Development and Evaluation of the Profile Synthesis Method for Approximate Floodplain Redelineation

Thomas A. Dickerson

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

Master of Science

in

Civil Engineering

Dr. Randel L. Dymond
Dr. David F. Kibler
Dr. Kathleen Hancock

November 15, 2007
Blacksburg, Virginia, USA

Keywords: Floodplain Mapping, Map Modernization, GIS.

Development and Evaluation of the Profile Synthesis Method for Approximate Floodplain Redelineation

Thomas A. Dickerson

ABSTRACT

In the United States, the floodplain maps used in the administration of the National Flood Insurance Program are created and maintained by the Federal Emergency Management Agency. Currently, a nationwide map modernization program is underway to convert the existing paper floodplain maps into a digital format, while continuing to improve the maps and expand the scope of the studies. The flood zones depicted on these maps are developed through engineering studies, using a variety of accepted methods to model and predict flood-prone areas. These methods are classified as detailed, limited detailed, or approximate, corresponding to varying levels of expense and accuracy. Current flood map revision activities across the nation typically consist of developing new hydraulic models, or reusing existing hydraulic model results in conjunction with new, more detailed LiDAR terrain models.

This research develops a profile synthesis method for redelineation of approximate flood boundaries, and evaluates the method's performance and usability. The profile synthesis method is shown to perform reliably on simple floodplain geometry, recreating a water surface profile based only on its floodplain boundaries. When applied to a real-world floodplain studied in a previous flood insurance study, the profile synthesis method is shown to perform adequately, with results comparable to an approximate hydraulic model developed in HEC-RAS. Methods similar to this profile synthesis method for reuse of existing approximate zone boundaries have not been widely documented or evaluated; nevertheless, methods such as this are believed to be common in the revision of approximate zone flood boundaries. As such, this work explores concepts which will be of interest to individuals actively involved in flood map revision and modernization.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1: Introduction

1.1 Flood Maps and the National Flood Insurance Program

The purpose of the National Flood Insurance Program is to share the risks and costs of flood damage, while encouraging actions which will lessen future flood risk to property. The National Flood Insurance Program (NFIP) originates from acts of Congress in 1968 and 1973, with additional revision in 1994. The results of these acts, and their amendments, are documented in the U.S. Code, Title 42, Chapter 50 (42 U.S.C. 4001 et. seq), and implemented in various parts of the Code of Federal Regulations, Title 44. The U.S. Code and the Regulations describe the scope of the National Flood Insurance Program, the requirements which shall apply to mortgage lenders, the requirements which apply to local governments, and the general methods for administering the program.

Currently, the Federal Emergency Management Agency (FEMA) manages the National Flood Insurance Program (NFIP). Part of its role in managing the program is to create and maintain floodplain maps and reports necessary for the assessment of flood risk. These maps and reports are created as a result of engineering studies on a community-by-community basis. A Flood Insurance Study (FIS) report contains water surface elevation profile graphs, data tables, and explanatory text. The Flood Insurance Rate Map (FIRM) depicts floodplain and floodway boundaries, cross-section locations, base flood elevations (BFEs), and basic planimetric data such as roads and political boundaries. These documents are used by insurance agents and mortgage lenders to assess flood risk, as well as by planners and floodplain managers to guide development and manage natural resources.

Flood maps depict a variety of flood zones, describing the expected likelihood of a flood event of a given severity, as well as the level of accuracy with which the flood zone has been studied. Approximate flood hazard studies delineate floodplain boundaries for the 1% annual chance flood event, but do not establish base flood elevations or flood depths (FEMA, 2003). These flood hazard areas are shown on flood insurance rate maps as Zone "A." Alternatively, detailed flood hazard studies use hydraulic models, such as HEC-RAS, to determine floodplain boundaries, floodway limits, *and* base flood elevations for the 1% annual chance flood; these

areas are shown as Zone "AE" on flood insurance rate maps. In general, studies to delineate approximate zones "A" are performed in more rural areas, where development pressure is minimal, or where it is financially unfeasible to perform a detailed study, but floodplains can be reasonably estimated. Approximate zones "A" may later be studied using detailed methods, and can be converted to zone "AE." Conversely, if the hydraulic conditions in a zone "AE" area change due to natural or anthropogenic causes, that area may be converted back to an approximate zone "A," in recognition that the engineering model is no longer valid.

Refer to Appendix A for an example of a portion of a flood insurance rate map.

1.2 Flood Map Modernization

At the time of the first major surge of flood map creation, in the mid to late 1970's, the cartographic techniques used to create maps for publication were primarily analog. Flood insurance rate maps were produced using procedures involving physical media and drafting. Since the mid to late 1990's, computerized Geographic Information Systems (GIS) have revolutionized the cartographic process, making it much easier to produce maps with even more detail and consistency than with analog processes. It is now common for maps and related decision-making to exist entirely in a computer, making physical printed media less essential. Regardless of whether maps are printed or not, the maintenance of digital data in a GIS makes it much easier to revise and republish floodplain mapping information.

Shortly after GIS revolutionized the cartographic world, LiDAR began to revolutionize the way in which topographic data is collected. LiDAR, which stands for Light Detection and Ranging, is essentially a system in which a laser emits a characteristic pulse of light, and a sensor observes the reflection and return of this pulse after striking a distant surface. For each pulse of light, the sensor calculates the time of return, and the associated distance. Airborne LiDAR equipment can collect massive quantities of these spot elevations. Post-processing these spot elevations to produce a bare-earth terrain model involves removing extraneous points from treetops and buildings, as well as the generation of elevation contours or other efficient representations of the terrain. LiDAR-based terrain models can be less expensive and more detailed than photogrammetry-based models, depending on the specifics of the project.

Recognizing these technological improvements, FEMA developed a Map Modernization plan which would include converting the nation's paper flood maps into a digital GIS format. Digital Flood Insurance Rate Maps, or DFIRMS, will replace the older paper rate maps. In addition to the format conversion, updates or restudies are also being incorporated into the new flood maps, as much as funding allows. Work on the program is being carried out by a variety of partners, including other federal agencies, state and local agencies, as well as private and public sector contractors to these agencies (FEMA, 2006). FEMA has developed a fresh set of engineering and mapping guidelines, built new data models, and hosts a variety of web-based applications to ensure consistency among the various partners working on this program.

The concept of a "redelineation" study also gained prominence during the Map Modernization program, based on the increasing availability of high-resolution terrain models from LiDAR data. Redelineation studies do not alter the effective water surface profiles; rather, the effective profiles from a previous study are spatially intersected with a new terrain model to produce the revised floodplain boundary shapes. Floodplain redelineation yields floodplain boundaries that align correctly on a new terrain model.

Chapter 2: Problem Definition

2.1 Problem Statement

A combination of methods is often employed to update a set of floodplain maps for a municipality, depending on the availability of data, the available funding, and the desired level of accuracy. Some map modernization consists entirely of digital conversion: georeferencing and digitizing the effective maps to produce a DFIRM. When new terrain data is available, a redelineation approach may be used to improve the shape of the floodplain boundaries, without modifying the underlying engineering models. Finally, when the existing hydrologic and/or hydraulic modeling is inadequate, a new study may be undertaken. Depending on the type of flood zone being updated, different methods may also be appropriate.

This work focuses on the methods used to update approximate flood zones, areas where no water surface profiles or base flood elevations are published. A variety of methods have been used in these areas, including "artful" refitting of existing boundaries, workmap-based redelineation methods, normal depth calculations, and depth-frequency regression curves. In these zones, the most rigorous approach would be to discard the existing approximate flood boundaries, construct new hydrologic and hydraulic models, and perform new hydraulic calculations to establish a water surface profile for floodplain delineation. As a less costly alternative to the rigorous approach, this thesis introduces and examines a method which shall be termed "approximate flood boundary redelineation by profile synthesis."

The objectives of this thesis can be summarized as follows:

1) Review traditional methods for zone flood modeling
2) Create and document the "profile synthesis method for approximate flood boundary redelineation"
3) Compare the profile synthesis method with a typical approximate hydraulic modeling method

2.2 Constraints

Historically, terrain data for flood mapping was often limited to USGS topographic maps. Approximate flood boundaries were typically drawn based on normal depth calculations for a limited number of cross-sections taken from topographic maps, or simply based on USGS flood-prone quadrangle maps. With the increasing availability of high-accuracy terrain and imagery data, many modern approximate flood studies are conducted based primarily on remotely-sensed data, with much more detail than the historic studies.

Terrain models suitable for floodplain modeling and mapping are being created from LiDAR datasets with various techniques; this is still an active field of investigation (e.g. Stonestreet, 2000). Such models are increasingly central to detailed flood studies, helping to reduce the need for field survey data. The development of accuracy standards and quality review procedures uniquely suited for such datasets is currently ongoing, with significant attention being paid to the diminished performance of such systems in dense woods and brush. Due to the limits of current airborne LiDAR systems, digital terrain models typically do not include any bathymetric (underwater) relief, and when stream channels are heavily vegetated, the accuracy of the terrain model near the banks of streams is also reduced. For small stream channels with low baseflow, the lack of underwater topography may have a minimal effect on the overall accuracy of the model. During a detailed study, streams and rivers are typically surveyed via traditional methods to supplement the remotely-sensed terrain model in and around the stream.

High-resolution aerial imagery has become increasingly common, and is typically acquired for planning purposes in a locality, or at a state level. For example, the Virginia Base Mapping Program produced 1-meter (or better) orthorectified aerial imagery for the entire state in 2002, and will continue in the future as funding becomes available (VGIN, 2003). High-resolution aerial imagery is important to a flood study as a source of information about floodplain roughness, stream crossings like bridges and culverts, and other urban features near the stream. Aerial imagery also provides insight into the extent of changes in the floodplain since the time of the previous study.

Redelineation projects also rely on data from the effective (historic) flood insurance study, such as the Flood Insurance Study (FIS) report, the Flood Insurance Rate Maps (FIRMs), and other workmaps and modeling data which may be stored in FEMA archives. While some forms of this data will always be available for previously studied communities, the accuracy of this data is often limited by map scale, cartographic generalizations that were necessary to produce useful, legible maps, or engineering assumptions necessary to simplify the modeling process. For example, on a FIRM published at 1"=2000', a lineweight of about 0.4mm, typical of a flood boundary lineweight, scales to a distance of about 30 feet on the ground. When the edges of a floodway and a flood boundary are separated by distances approaching this, issues of cartographic clarity often necessitated that the lines be generalized as though they were coincident. Aside from these generalizations, the process of georeferencing a scanned image of such a map in the GIS environment can introduce additional error.

Limited field data collection may be required for some approximate floodplain studies, if the effects of hydraulic structures in the floodplain are deemed to be significant enough to warrant modeling. This data would usually be limited to culvert diameters, pier widths, or critical elevations, and may be available from the locality's or state's infrastructure databases. Significant field data collection efforts would typically only be undertaken for higher-level studies, such as limited-detailed studies or detailed studies.

In general, flood map modernization projects are constrained either in terms of time, data availability, and/or funding. This thesis shall be similarly constrained to include only data and processes which could fit within the scope of a floodplain redelineation project. Specifically, this includes a high-resolution terrain model, aerial imagery, effective (historic) flood insurance study reports and maps, and little or no field data.

Chapter 3: Literature Review

3.1 Introduction

Much of the peer-reviewed literature in the field of floodplain modeling has focused on advanced modeling methods with theoretical importance but without immediate provision for practical application to the current flood map modernization program. This is understandable, given the near-universal dominance of HEC-RAS's one-dimensional approach to floodplain modeling in the United States, and FEMA's explicit endorsement of this time-tested approach. Nevertheless, some literature focusing on enhancements to the typical FEMA floodmapping framework were found, and a North Carolina (2007) technical paper provided a good example of current practice with regard to approximate zone hydraulic modeling.

Few articles were found discussing or comparing methods for approximate floodplain redelineation, and no critiques of the suitability of redelineation for specific situations were found. This was somewhat surprising, given the increasing use of such methods in practice. The absence of such information was one of the justifications for this research. In this thesis, the literature review was used to investigate the background of floodplain modeling, and gather information related to the proposed profile synthesis methodology.

3.2 Hydraulic Modeling

3.2.1 Hydrologic Engineering Center – River Analysis System (HEC-RAS)

Developed by the U.S. Army Corps of Engineers' Hydrologic Engineering Center (HEC), HEC-RAS is FEMA's de facto standard for modeling of riverine flooding sources (FEMA, 2005). The HEC-RAS program has a long history which can be traced back to a FORTRAN program which, by 1968, had been shared with at least 50 public and private offices (Eichert, 1968). The program has evolved greatly since then, and is now distributed primarily for Microsoft Windows. HEC-RAS is preferred by FEMA because it is freely available and has a large set of capabilities, such as steady and unsteady one-dimensional flow, basic bridge and culvert modeling, and floodway determination routines. HEC-RAS is well-documented: a User's Manual, an Applications Guide, and a Reference Manual are all available for the current version of the program.

To model one-dimensional steady flow, HEC-RAS's computational procedure iteratively solves the energy equation and an energy head loss equation between consecutive cross-sections along a stream, also invoking the momentum equation in certain situations, such as crossing through critical depth (Brunner, 2002). Traditionally, cross-sections were field-surveyed, and models relied on a minimal number representing only major change points along the stream. Now, with less costly field and aerial data collection techniques, it is possible to use many more cross-sections to represent the shape of the floodplain as accurately as possible. Despite improvements in the geometric data, many other parameters in the hydraulic model, such as roughness coefficients, expansion/contraction coefficients, and flow estimates, still require significant engineering judgment.

While HEC-RAS's predecessor HEC-2 was used primarily for detailed floodplain studies in the 1970s and 1980s, technological advances have made it feasible to conduct *approximate* floodplain studies in HEC-RAS. By using cross-sections extracted from digital topographic models, along with rough estimates for hydraulic parameters, a HEC-RAS geometric model can be automatically constructed by a computer with limited human guidance. Such models necessarily gloss over the specifics that would be represented in a detailed model, but can still utilize HEC-RAS for calculation of backwater effects caused by changes in floodplain shape and slope.

