File.CanDelete(flnDirGrid) ) then
' Grid.DeleteDataSet(flnDirGrid)
' else
' MsgBox.Error("Cannot remove temporary grid file "+flnDirGrid.AsString,
' "Unable to continue")
' return nil
' end
'end

'
' See if the Water-Flow script did its job
'
if ( thmSurfPts = Nil ) then
MsgBox.Error("Flow Direction Calculation Failed.","Stopping Procedure")
exit
end


ftbSurfPts = thmSurfPts.GetFTab
ftbSurfPts.Flush

'theView.AddTheme(thmSurfPts)

thmSurfPts.Invalidate(True)

Grid.SetAnalysisCellSize(#GRID_ENVTYPE_VALUE,numCellSize)
Grid.SetAnalysisExtent(#GRID_ENVTYPE_VALUE, theView.ReturnExtent)

'
' Interpolate new surface with the surface points
'
fldSurfPtsStage = ftbSurfPts.FindField("Stage")

'
' Set the radius and interpolation for surface generation
'

theRadius = Radius.MakeVariable(12,nil)
theInterp = Interp.MakeIDW(2,theRadius,nil)
```

```
'
' Interpolate the second water surface surface
'

grdSecondSurface = Grid.MakeByInterpolation(
ftbSurfPts, thePrj, fldSurfPtsStage, theInterp, NIL)

if ( grdSecondSurface.HasError) then
MsgBox.Error("Second Interpolation Failed.","Error")
return nil
end

'
'
' Determine new flooded surface
'
' Change cell size from 50 to elevation grid cell size
'
numCellSize = grdTopoElev.GetCellSize

'
' Set analysis area
'
Grid.SetAnalysisCellSize(#GRID_ENVTYPE_VALUE,numCellSize)
Grid.SetAnalysisExtent(#GRID_ENVTYPE_VALUE, theView.ReturnExtent)

'
' Note: Don't forget that the elevation file vertical units are in feet!

grdSecondFlood = grdTopoElev <= (grdSecondSurface *
numPointThemeConv.AsGrid)
if ( grdSecondFlood.HasError ) then
MsgBox.Error("Second Flood Delination Failed.","Error")
return nil
end

'
' Get rid of the stupid zeros (sigh)
'

thmSecondFlood = GTheme.Make(grdSecondFlood)
grdCleanSecondFlood = grdSecondFlood.ExtractByAttributes("([value] = 1)")

thmCleanSecondFlood = GTheme.Make(grdCleanSecondFlood)
thmCleanSecondFlood.SetName("Flooded Area")

thmSecondSurface = GTheme.Make(grdSecondSurface)
thmSecondSurface.SetName("Water Surface")
theView.AddTheme(thmSecondSurface)
```

```
thmSecondFlood.SetName("Improved Surf")

theView.AddTheme(thmSecondFlood)
theView.AddTheme(thmCleanSecondFlood)

'
' ==========================================================
'
' Calculate the second improved water surface, by calculating water surface
elevation
' by least path distance (not necessarily a linear distance)
'

'
' Calculate cost surface only (We don't need the direction grid anymore)
'
flnWorkDir = prjProject.GetWorkDir
flnNewDirGrid = FileName.Merge(flnWorkDir.AsString,"dirgrid1")

'if ( File.Exists(flnNewDirGrid) ) then
' if ( Grid.DeleteDataSet(flnNewDirGrid) ) then
' MsgBox.Info("Grid Deleted!!!","")
' else
' MsgBox.Info("Grid *not* deleted.","")
' end
'end


grdCostSurface2 =
grdPointTheme.CostDistance(grdCleanSecondFlood,NIL,NIL,NIL)

thmCostSurface2 = GTheme.Make(grdCostSurface2)
thmCostSurface2.SetName("Cost Surface2")
thmCostSurface2.SetActive(true)

'theView.AddTheme(thmCostSurface2)


Av.ShowMsg("Extracting by Attributes")

grdFinalFlood = grdCostSurface2 > 0.AsGrid

thmFinalFlood = GTheme.Make(grdFinalFlood)

if ( strFieldName.IsNull ) then
thmFinalFlood.SetName("Flood Surface: "+fldPointThemeStage.AsString)
else
thmFinalFlood.SetName("Flood Surface: "+strFieldName)
end
```