3.2.2 Models of Higher Dimensionality

In contrast to the one-dimensional approach taken by HEC-RAS, two-dimensional and hybrid 1-d/2-d models are becoming increasingly popular due to the promise of better accuracy in complex floodplains. While careful implementations of HEC-RAS can account for some side flow and side storage effects, these areas must be explicitly designated in advance by the user. On the other hand, two-dimensional hydraulic models are being used to calculate flood flows moving in multiple directions, without prior knowledge of the flow directions. These models typically perform calculations on a digital terrain model, with user-specified (but often calibrated) surface roughness values. Such models are especially critical in low-lying areas

where flow is controlled by dikes and levees which may be overtopped, or in urban areas with complex barriers to flow.

In Horritt and Bates (2002), HEC-RAS was compared to LISFLOOD-FP and TELEMAC-2D, models of increasing dimensionality and complexity. Specifically, LISFLOOD-FP is a hybrid model, representing channel flow with a 1-D approach, and floodplain flow using a decoupled 2-D approach. TELEMAC-2D is a fully 2-D model, using a finite element / finite difference approach to model flow. The performance of these models was compared on a certain reach of the Severn River in the UK. Depending on the type of calibration data available, the performance of the models varied. Interestingly, HEC-RAS was found to have the most consistent predictive performance, primarily because the selected study area was dominated by channel flow processes, rather than complex floodplain flow processes (Horritt, 2002).

3.3 GIS Applications in Hydraulic Modeling

3.3.1 HEC-GeoRAS

In the past five to ten years, GIS software extensions have been developed to facilitate quicker creation of more complex hydraulic models. The software has generally facilitated the goal of "automated floodplain mapping," although significant human guidance is still required. While a variety of proprietary software has been developed to automate floodplain mapping, HEC-GeoRAS is a publicly available program by the USACE which is representative of the type of functionality such software provides. HEC-GeoRAS is an extension to ESRI's ArcMap GIS software environment, guiding the user through the creation of data layers relevant for HEC-RAS, and then completing an export routine to build a GIS data file compatible for import into HEC-RAS. Additionally, HEC-GeoRAS can be used to post-process the results of a HEC-RAS model by delineating the flood boundary on a terrain model in the GIS (Ackerman, 2005).

This software enables approaches to floodplain modeling that are somewhat different from traditional methods. The use of GIS and digital terrain models allows many more cross-sections to be included in the hydraulic model. Previously, cross-sections were selected strategically to best represent field conditions, with the knowledge that each cross-section would require an

expensive field survey. Now, with a digital terrain model and automated data processing, cross-sections for approximate hydraulic modeling can be cut at denser spacing intervals, reducing the need for generalization or simplification of the floodplain model.

HEC-GeoRAS, or equivalent proprietary software, plays a key role in bridging the GIS and engineering software environments. This linkage has led to much development and application work in "automated floodplain mapping." Map modernization contractors are gradually embracing the new methods, while addressing some of the questions that it raises. For example, North Carolina released a series of issue papers to address technical questions, one of which provided guidelines for "automated approximate studies," including guidelines on modeling and mapping parameters (North Carolina, 2007). The automated approximate studies being conducted did not require any survey data, as they were based entirely on LiDAR terrain models. Constant values were assumed for many hydraulic parameters to reduce time and effort spent on model development, and basic standards for cross-section placement were developed. This approach, while still simplistic, is a significant improvement over previous approximate hydraulic modeling methods.

3.3.2 Non-Hydraulic Perspectives

While hydraulic modeling is typically required for engineering design and regulatory restrictions, there are applications where non-hydraulic (or quasi-hydraulic) methods have been used to estimate flood depth and extent. Such quasi-hydraulic methods may utilize statistical relationships to estimate flood depth as a function of upstream drainage area in a given geographic region. These analyses are very simplistic, but can fill a need for community preparedness planning in areas where detailed modeling has not been performed. Some quasi-hydraulic models have been discussed in the literature; for example, consider the work of Gall, Boruff, and Cutter (Gall, 2007).

3.4 Hydrologic Estimates

Hydraulic models, both approximate and detailed, rely on hydrologic flow estimates. Without a reasonable estimate of the quantity of water expected to arrive at a specific point along the

stream, the geometry of a hydraulic model is meaningless. Estimating peak discharges for ungaged streams is a non-trivial activity, with a high level of uncertainty. Methods to develop peak flow estimates include regional regression analyses and rainfall-runoff models. FEMA GSFHMP Appendix C.1.1.3 directs study contractors to utilize existing government studies where possible, to use USGS regional regression equations, or, in cases where runoff behavior is altered by dam storage or other conditions, to develop rainfall-runoff models with simplified river routing.

3.4.1 USGS Regional Regression Equations

The USGS has developed a computer program, National Flood Frequency (NFF), which incorporates regional regression equations for the entire country (Ries, 2002). The program operates with a windows interface, and essentially guides the user through the necessary input variables required to estimate the peak flowrates for a certain location. The program constrains the input data to accepted ranges for each variable, and prevents the types of user error that could occur due to an incorrectly entered formula. NFF is intended for estimates of peak flow at single locations, and does not integrate with other software. As such, NFF is not well suited for developing large sets of peak flow estimates for a series of points-of-interest in a GIS.

The NFF program is based on the results of many separate statewide regional regression studies performed for or by the USGS. For Virginia, the regression equations used in the NFF program originate from a 1995 USGS Water-Resources Investigations Report by Bisese (Bisese, 1995). This report divides Virginia into eight hydrologic regions with similar characteristics for the purposes of peak discharge estimation. The Toms Creek watershed falls near the edge of the Southern Valley and Ridge (SV) region, whose peak discharge estimating equations are reproduced in Table 1. These equations have a relatively small number of equivalent years of record, as well as a high standard error of prediction, indicating the significant uncertainty associated with the estimating equations.

Table 1. Regional regression equations from Bisese (1995)

Regression Equation	Standard error of prediction (percent)	Equivalent years of record
$Q_{(2)} = 45.7(A)^{0.880}$	45.0	1.7
$Q_{(5)} = 89.5(A)^{0.825}$	43.4	2.6
$Q_{(10)} = 127(A)^{0.800}$	44.2	3.3
$Q_{(25)} = 181(A)^{0.774}$	46.6	4.2
$Q_{(50)} = 228(A)^{0.759}$	49.1	4.7
$Q_{(100)} = 281(A)^{0.745}$	52.0	5.2
$Q_{(200)} = 339(A)^{0.733}$	55.3	5.5
$Q_{(500)} = 425(A)^{0.718}$	60.2	5.7

Note that A is area in square miles; Q is flowrate in cfs for the recurrence interval noted in subscript.

3.4.2 Hydrologic Modeling

Due to the uncertainty associated with statistical approaches to peak discharge estimation, such as regional regression equations, there are often situations which warrant the development of a hydrologic model. Hydrologic models, based on rainfall-runoff relationships, can reflect the attenuating effects of dams and other storage devices, and can be calibrated to match stream gaging records. However, on ungaged streams with no significant storage or attenuating features, there may be little justification to build a hydrologic model, as there may be no way to verify its accuracy.

3.5 Statistical Applications

3.5.1 Curve Fitting and the Moving Average

The use of regression methods to fit a curve to a set of data points is well researched, with new work occurring in the fields such as process control and pattern recognition. Basic regression

and curve-fitting methods are used in nearly all fields of science to understand relationships between variables, and to predict future responses to changes in a variable. A review of related academic literature uncovered no specific discussions or applications of curve-fitting to water surface profiles.

3.5.2 Assessing Results

The selection of a measure by which to compare the results of different hydraulic models is a non-trivial task. In the field of remote sensing and image interpretation, most accuracy measures are based on differences in the classification of image pixels. These types of approaches are two-dimensional by their very nature, and have been used to compare the results of floodplain models; for example, consider Gall (2007) or Horritt (2002).

When comparing the results of one-dimensional hydraulic models, however, a two-dimensional approach only adds unnecessary complexity. Since the only difference between the results of multiple one-dimensional hydraulic models will be the water surface profile, a method to compare the water surface profiles is more appropriate. In their comparison of bridge hydraulics calculations by three different one-dimensional models, the US Army Corps of Engineers compared modeled results with observed results. The three models were assessed primarily based on the average absolute error in predicted water surface elevation at selected points of analysis (Brunner, 1995).

Chapter 4: Methods

4.1 The Profile Synthesis Method

4.1.1 Objective and Scope

The profile synthesis method for approximate flood boundary redelineation is a process which reshapes effective flood boundaries to match a new terrain model. Approximate flood boundary redelineation by profile synthesis emerges conceptually from a desire to recycle and reuse existing data where possible, rather than starting from scratch each time a flood map is modernized. When funding for map modernization is limited, or when the extent of flood zone mapping is generally adequate, then there is little justification to expend a great effort in remodeling the approximate flood zones. This reflects the balance between that which is acceptable for regulatory purposes, and that which would be preferable from an academic perspective.

To redelineate floodplain boundaries in areas previously studied by approximate methods, the fundamental task is to recover or establish a water surface profile for each stream. While there is no published water surface data in these zones, analysis of the effective flood boundaries can lead to an estimate of the original water surface profile used to create the flood boundaries. Profile synthesis is a reverse-engineering of the flood maps: rather than creating flood boundaries based on a water surface profile, the profile synthesis algorithms seek to estimate a water surface profile based on given flood boundaries. Once the approximate water surface profile is synthesized, it is spatially intersected with the newest available terrain model to produce updated flood boundaries.

The profile synthesis method is not based on hydraulic calculations; rather, it aims to simulate the results that a hydraulic model might yield by generating a profile that is reasonable. These types of procedures are appropriately termed “quasi-hydraulics,” where “quasi” indicates resemblance or similarity, rather than truth or actuality. This limitation should result in a significant savings as compared to true hydraulic analyses which require the development of hydrologic and hydraulic models.

The procedure for approximate flood boundary redelineation by profile synthesis, shown in Figure 1, involves multiple data creation and analysis steps. Certain steps in the procedure rely on custom algorithms which, at present, can only be implemented in a scripting or programming language providing access to the contents of the geospatial data. In this research, these algorithms were developed in Visual Basic for Applications, the dominant customization language for ESRI's ArcMap desktop software. The necessary algorithms are discussed in the following sections.

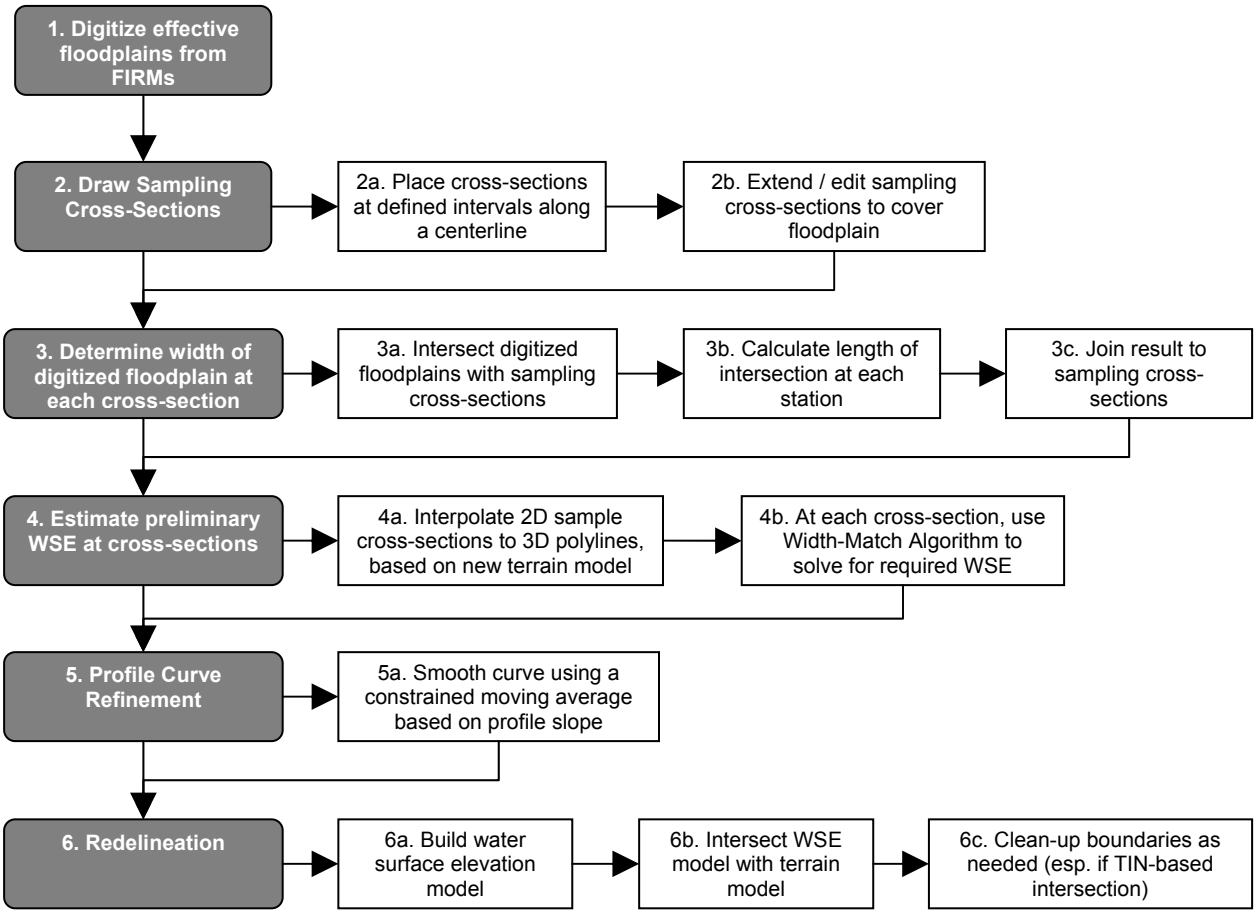


Figure 1. Procedure for approximate flood boundary redelineation by profile synthesis

4.1.2 Algorithms and Processes

4.1.2.1 Placement of Sampling Cross-Sections

Profile synthesis begins by sampling the effective floodplain with a series of cross-sectional lines. At these sample lines, the effective floodplain width and other cross-sectional properties are measured. To facilitate this step, an algorithm for the placement of sampling cross-sections was created. The algorithm creates polylines oriented perpendicular to the centerline at a user-specified interval with stationing values measured along the centerline. The method of stationing used to place these sample lines is not critical; any consistent measurement system is acceptable. Code for an ArcMap VBA UserForm, "Polyline Tics," is included in Appendix B. This macro places polyline "tic" marks at a specified interval along any continuous polyline. After placing these tic marks, they must be manually extended and oriented, if necessary.

When automatically placing sample cross-sections along a centerline, the degree of generalization of the centerline should be considered. It is not necessary for the centerline to follow small meanders in the stream centerline; depending on the level of detail in the stream centerline feature, there may be instances where meanders in the stream can cause an automatically-placed section line to be non-perpendicular to the broader shape of the floodplain. After automatic placement, these section lines should be manually edited to ensure that they are generally perpendicular to the entire floodplain, sometimes resulting in a "bent" cross-section. The subsequent steps in the overall process are not negatively affected by bent cross-section lines, so long as the digitized floodplain width is also measured along the same bent cross-section.

The width of the effective floodplain may be calculated at each sampling cross-section in the GIS environment. For example, in ArcGIS, it is relatively simple to intersect the sampling cross-sections with the digitized effective floodplain area, and calculate the lengths of the resulting line segments.

4.1.2.2 Width-Matching Method for WSE Estimation

After calculating the width of the effective floodplain at each sample cross-section, an algorithm to perform the width-match elevation estimate is necessary. This algorithm solves for the water surface elevation required to match the effective floodplain width, using an iterative solution method. At each cross-section, the water surface elevation yielding a water surface width equal to the digitized value is found by iteratively varying the proposed water surface elevation. This step is where the effective floodplain maps are "reverse-engineered;" the effective floodplain width is used to estimate the water surface elevation, rather than usual floodplain delineation procedures, which do the opposite. Pseudocode for the algorithm is provided in Figure 2, and the full code for the ArcMap VBA UserForm is included in Appendix C.

```
For each sample cross-section:

  Get vertex list (x, y, z)
  Transform vertex list into (x', z) system, where x' is 2-d distance along section
  Find minimum and maximum elevation
  Establish initial solution search window based on minimum and maximum elevation
  Set first proposed solution WSE value to half-way between minimum and maximum elevation

  Do until solution found (or no solution declared):
    For each segment in the cross-section:
      If proposed WSE intersects segment:
        Add intersection point to a list, and note segment orientation (rising or falling)
      End If
    Next segment
    Calculate net water surface width by inspecting list of intersections and orientations
    If current water surface width is within a tolerance of the target width:
      Report proposed WSE as Solution for this section, and exit Do Loop
    Else:
    If we have iterated for a long time with no result then declare "no solution"
      If current water surface width > target width:
        Set the max search window value = current WSE
      Else:
        Set the min search window value = current WSE
      End If
      Propose new WSE at half-way point of new search window
    End If
  Loop

Next sample cross-section
```

Figure 2. Pseudocode for width-matching algorithm to estimate water surface elevation

The algorithm proceeds at each cross-section by first generating a list of the vertices in the cross-section. These vertices are initially stored as x,y,z coordinates in the coordinate system of the source polyline feature class. Before proceeding, this vertex list will be transformed into an x',z coordinate system, where x' is the 2-d distance from the first vertex, measured *along* the cross-section segments. If a cross-section has internal bends and deflections, the coordinate transformation process will "stretch" the cross-section out.

After transforming the vertex list, the minimum and maximum vertex elevations will be extracted for use in establishing the search window. The search window is the range of elevation values in which the solution for the desired cross-sectional width is expected to occur. On the first iteration, the search window is set equal to the full range of elevations present in the cross-section. Each time the search window is updated, a proposed solution value for the water surface elevation (WSE) is set to the mid-point of the elevation range in the search window. If the proposed water surface elevation results in too large of a water surface width, then the search window for the next iteration will be the lower half of the current search window. If the proposed water surface elevation results in too small of a water surface width, then the search window for the next iteration will be the upper half of the current search window. Thus, the iterative solver operates as a type of binary division tree, reducing the size of the search window by half with each iteration. The inherent assumption in this algorithm is that raising the water surface elevation will increase the water surface width for the cross-section; this will be true so long as the cross-section does not include significantly undercut or overhanging features.

The main loop in the algorithm is based on varying the proposed WSE value until a solution is found. Within this main loop, the water surface width calculation uses a nested loop to cycle through the segments in the cross-section and determine the water surface width that results from the proposed water surface elevation. This calculation is done by testing whether each segment intersects with the proposed water surface elevation, and if so, noting the point of intersection and the orientation of the segment relative to the proposed water surface elevation. After the width is determined, it is checked against the target width. If the current width is not sufficiently close to the target width, the search window is narrowed and the main loop repeats with a revised proposed WSE.

An example of the algorithm's solution method is illustrated in Figure 3. In this example, the target width was 46.2408'. The algorithm uses 6 iterations to find a water surface elevation that yields a water surface width within 0.5' of the target width, at which point the algorithm stops and reports the water surface elevation from the final iteration. Note that the first trial is at the mid-point of the total range of elevations in the cross-section, the second trial is at the midpoint of the lower half, and so on by halves until the solution tolerance is met. By decreasing the allowable solution tolerance value, more iterations will be used to find a more precise solution.

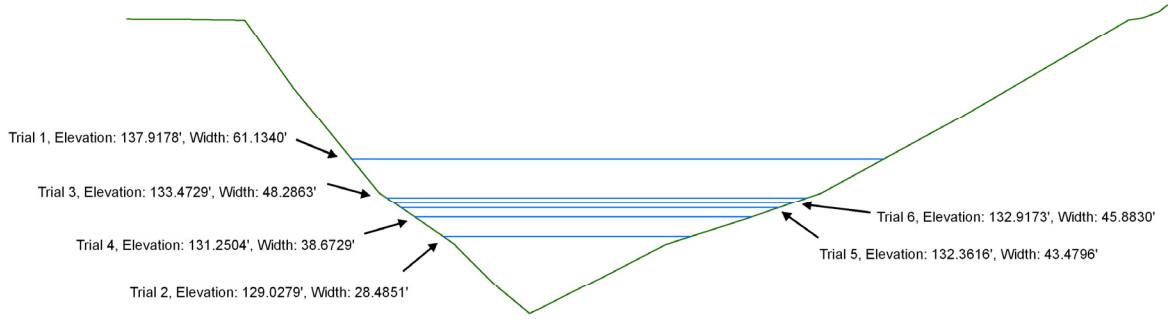


Figure 3. Example of Width-Match Algorithm Solution Process

The width-match algorithm is subject to certain limitations; specifically, the cross-section should not include any overhanging segments. While such cross-sections can possibly occur in reality, it is impossible to represent these conditions on the standard 2.5-dimensional surface models (either TIN or raster) used in current practice, so this limitation is acceptable. In addition to this limitation, some floodplain widths are impossible to achieve on certain complex cross-sectional shapes, and in these cases, the algorithm will yield no solution for that cross-section.

An example of a cross-sectional shape for which a certain water surface width cannot be found is illustrated in Appendix E. In this example, a series of cross-section vertices form a flat surface at an elevation of 1406'. This flat section causes the water surface width to change abruptly when the water level rises above an elevation of 1406'. The search algorithm cannot find a solution for a water surface width in the range of 165.5' to 214.5' due to the abrupt change in water surface width at an elevation of 1406'. While not yet implemented, the search algorithm could be

modified to report the elevation where the abrupt change occurs as an approximation to the solution in this situation.

Failure rates during real applications of the width-match algorithm may vary, but rates between 5-10% were encountered during initial testing. If enough cross-sections are placed along a flooding source, any cross-sections which yielded no solution can be omitted without affecting the overall shape of the profile. Alternatively, one could add a new cross-section just upstream or downstream of the problem cross-section, or use engineering judgment to estimate a water surface elevation for the problem cross-section.

The width-matching algorithm *is* affected by cross-sections which are not perpendicular to the overall floodplain in the vicinity of the floodplain boundary. Although extensive testing has not been performed, the use of cross-sections which are entirely askew to the floodplain could cause differences in estimated water surface elevation on the order of a few feet, depending on the specific conditions in each case. Therefore, if automatic cross-section placement scripts are used, a manual check should be performed to bend or orient the cross-sections to ensure that they are perpendicular to the overall shape of the floodplain in the vicinity of the floodplain boundary.

Finally, the entire width-matching approach requires that the original floodplain boundaries' shape generally conform to the shape of the new terrain model being used for the redelineation project. Areas of conflict can occur if the original floodplain boundaries are digitized from maps that cannot be accurately georeferenced, or if the original floodplain boundaries are based on a historic terrain model with areas that differ significantly from the new terrain model. An example of such a situation is shown in Appendix I.

4.1.2.3 Profile Curve Refinement

The width-matching solution method for estimation of water surface elevation can be used to create a synthesized profile graph along the stream centerline for a flooding source. However, the resulting profile graph should be considered "preliminary," as the profile shape may violate certain underlying rules governing water surfaces, namely: that water surface elevations should always decrease moving downstream. The preliminary profile may violate this rule due to the

fact that the digitized flood boundary was created from a different topographic model than is being used for the redelineation. As such, the profile will typically show some "noise," that is, data values are scattered around the true shape of the profile.

To address this problem, a variety of profile refinement algorithms were considered, as well as criteria upon which to judge the refinement algorithms. However, due to the approximate nature of this redelineation procedure, multiple refinement algorithms may be equally reasonable. Of all the methods considered, those which relied on shifting a fixed profile shape to minimize residuals from the raw data points were found to have the worst results. As a result, the selected method for profile refinement was a type of smoothing filter, based on a moving average of slope values. The full code for the ArcMap VBA UserForm is included in Appendix D.

While many variations on the concept of the moving average exist, they are all generally used to smooth out local fluctuations while maintaining the overall shape of a curve. The algorithm designed for this application is not a traditional moving average, as this algorithm considers the average of slope rather than elevation values. Using the average of slope, rather than of the actual elevation values, enables the algorithm to function on data with an irregular sampling interval. The algorithm draws a smoothed profile curve starting at the upstream (or downstream) point, with each subsequent point calculated based on an average of the slope from the current position to the next *n* data points. The selection of the *n* parameter determines how many points in advance of the current position will be included in the slope average. A smaller *n* value will make the algorithm more sensitive to local fluctuations, while a larger value will smooth out these fluctuations. Or put another way, a smaller *n* value retains more of the shape of the raw data points, while a larger *n* value retains only significant trends.

Equation 1. Moving average *of slope*

$$Y_{i+1} = Y_i + Y_i \left(\frac{\sum_{j=i+1}^{n} \frac{y_j - Y_i}{x_j - x_i}}{n} \right)$$

Where: i and j refer to specific data values

n defines the number of values included in the moving average window

x_i and x_j refer to the ith and jth station values

y_i and y_j refer to the ith and jth profile elevation values

Y_i and Y_j refer to the ith and jth smoothed profile elevation values

When this algorithm is applied to a profile curve, the moving slope average is based on the slope to the next n data points (the term in parenthesis). However, as the algorithm nears the end of the curve, the number of data points available for inclusion eventually becomes less than the specified window size *n*. When this occurs, the moving average is based on a diminishing number of points, until the profile eventually converges exactly on the final data point, except in the case described below.

This profile refinement method, based on a moving average of slope, is further constrained to ensure that the profile elevations always decrease moving downhill (or vice versa). At each step along the profile, if the value resulting from the algorithm violates the downhill flow rule, the value is rejected and the value from the previous point is repeated. As a result, in regions of high fluctuation (or noise), the curve developed by moving downstream may differ from the curve developed by moving upstream. Without any criteria by which to choose one direction over the other, the curves developed by moving in both directions are averaged to produce a single profile curve.

This profile curve refinement method was developed primarily for riverine streams in rural areas, studied only with approximate methods. To apply this method to streams with studied hydraulic structures, such as bridges or culverts, the water surface profiles above and below the structure should be refined separately to ensure that any abrupt profile elevation changes at the structure location are maintained.

4.1.3 Redelineation from Profile Synthesis Results

The results of the profile synthesis method are a set of water surface elevations, with one water surface elevation assigned to each sampling cross-section. This data constitutes the synthesized

water surface profile to be used for floodplain redelineation. The redelineated floodplain boundaries are obtained by intersecting the current terrain model with the water surface from the profile synthesis results. This type of surface intersection calculation is a standard function available in modern GIS software.

4.2 Approximate Hydraulic Modeling in HEC-RAS

4.2.1 Objective and Scope

As described in the literature review, so-called "automated" floodplain mapping methods are becoming popular in the industry. By using computer GIS software to manage data for the hydraulic model, more complex hydraulic models can be constructed than was possible in the past. While the advantages of modern LiDAR terrain models and computerized data management have the potential to improve floodplain model quality, approximate hydraulic modeling still involves significant simplifications. The so-called "automated" hydraulic modeling methods for approximate floodplain studies are often perceived as more accurate than reuse/reshaping methods, due to their increased complexity. However, the number of parameters and variables involved in a hydraulic model leaves significant room for variation in the results.

Approximate hydraulic modeling results *can* be expected to be more realistic in their treatment of backwater effects from obstructions such as road crossing embankments, so long as cross-sections are placed appropriately. Finally, new approximate hydraulic analysis is the only approach to approximate flood boundary delineation that can reliably predict changes in floodplain shape due to changes in the floodplain's hydrology or hydraulics.

The distinction between *approximate* hydraulic modeling and *detailed* hydraulic modeling is found only in the quality of the input data to the model; the same modeling software, HEC-RAS, is commonly used for both approximate and detailed hydraulic modeling. The approximate hydraulic modeling processes discussed in this section are characterized by their reliance on remotely-sensed terrain data to generate cross-sections, assumed parameter values, and no consideration of hydraulic structures.

4.2.2 Processes

4.2.2.1 Data Creation with HEC-GeoRAS

HEC-GeoRAS guides the user through the process of creating the necessary GIS data layers for export to HEC-RAS. These layers include features such as stream centerlines, cross-section locations, and overbank flow paths. Other similar software packages may attempt to automate some of this data creation, extracting stream centerlines from a terrain model, using automatic cross-section placement schemes, and so on.