```
theView.AddTheme(thmFinalFlood)
'
'
```

LMK.GetPointTheme.AVE

```
' Name: LMK.GetPointTheme

theView = Av.GetActiveDoc

'
_____________________________________________________________________

' Get a list of all Themes
'
lstAllThemes = theView.GetThemes

'
' Make a list to hold all feature themes
'
lstFeatureList = list.Make

'
' Find the point FThemes
'
for each th in lstAllThemes
if ( th.IS(FTheme)) then
ftbUTheme = th.GetFTab
shapefield = ftbUTheme.FindField("Shape")
shapetype = shapefield.GetType
if ( shapetype = #FIELD_SHAPEPOINT ) then
lstFeatureList.Add(th)
end
end
end

'
' Check to make sure the list is not empty
'
if ( lstFeatureList.Count = 0 ) then
MsgBox.Error("There are NO Point Feature Themes in this View","Error")
return{nil}

end

thmPointTheme = MsgBox.List(lstFeatureList,
"Select the Point Feature Theme that contains river surface elevations,"
"River Surface Point Theme")

if ( thmPointTheme = nil ), then
return {nil}
```

```
end

'
' Prompt for the field that contains water elevations
'
ftbPointTheme = thmPointTheme.GetFTab
lstAllPointFields = ftbPointTheme.GetFields
lstPointFields = List.Make

for each fld in lstAllPointFields
if ( fld.IsTypeNumber ), then
lstPointFields.Add(fld)
end
end

if ( lstPointFields.Count = 0 ), then
MsgBox.Error("There are NO numeric attributes attached for the points.","Error,
Returning")
return{nil}
end

fldPointStage = MsgBox.List(lstPointFields,
"Select field that contains the river elevations,"
"Select Field")

'
' Make sure a field was selected
'
if ( fldPointStage = nil ), then
MsgBox.Error("No Field Selected. Returning.","Error in Field Selection")
return{nil}
end

'
' Make sure the field is numeric
'
if ( fldPointStage.IsTypeNumber.Not ), then
MsgBox.Error("Field selected is not a numeric field. Ending Procedure,"
"Error: Non-numeric field for point theme")
return{nil}
end


'
' Ask for the units

boolUnits = MsgBox.YesNo("Are the elevations in feet?","Elevation Units",true)

'
' Set the conversion factor
```

```
'
if ( boolUnits = true ), then
numConvPointTheme = 0.3048
else
numConvPointTheme = 1.0
end

return { thmPointTheme, fldPointStage, numConvPointTheme }
```

LMK.Streamline.AVE

```
' Name: LMK.streamline
'
' Name: LMK.streamline
'
' Purpose: Convert line shapelfile into a densified point shape file with
' interpolated water elevation attributes
'
' Author: Jerry Ballard
' USACE Waterways Experiment Station
' Vicksburg, MS 39180
'
' Sponsor: CELMK-ED-H (Mr. Ron Goldman)
' 4155 Clay Street
' Vicksburg, MS 39180-3435
'
' Initial Coding: October 14,1998
'
' ==========================================================
'
' Arguments: (a list)
' 0 thmPointTheme - Gage Point Theme
' 1 fldPointTheme - Water Elevation Field to use from Gage Point Theme
' 2 numPointThemeConv - Conversion Factor (feet to meters, m to ft.)
'
' Returns: (a list)
' 0 thmInterpPointTheme - Interpolated Gage Point Theme
'
' Calls:
'
' Is Called By: LMK.DriverX.ave
'
'
' ==========================================================
'
' Get current view
'
theProject = Av.GetProject
```

```
theView = Av.GetActiveDoc

if ( theView = nil ) then
MsgBox.Info("The View was not found. Returning","Error")
return{nil}
end

'
' Get point field info from past arguments
'

thmPointTheme = Self.Get(0)
ftbPointTheme = thmPointTheme.GetFTab
fldPointThemeStage = Self.Get(1)
numPointThemeConv = Self.Get(2)