HEC-GeoRAS does not assist the user in developing the hydrologic flow estimates necessary for the model. The approximate hydraulic model requires discharge estimates at one or more locations along each studied stream. In large map modernization projects, a hydrologic model is developed prior to the hydraulic model, perhaps as part of a broader hydrologic modeling effort for an entire county.

4.2.2.2 Modeling in HEC-RAS

HEC-RAS can import the geometric model of a floodplain from a specific GIS data format, a format which HEC-GeoRAS exports from the ArcMap GIS environment. After importing the geometric model, various parameters need to be entered (or confirmed) prior to running the model. First, properties of the geometric model, such as the Manning's n values, the downstream reach lengths, and the expansion/contraction coefficients should be confirmed. Second, the flowrate estimates from the hydrologic model should be entered, and appropriate boundary conditions (normal depth, known depth, etc.) should be selected.

One of HEC-RAS's features which is relevant to approximate hydraulic modeling is cross-section interpolation. The capability to generate linearly interpolated cross-sections is necessary so that HEC-RAS can calculate water surface data at solution points spaced much more frequently than cross-sections. HEC-RAS generates warning messages when more cross-sections are advisable, for reasons such as rapid changes in velocity head, energy losses, or conveyance. Adding linearly interpolated cross-sections allows HEC-RAS to perform more

precise calculations, and to the extent that the interpolated cross-sections closely match reality, the model results will be more accurate.

4.2.3 Redelineation from Approximate Hydraulic Model Results

Once a model is run and the water surface profile is accepted as complete, HEC-RAS can export the profile back to a GIS format file which can be imported by HEC-GeoRAS. The exported data contains water surface elevations at each cross-section in the model. In the GIS environment, HEC-GeoRAS can be used to process the results, and generate a water surface model for use with typical floodplain delineation techniques. Typically, the water surface model is intersected with a terrain model to yield updated floodplain boundaries.

Chapter 5: Case Applications

Two case applications of the profile synthesis method are used to assess the validity of the profile synthesis method for approximate flood boundary redelineation. In the first case, a synthetic example, the basic capability of the method is tested. In the second case, a real-world example from Toms Creek, the results of the profile synthesis method are compared with the results obtained by approximate hydraulic modeling. These tests provide a basis to evaluate the accuracy of the profile synthesis method. Based on the results of the tests, recommendations regarding the usage of the profile synthesis method in various situations are made. While accuracy is an important objective, other considerations, such as cost, sensitivity, and required expertise may also factor into the selection of a particular method. The case applications determine to what extent a simpler, cheaper method (profile synthesis) can be used in lieu of a significantly more intensive and costly method (approximate hydraulic modeling).

5.1 Synthetic Example

To demonstrate the basic capability of the width-match method for profile synthesis, a synthetic example was created with simple floodplain geometry (shown in Figure 4). An arbitrary water surface elevation profile was created to test the ability of the width-match algorithm to recreate the profile. This arbitrary water surface profile was intersected with the synthetic terrain model to delineate the initial floodplain boundary. This initial floodplain boundary is analogous to the floodplain boundaries digitized from an effective flood map or FIRM. Cross-sectional lines are used to sample the floodplain width, and the width-match algorithm is used to determine the water surface elevation necessary to achieve the specified floodplain width. The solutions found for the water surface elevation should ideally recreate the arbitrary water surface profile used to delineate the initial floodplain boundary.

The first test of the width-match method for profile synthesis used cross-sectional lines placed nearly perpendicular to the primary flow direction of the floodplain. A second test with cross-sectional lines skewed 15 degrees from perpendicular was also performed, to check the sensitivity of the process to minor errors in cross-section placement. The results of these tests are described in Chapter 6.

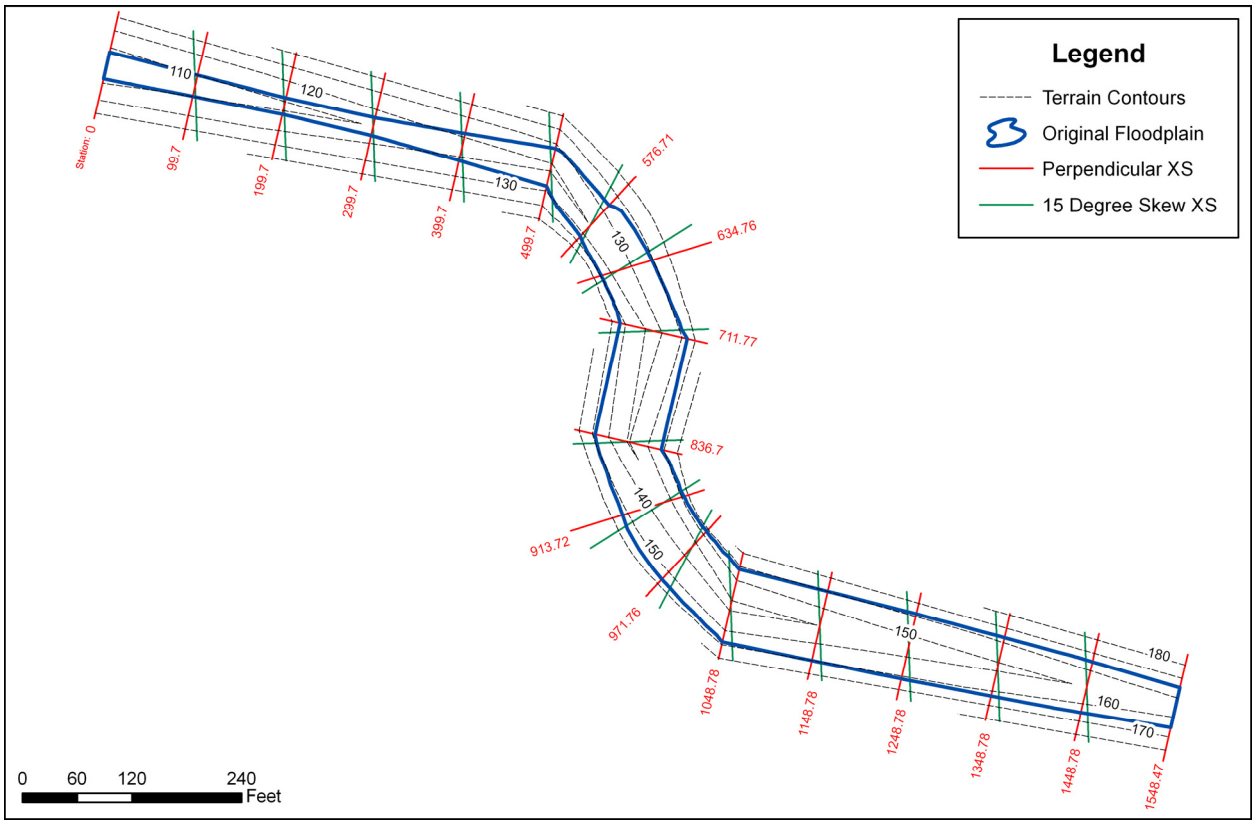


Figure 4. Synthetic example testing the capabilities of the width-match method

5.2 Toms Creek in Blacksburg, Virginia

While the synthetic example tests the basic principles of the width-matching algorithm used in profile synthesis, a second example application using realistic conditions is necessary to demonstrate the usability of the profile synthesis method. In this example, the profile synthesis method is compared against the results of an approximate hydraulic model. For realism, a study area was selected on Toms Creek in Blacksburg, Virginia. The Town of Blacksburg's effective FIRMs date to 1980, and show the effective floodplain areas on Toms Creek. These FIRMs do not show any portions of the floodplain which fall outside of the town limits, and since Toms Creek functions as the town boundary along part of its length, some of the floodplain areas are missing. To fill in this gap, it was necessary to recover a historic workmap used in the process of creating the effective FIRMs. As shown in Figure 5, the effective flood insurance study divided the creek into a detailed zone AE, an approximate zone A, and an approximate zone B on the Shadow Lakes Triburary (refer to Appendix H for a map with XS stationing). The flood study established a water surface profile for the detailed zone, with published profile sheets and base

flood elevations. The approximate zones do not have published water surface data. Therefore, in the detailed zone on Toms Creek, the results of the profile synthesis method and the approximate hydraulic model can also be compared to the water surface profile published in the effective flood insurance study.

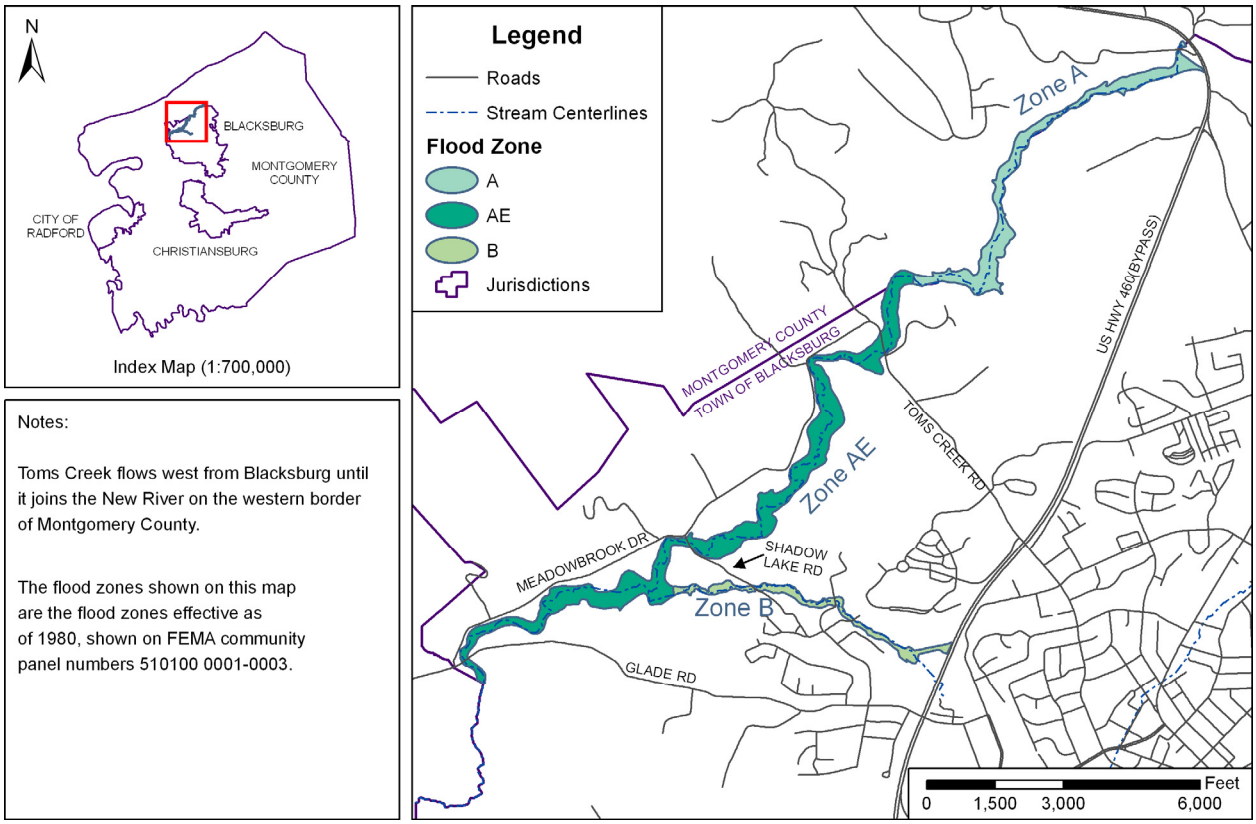


Figure 5. Effective (1980) Flood Zones on Toms Creek in Blacksburg, Virginia

For all modeling methods, a Triangulated Irregular Network (TIN) built from LiDAR-based contour data was used as the base terrain model for interpolating the cross-section geometry. A separate, coarser raster elevation model from the USGS Seamless Data Distribution website was also referenced for general hydrologic estimates. Aerial imagery from the 2002 VBMP fly-over served as a reference for ground conditions in the study area.

5.2.1 Approximate Hydraulic Model

The approximate hydraulic model for Toms Creek is built from a collection of data inputs: floodplain geometry as represented by cross-sections, flowrate estimates from a hydrologic

model, boundary conditions for flow depth, and a variety of engineering parameters such as roughness and expansion/contraction coefficients, most of which are assumed in accordance with common practice. HEC-GeoRAS was used to prepare GIS data for modeling in HEC-RAS.

In building the approximate hydraulic model, the same general approach documented in North Carolina Issue Paper 35 (2007) was used, with some modifications to the cross-section spacing. For this model, cross-sections were placed in the effective detailed study zone at the same locations as the lettered and mapping cross-sections used in an ongoing flood boundary redelineation project for FEMA. This placement is intended to allow for comparison between the results of the effective flood study and the approximate hydraulic model. In the remaining approximate zones, cross-sections were placed at a fairly dense spacing, with an average interval of 200 feet.

5.2.1.1 Peak Flow Estimates

The existing flood insurance study report for Toms Creek in Blacksburg, Virginia includes peak discharge values for three points along the creek: the downstream study limit, the crossing of Shadow Lake Road, and the upstream study limit. In addition to the 100-yr peak discharge in cubic feet per second, the report also includes the drainage area contributing flow to each point of interest. To permit direct comparison of the effective study and the new approximate hydraulic model, it would be ideal to reuse the effective hydrologic model. However, to assess the reasonableness of the effective hydrologic model, it was compared to the estimates that would be obtained by using the most recent USGS regional regression equations.

Table 2 compares the hydrologic estimates from the effective flood insurance study with those obtained by applying the most recent USGS regional regression equations to a raster-based watershed model (refer to Appendix F). The discharges used in the flood study are significantly higher; about three standard errors higher than the USGS regression equation results. This difference raises some questions as to the accuracy of one or both of the methods. The effective flood study report states that the estimates were based on a log-Pearson Type III method regional study of gages in southwestern Virginia and northwestern North Carolina, citing an unpublished Army Corps of Engineers report. While this study has not been located, it is certainly a different,

older study than the 1995 USGS study report by Bisese. The 1995 Bisese study's use of hydrologic regions is a probable source of the drastic difference between the new and old estimates. In this study, the Toms Creek watershed falls near the edge of the Southern Valley and Ridge hydrologic regions. If it were located just over the eastern continental divide in the Central Valley and Ridge region, the use of that region's estimating equations would yield significantly higher values, perhaps agreeing more closely to the effective flood study report. Or, stated differently, the effective flood study report's hydrologic estimates likely did not make the fine regional distinctions that the 1995 USGS study by Bisese makes, resulting in discharge estimates that are higher, due to the inclusion of gage data from across many heterogeneous regions.