'
' Get a list of all Themes
'
lstAllThemes = theView.GetThemes

'
' Make a list to hold all feature Themes
'
lstFeatureList = list.Make

'
' Find the line FThemes
'
for each th in lstAllThemes
if ( th.IS(FTheme)) then
ftbUTheme = th.GetFTab
shapefield = ftbUTheme.FindField("Shape")
shapetype = shapefield.GetType
if ( shapetype = #FIELD_SHAPELINE ) then
lstFeatureList.Add(th)
end
end
end

'
' Check to make sure the list is not empty
'
if ( lstFeatureList.Count = 0 ) then
MsgBox.Error("There are NO Line Feature Themes in this View","Error")
return{nil}
end

thmLineTheme = MsgBox.List(lstFeatureList,
"Select the Line Feature Theme that contains channel centerlines,"
```

```
"Channel Surface Line Theme")

if ( thmLineTheme = nil ), then
return{nil}
end


'
' Prompt for the field that contains starting gage id
'
ftbLineTheme = thmLineTheme.GetFTab
lstLineFields = ftbLineTheme.GetFields

if ( lstLineFields.Count < 3 ), then
MsgBox.Error("There are NO attributes attached for the lines," "Error,
Returning")
return{nil}
end

fldStartLineStage = MsgBox.List(lstLineFields,
"Select field that contains the starting gage ids,"
"Select Field")

'
' Make sure a field was selected
'
if ( fldStartLineStage = nil ), then
MsgBox.Error("No Field Selected. Returning.","Error in Field Selection")
return{nil}
end

fldEndLineStage = MsgBox.List(lstLineFields,
"Select field that contains the ending gage ids,"
"Select Field")

'
' Make sure a field was selected
'
if ( fldEndLineStage = nil ), then
MsgBox.Error("No Field Selected. Returning.","Error in Field Selection")
return{nil}
end

strDensityInterval = MsgBox.Input("Specify Density Interval for Stream
Centerlines (meters),"
"Density Interval","1000")

if ( strDensityInterval.IsNumber ), then
numDensityInterval = strDensityInterval.AsNumber
else
```

```
MsgBox.Error("Interval Specified is not numeric. Returning.","Error in Density
Interval")
return{nil}
end

'
' Specify the output shapefile...
'
'
' Create a temporary shape file for the water points
'
flnCWD = theProject.GetWorkDir
shpName = flnCWD.MakeTmp("cline","shp")

'defaultName = FileName.Make("$HOME").MakeTmp("shape","shp")
'shpName = FileDialog.Put( defaultName,"*.shp","Output Shape File" )
if (shpName = nil) then
exit
end

shpName.SetExtension("shp")
ftbPointFTab = Ftab.MakeNew(shpName,Point)
lstFields = List.Make
lstFields.Add(Field.Make("Stage",#FIELD_FLOAT, 16,2))
ftbPointFTab.AddFields(lstFields)
fldPointShape = ftbPointFTab.FindField("Shape")
fldPointStage = ftbPointFTab.FindField("Stage")

'
' Process each line
'

fldLineShape = ftbLineTheme.FindField("Shape")
fldPointThemeID = ftbPointTheme.FindField("Id")

for each rec in ftbLineTheme

' Get Starting and Ending Stage IDs for lookup
'
numStartLineStage = ftbLineTheme.ReturnValueNumber( fldStartLineStage,
rec)
numEndLineStage = ftbLineTheme.ReturnValueNumber( fldEndLineStage, rec)

'
' Search the point gage FTab and find the actual water level at each stage
'
for each srec in ftbPointTheme
numID = ftbPointTheme.ReturnValueNumber( fldPointThemeID, srec )
```

```
if ( numID = numStartLineStage ), then
numStartWaterStage = ftbPointTheme.ReturnValueNumber(
fldPointThemeStage, srec )
numStartWaterStage = numStartWaterStage * numPointThemeConv
end
if ( numID = numEndLineStage ), then
numEndWaterStage = ftbPointTheme.ReturnValueNumber(
fldPointThemeStage, srec )
numEndWaterStage = numEndWaterStage * numPointThemeConv
end

end
'
' Get the shape object (it should be a PolyLine)
'
shpLineObject = ftbLineTheme.ReturnValue( fldLineShape, rec )
'
' Densify at an interval
'
plineDensified = shpLineObject.ReturnDensified(numDensityInterval)
'
' Get the length
'
numPlineLength = plineDensified.ReturnLength
'
' Compute water slope
'
numWaterSlope = ( numEndWaterStage - numStartWaterStage ) /
numPlineLength
'msgbox.report(numPlineLength.AsString + " " + numWaterSlope.AsString,"")