Table 2. Comparison of Effective Study Discharges and Regional Regression Estimates

	Drainage Area (sq. mi.)		Q100 (cfs)		
Point of Interest	Effective FIS	Delineated from Elevation Raster (for comparison)	Effective FIS	1995 USGS, based on FIS Drainage Area	1995 USGS, + 3 standard errors
Lower Study Limit	14.3	13.7	5200	2040	5220
Shadow Lake Rd.	11.4	10.7	4500	1720	4410
Upper Study Limit	8.2	7.5	3700	1350	3450

In addition to the differences in flow estimates, there were slight differences in the delineated watershed area. The effective flood insurance study report consistently reported slightly higher watershed areas than the results obtained by delineation algorithms on a digital terrain model. The cause of this discrepancy is unknown, but is not nearly large enough to account for the discharge difference.

To enable better comparison with the effective flood study, discharge estimates for the approximate detailed study are based on the 1995 regional USGS equation plus three standard errors, a compromise which yields values similar to the effective flood study. If comparison with existing models were not the objective, more research into the hydrologic estimates would be warranted to justify the use of discharge values more in line with the current USGS regional regression equations. Using the 1995 regional USGS equation plus three standard errors, flow estimates were made at intermediate locations throughout the model based on upstream drainage area. The results of this estimation are shown in Table 3.

Table 3. Flow Estimates used in Approximate Hydraulic Model

Toms Creek			
XS Station	Area	USGS Q100	USGS Q100 + 3 std err
	sq. mi.	cfs	cfs
32113.559	3.2	670	1710
31712.52	4.9	920	2350
30170.971	5.2	960	2450
27985.902	5.6	1010	2580
26254.857	6.2	1100	2810
23525.719	7.2	1230	3140
22080.137	7.5	1270	3240
19556.285	8	1320	3380
18654.424	9	1450	3700
17462.203	9.5	1510	3860
8928.916	10.7	1640	4200
6565.7363	12.1	1800	4610
3036.4551	12.9	1890	4830
1873.942	13.2	1920	4910
739.1325	13.7	1970	5040
Shadow Lakes Tributary			
XS Station	DA (sq. mi.)	USGS Q100	USGS Q100 + 3 std err
	sq. mi.	cfs	cfs
8319.3516	0.3	120	300
5277.0825	0.6	200	500
163.5135	0.9	260	670

5.2.1.2 Hydraulic Model Construction

The hydraulic model for Toms Creek was built using HEC-GeoRAS in ESRI's ArcMap. Using HEC-GeoRAS, cross-sections were drawn to mimic the effective flood study cross-sections where applicable, and at a relatively dense spacing elsewhere. In the detailed study zone on Toms Creek, the average cross-section spacing was about 500 feet, while in the approximate zones on Toms Creek and Shadow Lakes Tributary, the average cross-section spacing was about 200 feet. The cross-section vertex elevations were interpolated based on the terrain TIN, which provides excellent floodplain shape with limited channel definition. This data was exported into HEC-RAS version 3.1.3, where hydrologic and hydraulic parameters were entered manually.

Hydraulic parameters added in HEC-RAS include Manning's n, expansion and contraction coefficients, and boundary conditions. Following the guidance in North Carolina Issue Paper 35, Manning's n values of 0.1 were used throughout, and values of 0.3 and 0.5 were used throughout for the contraction and expansion coefficients, respectively. The boundary conditions for the furthest upstream and downstream limits were assumed to be normal depth, based on slopes estimated from the terrain model. To help refine the modeled water surface profiles, functionality within HEC-RAS was used to interpolate supplemental cross-sections. These cross-sections are computed based on the assumption of linear change between known cross-sections. A plan view of the geometric model is shown in Figure 6.

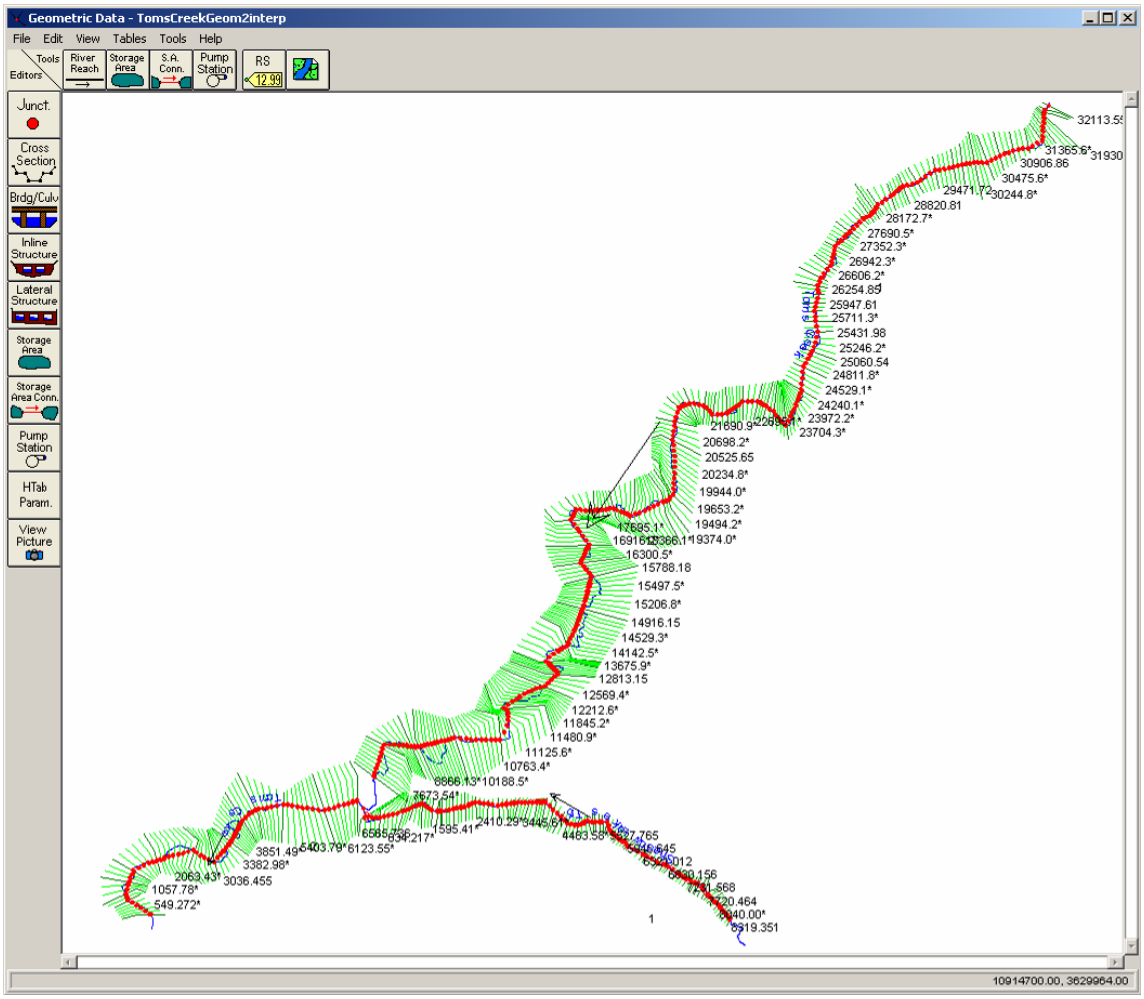


Figure 6. Plan view of HEC-RAS geometric model (green XS marked [*] are interpolated)

5.2.2 Profile Synthesis Method

The results of the profile synthesis method for approximate flood boundary redelineation will be compared with the approximate hydraulic model for the Toms Creek study area. To compare the profile synthesis method with the approximate hydraulic model as closely as possible, the same cross-sections were used in each (although a few were extended slightly). For profile synthesis, the procedures documented in the procedure diagram of Figure 1 were followed for the Toms Creek study area. Starting by digitizing the effective floodplains from available flood insurance rate mapping, the width-match algorithm was used to estimate points on the water surface profile. A smoothed line was fitted through these points, and this water surface was used for redelineation of the flood boundaries.

The results of the profile synthesis method for Toms Creek, as well as comparisons to the approximate hydraulic model results and effective flood insurance study profiles, are documented in Chapter 6.

Chapter 6: Results and Conclusions

6.1 Discussion of Results

The results of the two case applications, a synthetic example and a real-world example, are used to assess the validity of the profile synthesis method for approximate flood boundary redelineation. The primary results of both the profile synthesis and approximate hydraulic analysis methods are the water surface elevations at the cross-sections, forming a water surface profile along the length of the stream. This profile is similar to the water surface profile shown in flood insurance study reports, commonly used to estimate water surface elevations at intermediate locations along the stream. In addition to viewing results as a profile, the water surface elevations can be used to delineate floodplain boundaries on the terrain model.

As noted in the literature review, various methods have been used to compare the results of floodplain models, with the most appropriate method depending on the nature of the model. Comparison of the results of the one-dimensional methods considered here can be performed on the water surface profile by calculating the area between the two water surface profiles, in addition to other descriptive statistics. Since the primary results of the two methodologies are water surface elevations on a profile graph, comparing the results on a profile curve is the most direct way to assess the differences.

6.1.1 Synthetic Example

The results of the synthetic example tests are reported in Table 4. The arbitrary water surface profile was recreated accurately by the profile synthesis method using perpendicular cross-sectional lines; the average error across all the points of analysis was 0.066 feet. The use of cross-sectional lines placed at a 15 degree skew angle yielded lower overall accuracy, with an average error of 0.234 feet. However, as illustrated in Figure 7, higher estimation errors from non-perpendicular cross-sections were found only in the vicinity of the bends in the stream. This is to be expected, as the skew cross-sections intersect the floodplain boundaries at even more disparate elevations in the bends than in the straight sections.

In both tests of the synthetic example, the width-match algorithm's solution search tolerance is set to stop iteration and report a solution when the iteratively varied water surface elevation yields a floodplain width within 0.5 feet of the target. This width-based process control can potentially result in some variation in water surface elevation estimation error depending on the slope of the floodplain walls. Setting the tolerance value lower could reduce that error in ideal conditions, but on complex cross-sections, setting the tolerance too low could result in a failure to find a solution in some cases.

It is important to recognize that in this synthetic example, the floodplain boundaries used for width take-offs and subsequent profile synthesis were in precise geometric alignment with the terrain model used in the original delineation. As a result, the methodology performs very well, and the results of the width-matching algorithm do not need any smoothing or curve-fitting to produce an adequate water surface profile. In real-world applications of the profile synthesis method, the floodplain boundaries will generally not align with the terrain model being used for redelineation for two reasons. First, the floodplain boundaries are being extracted from a paper map subject to cartographic necessities such as generalization and line thickening. Second, the terrain model being used for redelineation is generally not the same as the terrain model used in the original study. As a result, the results of the width-matching algorithm will tend to require smoothing or filtering.

Table 4. Results of the Synthetic Example

			Profile Synthesis Results			
Stream Station (ft)	Original Water Surface Elevation (ft)	Stream Bed Elevation (ft)	Perpendicular XS Water Surface Elevation Estimate (ft)	Perpendicular XS Estimation Absolute Error (ft)	15 Degree Skew XS Water Surface Elevation Estimate (ft)	15 Degree Skew XS Estimation Absolute Error (ft)
0.00	108.00	100.00	107.97	0.031	107.97	0.031
99.70	110.00	103.48	110.12	0.122	110.00	0.001
199.70	112.00	106.97	112.02	0.020	111.99	0.012
299.70	116.00	110.47	116.04	0.044	115.98	0.018
399.70	122.00	113.96	121.93	0.074	121.98	0.019
499.70	128.00	117.45	128.07	0.075	128.73	0.733
576.71	133.00	120.14	132.92	0.083	133.18	0.177
634.76	139.00	122.16	139.07	0.068	138.84	0.164
711.77	144.00	124.85	143.98	0.022	142.86	1.136
836.70	148.00	129.22	148.07	0.075	148.29	0.288
913.72	153.00	131.90	153.10	0.099	153.87	0.871
971.76	156.00	133.93	155.83	0.171	156.10	0.105
1048.78	158.00	136.62	157.87	0.131	157.62	0.385
1148.78	160.00	140.11	160.03	0.033	160.07	0.070
1248.78	162.00	143.60	161.93	0.069	162.01	0.007
1348.78	163.00	147.10	163.03	0.033	162.91	0.093
1448.78	164.00	150.59	164.00	0.001	164.07	0.066
1548.47	165.00	154.07	164.96	0.041	164.96	0.041
			Mean	0.066		0.234
			Median	0.068		0.082
			Min	0.001		0.001
			Max	0.171		1.136