'
' Convert the polyline to a list to extract points
'
lstPolyLine = plineDensified.AsList

for each shapePart in lstPolyLine
for each xyPoint in shapePart
numPosition = plineDensified.PointPosition( xyPoint ) / 100.0
numWaterLevel = numStartWaterStage + (numWaterSlope * numPosition *
numPlineLength )

numPtrec = ftbPointFTab.AddRecord
ftbPointFTab.SetValueNumber( fldPointStage, numPtrec, numWaterLevel )
ftbPointFTab.SetValue( fldPointShape, numPtrec, xyPoint )
end
end
end
```

```
thmInterpPointTheme = FTheme.Make(ftbPointFTab)
return {thmInterpPointTheme}
```

LMK.Secondary.AVE

```
' Name: LMK.Secondary
'
' Arguments: (a list)
' 0 thmPointTheme - Gage Point Theme
' 1 fldPointTheme - Water Elevation Field to use from Gage Point Theme
' 2 numPointThemeConv - Conversion Factor (feet to meters, m to ft.)
'
' Get current view
'
theProject = Av.GetProject

theView = Av.GetActiveDoc

if ( theView = nil ) then
MsgBox.Info("The View was not found. Returning","Error")
return{nil}
end

'
' Get point field info from past arguments
'
thmPointTheme = Self.Get(0)
ftbPointTheme = thmPointTheme.GetFTab
fldPointThemeStage = ftbPointTheme.FindField("Stage")

‘
' Get a list of all Themes
'
lstAllThemes = theView.GetThemes

'
' Make a list to hold all feature Themes
'
lstFeatureList = list.Make

'
' Find the line FThemes
'
for each th in lstAllThemes
if ( th.IS(FTheme)), then
ftbUTheme = th.GetFTab
shapefield = ftbUTheme.FindField("Shape")
shapetype = shapefield.GetType
```

```
if ( shapetype = #FIELD_SHAPELINE ) then
lstFeatureList.Add(th)
end
end
end

'
' Check to make sure the list is not empty
'
if ( lstFeatureList.Count = 0 ), then
MsgBox.Error("There are NO Line Feature Themes in this View","Error")
return{nil}
end

thmLineTheme = MsgBox.List(lstFeatureList,
"Select the Line Feature Theme that contains secondary channels,"
"Secondary Channel Line Themes")

if ( thmLineTheme = nil ), then
return{nil}
end

'
'
ftbLineTheme = thmLineTheme.GetFTab
fldLineShape = ftbLineTheme.FindField("Shape")
fldPointShape = ftbPointTheme.FindField("Shape")
'
' Create a temporary shape file for the secondary water points
'
flnWorkDir = theProject.GetWorkDir
flnSecondPointTheme = flnWorkDir.MakeTmp("sline","shp")

ftbSecondPointFTab = FTab.MakeNew(flnSecondPointTheme,Point)
lstFields = List.Make
lstFields.Add(Field.Make("Stage",#FIELD_FLOAT,16,2))
ftbSecondPointFTab.AddFields(lstFields)
fldSecondPointShape = ftbSecondPointFTab.FindField("Shape")
fldSecondPointStage = ftbSecondPointFTab.FindField("Stage")

'Process each line

for each rec in ftbLineTheme

shpLineObject = ftbLinetheme.ReturnValue( fldLineShape, rec )
'MsgBox.Report(shpLineObject.AsString,"")
plineDens = shpLineObject.ReturnDensified(150)
'lstPolyLine = shpLineObject.AsList
lstPolyLine = plineDens.AsList
```

```
for each shapePart in lstPolyLine

xyPointFirst = shapePart.Get(0)
'MsgBox.Info(xyPointFirst.AsString,"")
ftbPointTheme.SelectByPoint(xyPointFirst,100.0,#VTAB_SELTYPE_NEW)

if ( ftbPointTheme.GetSelection.Count > 0 ) then
setPointTheme = ftbPointTheme.GetSelection
for each prec in setPointTheme
numPointStage = ftbPointTheme.ReturnValueNumber(
fldPointThemeStage,prec)
'MsgBox.Info(numPointStage.AsString,"")
end
end

for each xyPoint in shapePart
numPtrec = ftbSecondPointFTab.AddRecord
ftbSecondPointFTab.SetValueNumber( fldSecondPointStage, numPtrec,
numPointStage )
ftbSecondPointFTab.SetValue( fldSecondPointShape, numPtrec, xyPoint )
'MsgBox.Info(xyPoint.AsString,"")
end
end
end

thmSecondPointTheme = FTheme.Make(ftbSecondPointFTab)
ftbPointTheme.Flush

return {thmSecondPointTheme, 1}
```

LMK.line2dpts.ave

```
' Name: LMK.line2dpts
' Name: line2dpts.ave
'
' Purpose: Convert line shape file into a densified point shape file with
' water elevation attributes
'
'
' Get current project
'
theProject = Av.GetProject

theView = Av.GetActiveDoc

if ( theView = nil ) then
MsgBox.Info("The View was not found. Returning","Error")
return{nil}
end
```

```
'
' Get a list of all Themes
'
lstAllThemes = theView.GetThemes

'
' Make a list to hold all feature Themes
'
lstFeatureList = list.Make

'
' Find the line FThemes
'
for each th in lstAllThemes
if ( th.IS(FTheme)) then
ftbUTheme = th.GetFTab
shapefield = ftbUTheme.FindField("Shape")
shapetype = shapefield.GetType
if ( shapetype = #FIELD_SHAPELINE ) then
lstFeatureList.Add(th)
end
end
end

'
' Check to make sure the list is not empty
'
if ( lstFeatureList.Count = 0 ) then
MsgBox.Error("There are NO Line Feature Themes in this View","Error")
return{nil}
end

thmLineTheme = MsgBox.List(lstFeatureList,
"Select the Line Feature Theme that contains river cross section elevations",
"River Cross Section Line Theme")

if ( thmLineTheme = nil ) then
return{nil}
end

'
' Prompt for the field that contains water elevations
'
ftbLineTheme = thmLineTheme.GetFTab
lstLineFields = ftbLineTheme.GetFields

if ( lstLineFields.Count < 3 ) then
MsgBox.Error("There are NO attributes attached for the lines.","Error,
Returning")
return{nil}
```

```
end

fldLineStage = MsgBox.List(lstLineFields,
"Select field that contains the river elevations,"
"Select Field")

'
' Make sure a field was selected
'
if ( fldLineStage = nil ) then
MsgBox.Error("No Field Selected. Returning.","Error in Field Selection")
return{nil}
end

'
' Ask for the units
'
boolUnits = MsgBox.YesNo("Are the elevations in feet?","Elevation Units",true)

'
' Set the conversion factor
'
if ( boolUnits = true ) then

numConvLineTheme = 0.3048
else
numConvLineTheme = 1.0
end

'
' Prompt for new shape file name
'
' Specify the output shapefile...
'
defaultName = FileName.Make("$HOME").MakeTmp("shape","shp")
shpName = FileDialog.Put( defaultName,"*.shp","Output Shape File" )
if (shpName = nil), then
exit
end

shpName.SetExtension("shp")
ftbPointFTab = Ftab.MakeNew(shpName,Point)
lstFields = List.Make
lstFields.Add(Field.Make("Stage",#FIELD_FLOAT, 16,2))
ftbPointFTab.AddFields(lstFields)
fldPointShape = ftbPointFTab.FindField("Shape")
fldPointStage = ftbPointFTab.FindField("Stage")

'
' Process each line
```

```
'

fldLineShape = ftbLineTheme.FindField("Shape")

for each rec in ftbLineTheme

'
' Get the line object and its field attribute
'
shpLineObject = ftbLineTheme.ReturnValue( fldLineShape, rec )
numLineStage = ftbLineTheme.ReturnValueNumber( fldLineStage, rec)
numLineStage = numLineStage * numConvLineTheme