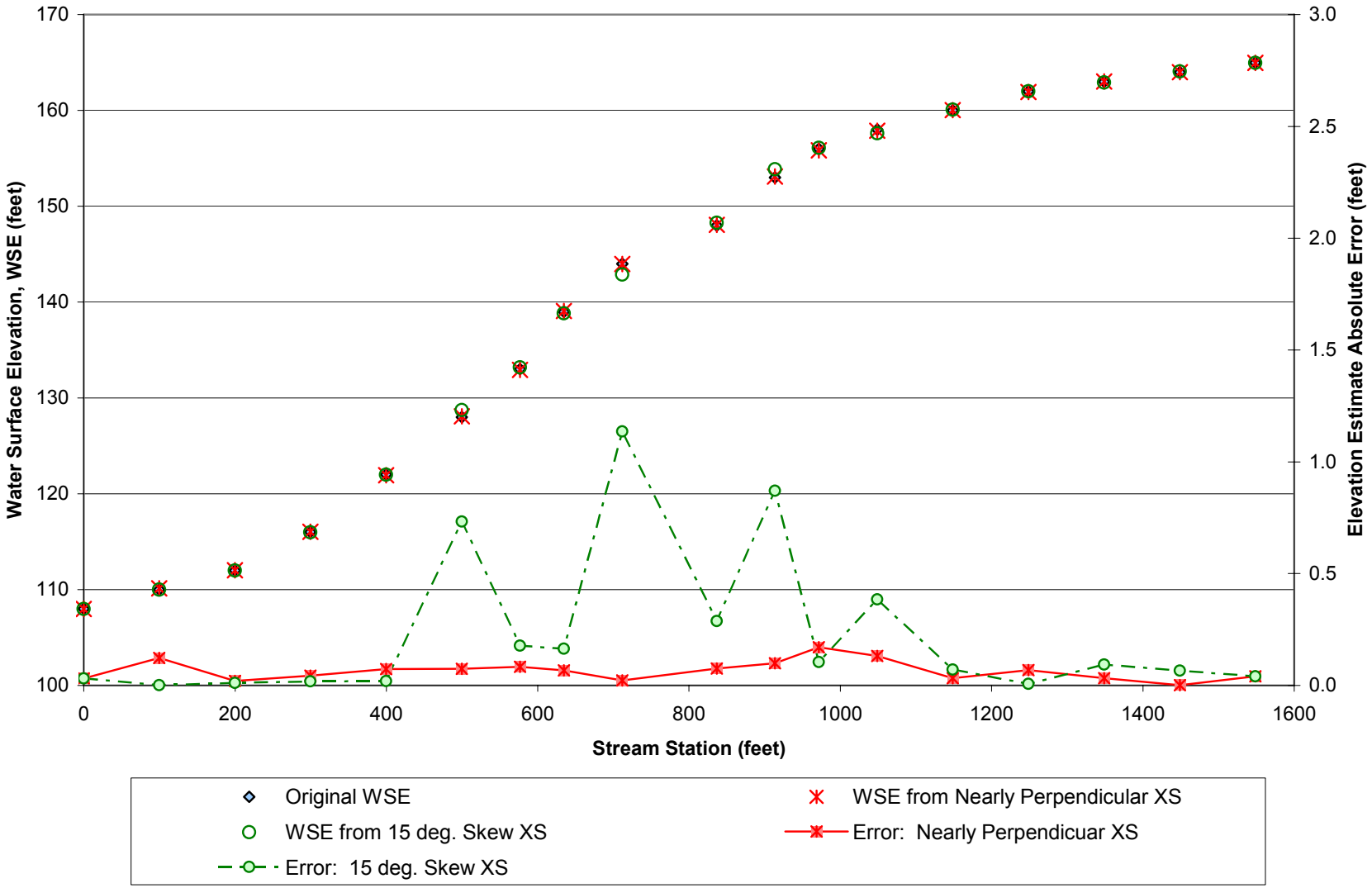


Figure 7. Synthetic Example Water Surface Profile and Estimation Error

6.1.2 Toms Creek in Blacksburg, Virginia

In Figure 8 and Figure 9 the results of the profile synthesis method and the approximate hydraulic model for Toms Creek study area are compared in profile view. Figure 8 shows the approximate hydraulic analysis from HEC-RAS, the profile synthesis method result, as well as points from the effective FIS profile. The minimum cross-section elevation depicted on these profiles was extracted from the terrain model. Since the terrain model does not accurately capture the true stream bed elevation, the minimum cross-section elevation shown on these profiles may be slightly above the true streambed elevation. In Figure 9, for Shadow Lakes Tributary, no FIS data is shown because there are no published water surface elevations in approximate zones.

The profile differences between each method are compared numerically in Table 5. The area between the curves indicates the net difference between a pair of results. The average profile elevation difference is calculated by dividing the area between two curves by the total length of the comparison. For example, the comparison pair "HEC-RAS vs. Profile Synthesis" on Toms Creek is a comparison of the approximate hydraulic model from HEC-RAS with the profile synthesis method. The average profile elevation difference of 2.33' indicates that the HEC-RAS result is 2.33 feet above the profile synthesis result, on average.

The HEC-RAS results are consistently higher than the other methods along most of Toms Creek; there are a variety of possible causes for this outcome. First, the HEC-RAS model was built from a LiDAR-based terrain model that does not represent the channel banks or bathymetry, effectively reducing the available conveyance area of any given cross-section. This effect could be ameliorated by superimposing arbitrary channel geometry at the base of the floodplain, or by refining the LiDAR model through a series of field surveys. Second, the roughness and expansion/contraction coefficients used in the model were based on values recommended from North Carolina Issue Paper 35 (2007). The recommended Manning's n-value of 0.1 would normally correspond to a forested floodplain; lower values would be more realistic for areas of the Toms Creek floodplain covered by open pasture. Third, the omission of bridge and culvert openings exaggerates the backwater effects upstream of such features. So, while the HEC-RAS model could be improved with more detailed input data, this would be beyond the scope of most

approximate hydraulic models. The model developed using the default input data is representative of the level of effort that can be expected for approximate floodplain modeling.

The profile synthesis results appear to agree reasonably well with the effective FIS data in the detailed study zone on Toms Creek. The fact that the profile synthesis method comes closer to replicating the effective FIS profile than the HEC-RAS method indicates that the shapes of the new and old terrain models are fairly similar.

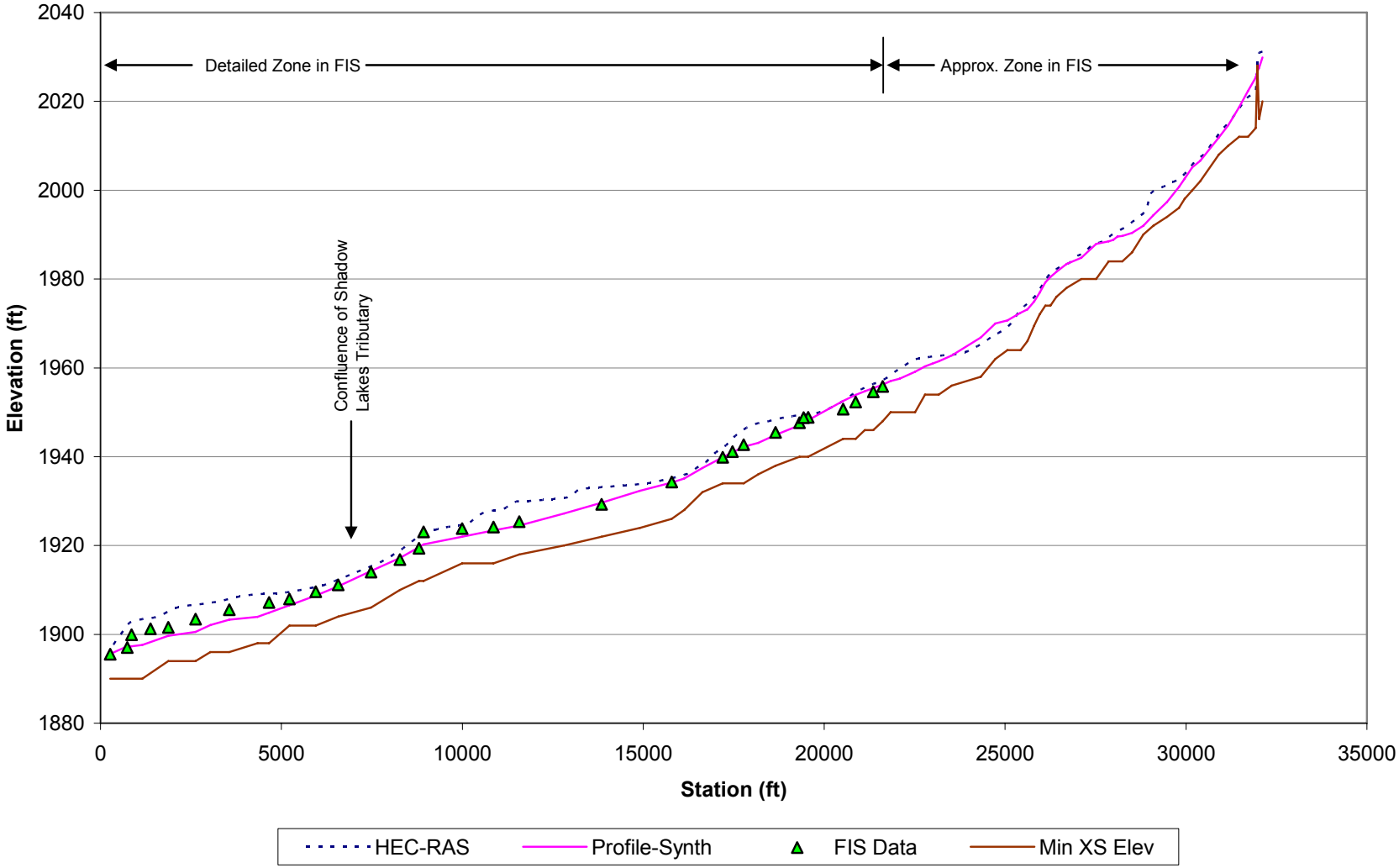


Figure 8. Toms Creek Profile Graph

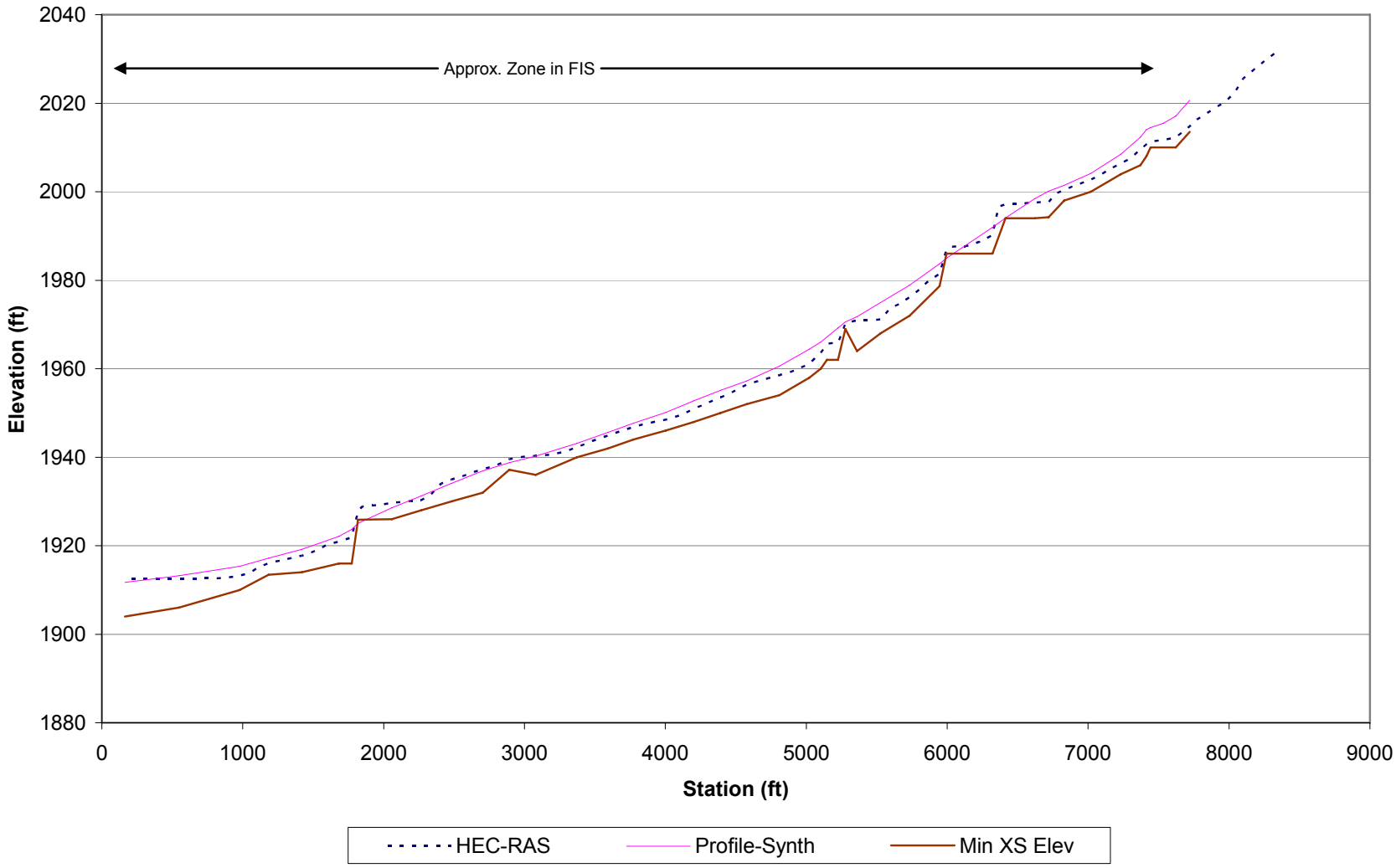


Figure 9. Shadow Lakes Tributary Profile Graph

Table 5. Comparison of Results for Toms Creek Example

Comparison Pair		Net Difference (ft*ft)	Length of Comparison (ft)	Average Profile Elevation Difference (ft)
Toms Creek	HEC-RAS vs. Profile Synthesis	74255.48	31849.07	2.33148
	Profile Synthesis vs. Effective FIS	-15854.9	21354.82	-0.74245
	HEC-RAS vs. Effective FIS	48683.88	21354.82	2.279761
Shadow Lakes Tributary	HEC-RAS vs. Profile Synthesis	-7656.04	7556.946	-1.01311

6.2 Conclusions Regarding the Use of the Profile Synthesis Method

The results of the case applications show that the profile synthesis method is capable of reproducing a water surface profile with reasonable accuracy, provided that certain conditions are met. Under the ideal conditions of the synthetic example, the method reproduced the original profile to within about 0.1 feet. Under the real-world conditions of the Toms Creek example, the method produced a profile that was within about a foot of the original flood insurance study profile, and within about two feet of an approximate hydraulic model. The profile synthesis method should be adequate to improve the shape of riverine approximate flood zones, as minor errors owing to the quasi-hydraulic methodology will tend to be outweighed by the improvements made by redelineating the floodplain boundaries in a consistent manner using a high-resolution terrain model.