'msgbox.report(shpLineObject.AsString,"")
'
' Densify the line to my liking (20-m)
'
plineDensified = shpLineObject.ReturnDensified(20)

'
' Convert the polyline to a list to extract points
'
lstPolyLine = plineDensified.AsList

for each shapePart in lstPolyLine
for each xyPoint in shapePart
numPtrec = ftbPointFTab.AddRecord
ftbPointFTab.SetValueNumber( fldPointStage, numPtrec, numLineStage )
ftbPointFTab.SetValue( fldPointShape, numPtrec, xyPoint )
end
end
'msgbox.report(plineDensified.AsString,"")
end


thmPointTheme = FTheme.Make(ftbPointFTab)

return {thmPointTheme, fldPointStage }
```

LMK.Water-Flow.ave

```
' Name: LMK.Water-Flow
'
' Name: LMK.Water-Flow
'
' Title: Calculate shortest water flow path
'
' Topics: CostDistance, CellValue
'
```

```
' Description: This script calculates the shortest water flow path of a flood
' surface using a direction grid from CostDistance.
'
' Requires: Spatial Analyst
'
' Self:
' (0) File name of the direction grid
' (1) Grid of water surface elevations
' (2) FTab of the stage point theme
' (3) Field of surface elevations of the stage point theme
' (4) Number of Sample Steps
'
' Returns: Theme composed on points with attributed elevations
'
' Author: Jerry Ballard
' USACE-WES-EN-C
' 3909 Halls Ferry Road
' Vicksburg, MS 39180
'
' Sponsor: USACE Mississippi Valley District
'
' Date: 03 Nov. 1997
'
' Modifications:
'
' 10/16/98 - Added number of sample steps to be passed in instead
' of constant
'
'==========================================================
'==========================================================
'
'
' Check to make sure we have at least five arguments
'
if ((Self.IS(List).Not) or (Self.count < 5)) then
return nil
end

flnDirGrid = Self.Get(0)
grdInitialSurface = Self.Get(1)
ftbPointTheme = Self.Get(2)
fldPointTheme = Self.Get(3)
numSampleSteps = Self.Get(4)

'
'thmSurfPts = av.Run("LMK.Water-Flow",{flnDirGrid,grdInitialSurface,
' ftbPointTheme, fldPointTheme,
' numSampleSteps} )
'
'
```

```
'=========================================================
'
'
' Convert direction grid filename to a source name
'
srcDirGrid = Grid.MakeSrcName(flnDirGrid.AsString)
'
' Check and see if it is valid
'
if (srcDirGrid = nil ) then
MsgBox.Error("LMK.Water-flow script got an invalid Direction Grid
Name","Error")
return nil
end

'
' Convert to Theme and get Grid property
'
thmDirGrid = Theme.Make(srcDirGrid)
grdDirGrid = thmDirGrid.GetGrid

'
' Create a temporary shape file for the water points
'
prjProject = av.GetProject
flnWorkDir = prjProject.GetWorkDir
'flnCWD = FileName.GetCWD
flnWaterPoints = flnWorkDir.MakeTmp("wfp","shp")
ftbWaterPoints = FTab.MakeNew(flnWaterPoints,Point)

'
' Add fields
'
lstFields = List.Make
lstFields.Add(Field.Make("Stage",#FIELD_FLOAT,16,2))
ftbWaterPoints.AddFields(lstFields)
fldShapeField = ftbWaterPoints.FindField("Shape")
fldStageField = ftbWaterPoints.FindField("Stage")

'
' Copy all the field data from the initial stage point theme
'
fldPointThemeShapeField = ftbPointTheme.FindField("Shape")

for each rec in ftbPointTheme
shpCurrentShape = ftbPointTheme.ReturnValue( fldPointThemeShapeField, rec)
numCurrentStage = ftbPointTheme.ReturnValueNumber( fldPointTheme,rec)

xyPoint = shpCurrentShape.ReturnCenter
recWaterPoints = ftbWaterPoints.AddRecord
```