The profile synthesis method for approximate floodplain redelineation is only appropriate for use in conjunction with very detailed terrain data, such as that acquired by LiDAR. The level of detailed required in the terrain data will depend on the width and depth of the floodplains to be redelineated. For example, redelineation of shallow floodplains with little out-of-bank flow will require the most detailed terrain data available, while redelineation of deep floodplains dominated by out-of-bank flow may be performed using less detailed terrain data. The determination of whether the available terrain data will support the use of the profile synthesis method must be left to the engineering judgment of the professional in each specific case.

Much of the same base data, such as stream centerlines, cross-sections, and terrain models, is required to implement both the profile synthesis and approximate hydraulic modeling methods. However, the approximate hydraulic modeling method also requires the development of hydrologic flow estimates and hydraulic parameters for use in HEC-RAS. While the hydraulic parameters are often generalized for the purposes of approximate studies, the development of hydrologic flow estimates will often necessitate at least a simple watershed model. These additional tasks related to the approximate hydraulic model could require a few days to a week for hydrologic development, and then one to two days per studied stream for hydraulic model

development. By contrast, the profile synthesis method for approximate floodplain redelineation can reasonably be performed in less than one day per studied stream.

The two methods have different requirements for operator training. While both methods rely on GIS software to manage the base data, the approximate hydraulic modeling method also requires significant knowledge of hydrologic and hydraulic analysis techniques, including the ability to use HEC-RAS. As a result, while the profile synthesis method could potentially be performed by GIS analysts with some water resources engineering experience, the approximate hydraulic modeling method requires a much deeper engineering background. Of course, engineering judgment is still necessary to determine whether the conditions necessary for use of the profile synthesis method have been met, and whether the results are reasonable.

Flood map modernization projects have a limited budget, and the selection of methods to be used in a study is often governed by cost. Without a method such as the profile synthesis method, a map modernization project might be forced to resort to simply transferring the approximate boundaries from the old maps, due to the cost and learning curve commonly associated with a complete restudy of the approximate zones. In flood map redelineation studies especially, the profile synthesis method makes it feasible to improve the quality of approximate zone mapping.

6.3 Future Research

This paper discussed a profile synthesis method for approximate flood boundary redelineation, in comparison to an approximate hydraulic modeling method, which itself is increasing in popularity. The profile synthesis method is interesting in its ability to exploit and utilize existing data. This method has applicability to current flood map modernization work; in fact, consulting firms are probably already using similar methods, although these may not be documented publicly.

The algorithms and workflows presented in this thesis can be expanded upon to improve their efficiency and accuracy. For example, the width-matching macro could be improved in its handling of cross-sectional shapes with abrupt changes in the elevation/width relationship. Related research can continue to improve 1-d modeling workflows in general. For example, the

development and refinement of tools which can automatically create the necessary GIS layers for hydraulic modeling from a terrain model could vastly decrease model preparation time. While some work in this area is already being performed, there is still a need for more intelligent cross-section placement algorithms, as well as techniques for automatic placement of other features required for model construction. These workflow enhancements can continue to advance the field of "automated" approximate hydraulic modeling, in hopes that the burdens of developing the models to perform such analyses can be overcome.

This thesis did not determine which measures are most appropriate to compare two water surface profiles. Two methods were used in this research: absolute differences at each cross-section and total area between profiles. Neither of these measures was entirely adequate to describe the variations in error along the length of the stream profile, and so human interpretation of the profile graphs was still necessary to fully assess the differences in the results. Future research into methods for assessing these errors more descriptively, perhaps by separating the profile into zones based on proximity to road crossings, might prove useful if many water surface profiles must be compared quantitatively.

At a broader level, research that compares the traditional 1-dimensional modeling strategies to newer, multi-dimensional models can be pursued. Currently, most 2-d models are being used in a deliberative fashion to develop highly-detailed models of complex situations, and are not commonly utilized on a county-wide basis. The FEMA map modernization program can act as a practical viewpoint from which to investigate and compare the use of models of varying dimensionality and rigor; as more higher dimensionality models are developed and refined, they may find more frequent application in flood mapping projects.

References

Ackerman, C. T. (2005). "HEC-GeoRAS User's Manual, version 4." U.S. Army Corps of Engineers, Hydrologic Engineering Center, report number CPD-83. *Available at: http://www.hec.usace.army.mil/software/hec-ras/documents/HEC-GeoRAS4_UsersManual.pdf*

Bisese, J. A. (1995). "Methods for estimating the magnitude and frequency of peak discharges of rural, unregulated streams in Virginia." U.S. Geological Survey Water-Resources Investigations Report 94-4148.

Brunner, G.W. (2002). *HEC-RAS Hydraulic Reference Manual, Version 3.1.* U.S. Army Corps of Engineers, Hydrologic Engineering Center, report number CPD-69.

Brunner, G. W. and Hunt, J. H. (1995). "A comparison of the one-dimensional bridge hydraulic routines from: HEC-RAS, HEC-2 and WSPRO." U.S. Army Corps of Engineers, Hydrologic Engineering Center, report number RD-41.

Eichert, B. S. (1968). "Survey of programs for water surface profiles." U.S. Army Corps of Engineers, Hydrologic Engineering Center, Technical Paper #11. *Available at: http://www.hec.usace.army.mil/publications/TechnicalPapers/TP-11.pdf*

Federal Emergency Management Agency (FEMA). (2003). "Guidelines and Specifications for Flood Hazard Mapping Partners," April 2003. *Available at: http://www.fema.gov/plan/prevent/fhm/dl_cgs.shtm*

Federal Emergency Management Agency (FEMA). (2005). "Hydrologic Models Meeting the Minimum Requirement of NFIP, Effective June 2005." *Available at: http://www.fema.gov/plan/prevent/fhm/en_hydro.shtm*

Federal Emergency Management Agency (FEMA). (2006). "Flood Map Modernization: Mid-Course Adjustment – Executive Overview," March 27, 2006. *Available at: http://www.fema.gov/library/viewRecord.do?id=2195*

Gall, M., Boruff, B., and Cutter, S. (2007). "Assessing Flood Hazard Zones in the Absence of Digital Floodplain Maps: Comparison of Alternative Approaches." Natural Hazards Review, 8(1), 1-12.

Horritt, M.S. and Bates, P.D. (2002). "Evaluation of 1D and 2D numerical models for predicting river flood inundation." Journal of Hydrology, 268: 87-99.

North Carolina Cooperating Technical State Mapping Program. (2007). Issue 35: Guidelines for Automated Approximate Studies. *Available at: http://www.ncfloodmaps.com/pubdocs/issue_papers/ip35_final_automated_zone_as.pdf*

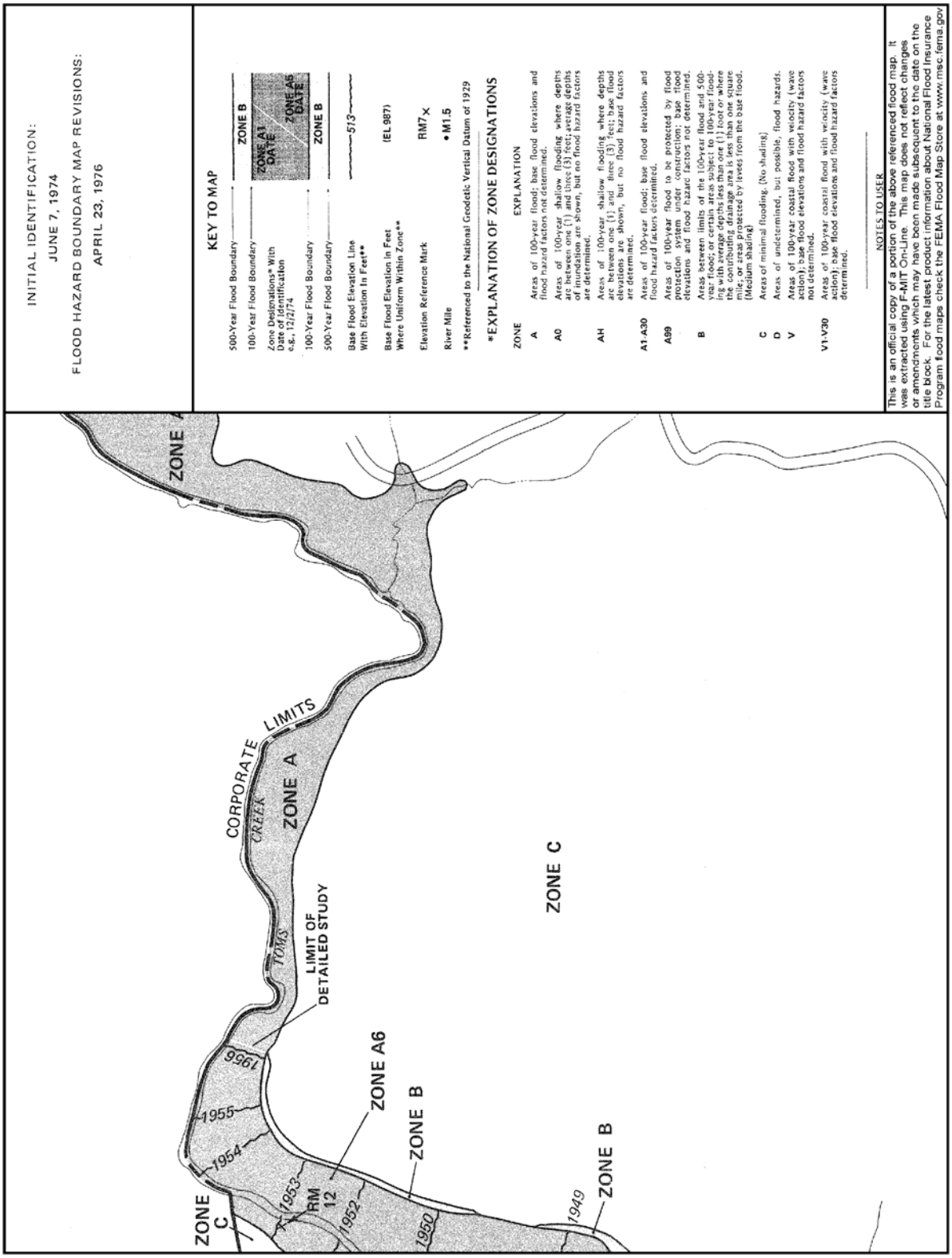
Ries, K.G., III, and Crouse, M.Y. (2002). "The National Flood Frequency Program, Version 3: A Computer Program for Estimating Magnitude and Frequency of Floods for Ungaged Sites." U.S. Geological Survey Water-Resources Investigations Report 02-4168.
The USGS National Flood Frequency (NFF) program is available at http://water.usgs.gov/software/nff.html

Stonestreet, S.E. and Lee, A. S. (2000). "Use of LIDAR Mapping for Floodplain Studies." ASCE Conf. Proc. 104, 58.

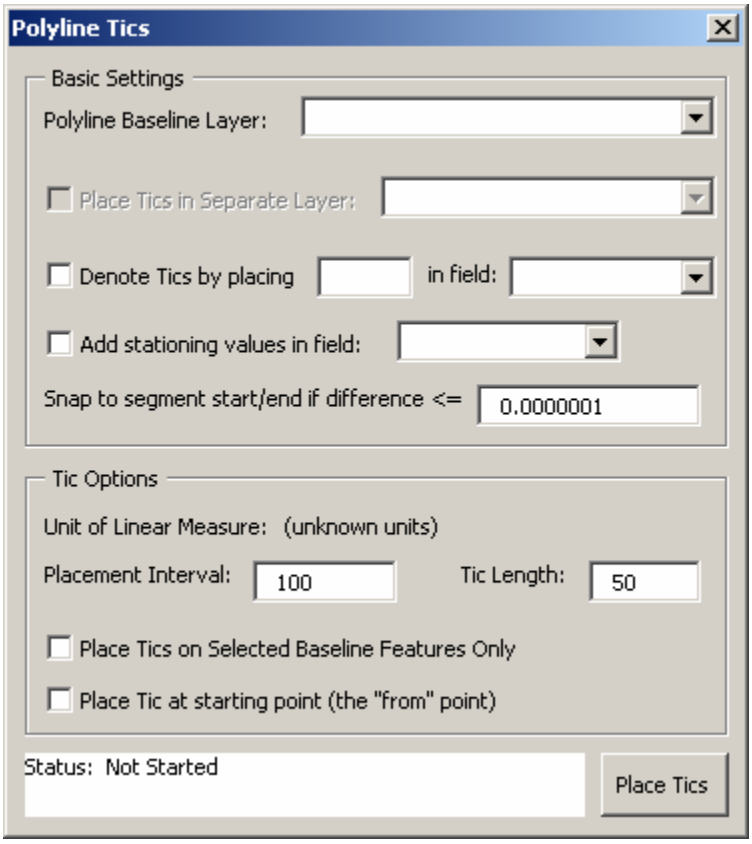
Virginia Geographic Information Network (VGIN). (2003). "Virginia Base Mapping Program (VBMP) Digital Orthophotography Project Handbook." Richmond, VA. *Available at: http://www.vgin.virginia.gov/VBMP/VBMP.htm*

Appendices

Appendix A: Extract from Flood Insurance Rate Map (FEMA Map ID: 5101000003B)