```
ftbWaterPoints.SetValueNumber( fldStageField, recWaterPoints,
numCurrentStage)
ftbWaterPoints.SetValue( fldShapeField, recWaterPoints, xyPoint)
end

'
'
' Get Grid characteristics
'
numCellSize = grdInitialSurface.GetCellSize
rectGridExtent = grdInitialSurface.GetExtent
numRows = grdInitialSurface.GetNumRowsandCols.Get(0)
numCols = grdInitialSurface.GetNumRowsandCols.Get(1)

thePrj = Prj.MakeNull

theX = rectGridExtent.GetLeft
theY = rectGridExtent.GetTop

'==========================================================
'
'
' Begin sampling loop
'
numXinc = (rectGridExtent.GetRight - rectGridExtent.GetLeft ).abs /
numSampleSteps
numYinc = (rectGridExtent.GetTop - rectGridExtent.GetBottom ).abs /
numSampleSteps

av.ShowMsg("Sampling water flow routes...")
for each ystep in 1..numSampleSteps
doMore = av.SetStatus( (ystep/numSampleSteps) * 100 )

for each xstep in 1..numSampleSteps

ptX = theX + (numXinc * (xstep - 1)) + ((numXinc / 2.0) * ((ystep -1) mod 2))
ptY = theY - (numYinc * (ystep - 1))

p = Point.Make(ptX,ptY)
theCellValue = grdDirGrid.CellValue(p,thePrj)

lstPointList = List.Make
lstPointList.Add(p)

for each n in 1..500

if ( theCellValue.IsNull), then
' MsgBox.Info("Null point value","")
break
end
```

```
if ( theCellValue = 1 ), then
ptX = ptX + numCellSize

elseif ( theCellValue = 2 ), then
ptX = ptX + numCellSize
ptY = ptY - numCellSize

elseif ( theCellValue = 3 ), then
ptY = ptY – numCellSize

elseif ( theCellValue = 4 ), then
ptX = ptX - numCellSize
ptY = ptY – numCellSize

elseif ( theCellValue = 5 ), then
ptX = ptX – numCellSize

elseif ( theCellValue = 6 ), then
ptX = ptX - numCellSize
ptY = ptY + numCellSize

elseif ( theCellValue = 8 ), then
ptX = ptX + numCellSize
ptY = ptY + numCellSize

elseif ( theCellValue = 7 ), then
ptY = ptY + numCellSize

elseif ( theCellValue = 0 ), then
' Get the water surface elevation value
numWaterElev = grdInitialSurface.CellValue(p,thePrj)

' write out all the points to the shape file
for each mylstpt in lstPointList
recWaterPoints = ftbWaterPoints.AddRecord
ftbWaterPoints.SetValueNumber(
fldStageField, recWaterPoints, numWaterElev)
ftbWaterPoints.SetValue( fldShapeField, recWaterPoints, mylstpt)
end

' Don't forget this!
break
end

p = Point.Make(ptX, ptY)
theCellValue = grdDirGrid.CellValue(p,thePrj)
lstPointList.Add(p)

end ' each n loop
```

```
lstPointList.Empty
end
end

av.ClearMsg

thmWaterPoints = Ftheme.Make(ftbWaterPoints)

return thmWaterPoints
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
July 2004

2. REPORT TYPE
Final report

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE
Flood Event Assessment Tool (FEAT) User's Manual and Technical Documentation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jerrell R. Ballard, Jr., Margaret Rose Kress

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Environmental Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road, Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER
ERDC/EL SR-04-3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Engineer District, Vicksburg
4155 E. Clay Street
Vicksburg, MS 39183

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Flood Event Assessment Tool (FEAT) is a prototype geospatial modeling tool that uses river and stream gage data, landscape digital elevation models (DEMs), main and secondary channel centerlines or cross sections to generate a geospatial-based flood surface. The primary objective in developing FEAT was to improve the accuracy and decrease the time and effort required to determine dynamic flood surfaces and provide information for decision-making. This guide provides an overview of FEAT, information about the algorithms, software and hardware requirements, installation procedures and operation, user input requirements, and examples of model output products.

15. SUBJECT TERMS
Digital elevation models
Flood event assessment
Geospatial flooding
Geospatial modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED		62	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18