Appendix B: ArcMap VBA Macro: PolylineTics



Screenshot of User Form

Source Code:

```
'Created by Thomas Dickerson
'Required references:  all the usual ESRI libraries
'                MS Scripting Runtime (for File System Object items) --???? this is not used???
'ToDo:  Solve problem of invisible tics when zoomed in under default symbology;
'(this is related to the spatial index:  *.sbn and *.sbx)
'for now, you can just delete the spatial index files (and/or rebuild the index)

Public pMxDoc As IMxDocument
Public pMap As IMap
Public pActiveView As IActiveView

Private Sub UserForm_Initialize()

Dim i As Long, testFeatLayer As IFeatureLayer

Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
Set pActiveView = pMap

'populate comboboxe(s)
For i = 0 To pMap.LayerCount - 1
```

```
    If TypeOf pMap.Layer(i) Is IFeatureLayer Then
        Set testFeatLayer = pMap.Layer(i)
        If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
            cboBaselineLayer.AddItem testFeatLayer.Name
        End If
    End If
Next i

Set testFeatLayer = Nothing


End Sub

Private Sub cboBaselineLayer_Change()

Dim i As Long, j As Long, testFeatLayer As IFeatureLayer, srInfo As ISpatialReferenceInfo, linUM As ILinearUnit
Dim geoDS As IGeoDataset, projCS As IProjectedCoordinateSystem

cboTicField.Clear
cboStationField.Clear

For i = 0 To pMap.LayerCount - 1
    If TypeOf pMap.Layer(i) Is IFeatureLayer Then
        Set testFeatLayer = pMap.Layer(i)
        If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
            If cboBaselineLayer.Text = testFeatLayer.Name Then
                For j = 0 To testFeatLayer.FeatureClass.Fields.FieldCount - 1
                    cboTicField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
                    cboStationField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
                Next j

            'get the unit of linear measure
            Set geoDS = testFeatLayer
            Set srInfo = geoDS.SpatialReference
            lblStatus.Caption = "Baseline SR: " & srInfo.Name & vbNewLine & "Status:  Not Started"
            If TypeOf geoDS.SpatialReference Is IProjectedCoordinateSystem Then
                Set projCS = geoDS.SpatialReference
                Set linUM = projCS.CoordinateUnit
                lblLengthUnits.Caption = linUM.Name
            ElseIf TypeOf geoDS.SpatialReference Is IGeographicCoordinateSystem Then
                lblLengthUnits.Caption = "N/A:  Geographic System"
            Else
                lblLengthUnits.Caption = "Unknown"
            End If

            Exit For
            End If

        End If
    End If
Next i

Set testFeatLayer = Nothing
Set srInfo = Nothing
Set linUM = Nothing
Set geoDS = Nothing
```

```
Set projCS = Nothing

End Sub

Private Sub cmdRun_Click()

Dim i As Long, ticfeatcount As Long
Dim FeatLayer As IFeatureLayer, FeatClass As IFeatureClass
Dim FeatCursor As IFeatureCursor, FeatSel As IFeatureSelection
Dim currfeat As IFeature, segcol As ISegmentCollection, currseg As ISegment
Dim ticPoint As IPoint, ticLine As ILine, trans2D As ITransform2D, newTic As IFeature, newTicPline As IPolyline
Dim pi As Double, constLine As IConstructLine, nextSeg As ISegment
pi = 4 * Atn(1)

Dim TicInterval As Double, TicLength As Double
Dim TicBuild As Double, SegConsumption As Double
Dim TicStation As Double

For i = 0 To pMap.LayerCount - 1
   If TypeOf pMap.Layer(i) Is IFeatureLayer Then
      Set FeatLayer = pMap.Layer(i)
      If FeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
         If cboBaselineLayer.Text = FeatLayer.Name Then
            Set FeatClass = FeatLayer.FeatureClass
            Exit For
         End If
      End If
   End If
Next i

If FeatLayer Is Nothing Or FeatClass Is Nothing Then
   MsgBox "Failed to set layer / feature class; aborting."
   GoTo Abort
End If

'------------preliminary checks--------------------
'Placement Interval
If IsNumeric(txtTicPlacementInterval.Text) = True Then
   If txtTicPlacementInterval.Text > 0 Then
      TicInterval = txtTicPlacementInterval.Text
   Else
      MsgBox "Tic Placement Interval must be positive; aborting."
      GoTo Abort
   End If
Else
   MsgBox "Tic Placement Interval must be a positive number; aborting."
   GoTo Abort
End If
'Length
If IsNumeric(txtTicLength.Text) = True Then
   If txtTicLength.Text > 0 Then
      TicLength = txtTicLength.Text
   Else
      MsgBox "Tic Length must be positive; aborting."
      GoTo Abort
   End If
```

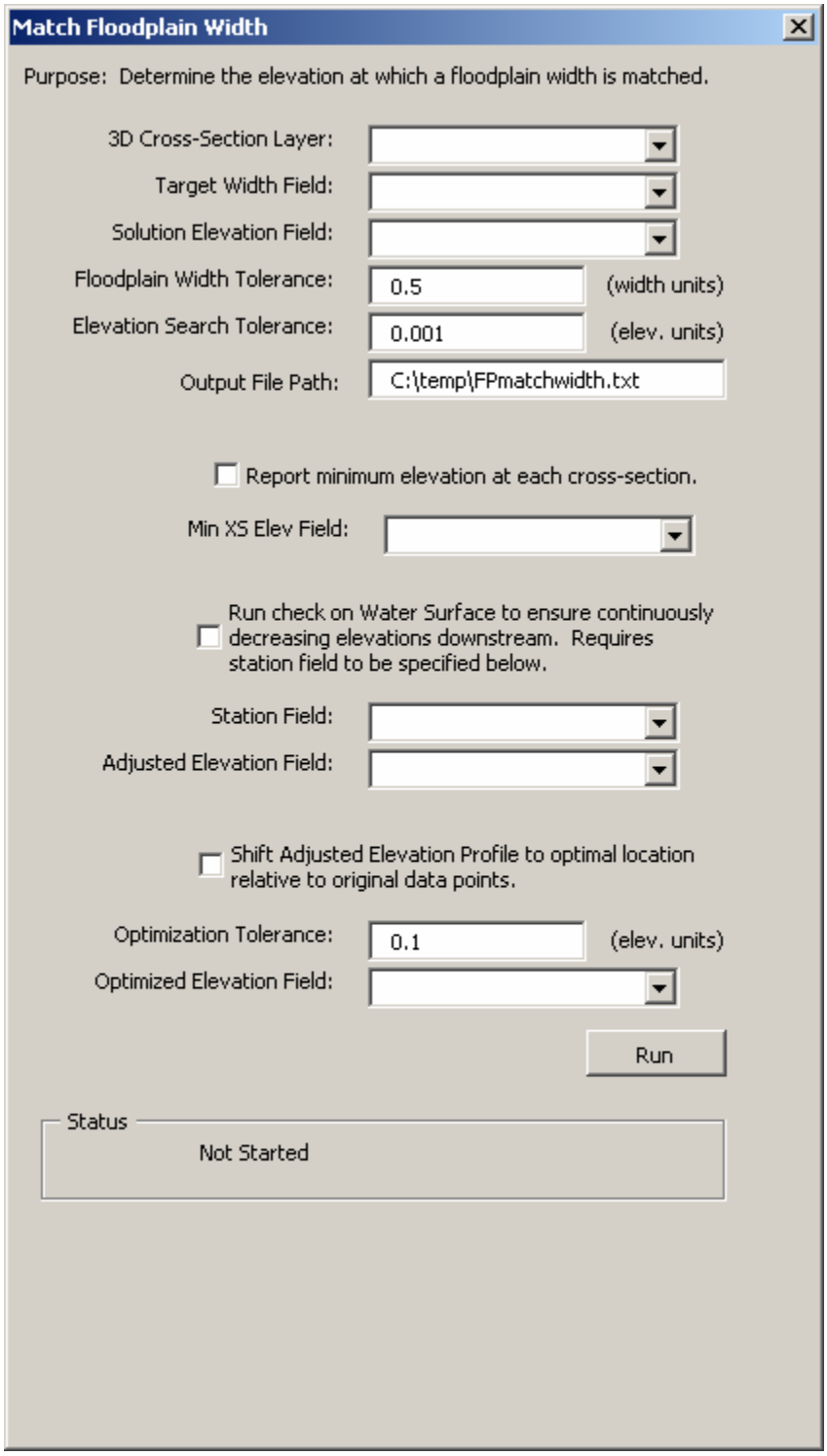
```
Else
   MsgBox "Tic Length must be a positive number; aborting."
   GoTo Abort
End If
'Field denotation
If chkTicDenote.Value = True Then
   'make sure the field type and the proposed indicator are compatible
   If IsNumeric(txtTicIndicator.Text) = False Then
      If Not FeatClass.Fields.Field(cboTicField.ListIndex).Type = esriFieldTypeString Then
         MsgBox "A non-numeric tic indicator was entered, and the specified field is not of the string data type;
aborting."
         GoTo Abort
      End If
   End If

End If

If chkAddStation.Value = True Then
   'make sure the field type is numeric
   If Not FeatClass.Fields.Field(cboStationField.ListIndex).Type = esriFieldTypeDouble Then
      MsgBox "The station field is not double"
      GoTo Abort
   End If
End If

If chkTicSelectedOnly.Value = True Then
   'set a cursor of only the selected features
   Set FeatSel = FeatLayer
   FeatSel.SelectionSet.Search Nothing, False, FeatCursor
Else
   Set FeatCursor = FeatClass.Search(Nothing, False)
End If

Set currfeat = FeatCursor.NextFeature

ticfeatcount = 0
Do While Not currfeat Is Nothing
   TicStation = 0
   ticfeatcount = ticfeatcount + 1
   TicBuild = 0
   SegConsumption = 0
   i = 0

   Set segcol = currfeat.Shape

   If chkIncludeStart.Value = True Then
      'add a tic at the start of the feature
      Set currseg = segcol.Segment(0)
      Set ticLine = New Line
      currseg.QueryTangent esriNoExtension, 0, False, TicLength / 2, ticLine
      Set trans2D = ticLine
      trans2D.Rotate ticLine.FromPoint, pi / 2
      Set ticPoint = New Point
      ticLine.QueryPoint esriExtendEmbedded, -TicLength / 2, False, ticPoint
      ticLine.FromPoint = ticPoint
      Set newTic = FeatClass.CreateFeature
```

```
        Set newTicPline = New Polyline
        newTicPline.FromPoint = ticLine.FromPoint
        newTicPline.ToPoint = ticLine.ToPoint
        Set newTic.Shape = newTicPline
        If chkTicDenote.Value = True Then
           newTic.Value(cboTicField.ListIndex) = txtTicIndicator.Text
        End If
        'If chkAddStation.Value = True Then
        '    newTic.Value(cboStationField.ListIndex) = 0 'this is the start (the from point)
        'End If
        newTic.Store
    End If

    Do While Not i > segcol.SegmentCount - 1
        Set currseg = segcol.Segment(i)

        Debug.Print "Debug:"
        Debug.Print "i= " & i
        Debug.Print "TicBuild= " & TicBuild
        Debug.Print "SegConsumption= " & SegConsumption
        Debug.Print "TicStation= " & TicStation

        Set ticPoint = Nothing
        If Abs((TicBuild + currseg.Length - SegConsumption) - TicInterval) <= txtSnapPrecision.Text Then
           Debug.Print "about equal"
           'tic should be placed exactly on the "To Point" of the current segment
           'place it...
           'it would be good to calculate the average between the two possible segment tangent normals
           Set ticLine = New Line
           If Not i + 1 > segcol.SegmentCount - 1 Then
              Set nextSeg = segcol.Segment(i + 1)
              Set constLine = New Line
              constLine.ConstructAngleBisector currseg.FromPoint, currseg.ToPoint, nextSeg.ToPoint, TicLength / 2,
True
              Set ticLine = constLine
           Else
              'we are going to end exactly on the last vertex; no need to calculate averages
              currseg.QueryTangent esriNoExtension, TicInterval - TicBuild + SegConsumption, False, TicLength / 2,
ticLine
              Set trans2D = ticLine
              trans2D.Rotate ticLine.FromPoint, pi / 2
           End If
           Set ticPoint = New Point
           ticLine.QueryPoint esriExtendEmbedded, -TicLength / 2, False, ticPoint
           ticLine.FromPoint = ticPoint
           Set newTic = FeatClass.CreateFeature
           Set newTicPline = New Polyline
           newTicPline.FromPoint = ticLine.FromPoint
           newTicPline.ToPoint = ticLine.ToPoint
           Set newTic.Shape = newTicPline
           If chkTicDenote.Value = True Then
              newTic.Value(cboTicField.ListIndex) = txtTicIndicator.Text
           End If
           If chkAddStation.Value = True Then
              newTic.Value(cboStationField.ListIndex) = TicStation + TicInterval - TicBuild + SegConsumption
           End If
```

```
            Debug.Print "Station = " & TicStation + TicInterval - TicBuild + SegConsumption

            TicStation = TicStation + currseg.Length

            newTic.Store
            TicBuild = 0
            SegConsumption = 0
            i = i + 1

        ElseIf TicBuild + currseg.Length - SegConsumption > TicInterval Then
            'tic will be placed somewhere on the current segment
            'place it...
            'first, get the tic point
            'currseg.QueryPoint esriNoExtension, TicInterval - TicBuild, False, ticPoint
            Set ticLine = New Line
            currseg.QueryTangent esriNoExtension, TicInterval - TicBuild + SegConsumption, False, TicLength / 2,
ticLine
            Set trans2D = ticLine
            trans2D.Rotate ticLine.FromPoint, pi / 2
            Set ticPoint = New Point
            ticLine.QueryPoint esriExtendEmbedded, -TicLength / 2, False, ticPoint
            ticLine.FromPoint = ticPoint
            Set newTic = FeatClass.CreateFeature
            Set newTicPline = New Polyline
            newTicPline.FromPoint = ticLine.FromPoint
            newTicPline.ToPoint = ticLine.ToPoint
            Set newTic.Shape = newTicPline
            If chkTicDenote.Value = True Then
                newTic.Value(cboTicField.ListIndex) = txtTicIndicator.Text
            End If
            Debug.Print "Station = " & TicStation + TicInterval - TicBuild + SegConsumption
            'TicStation = TicStation + currseg.Length
            If chkAddStation.Value = True Then
                newTic.Value(cboStationField.ListIndex) = TicStation + TicInterval - TicBuild + SegConsumption
            End If
            newTic.Store
            SegConsumption = SegConsumption + TicInterval - TicBuild
            TicBuild = 0

        Else
            'the tic will be on the next segment (or the next, or the next...)
            'add the current segment's legth to the TicBuild length running total and loop
            TicBuild = TicBuild + currseg.Length
            TicStation = TicStation + currseg.Length
            i = i + 1
        End If

    Loop

Set currfeat = FeatCursor.NextFeature
Loop


Abort:
Set FeatLayer = Nothing
Set FeatClass = Nothing
```

```
Set FeatCursor = Nothing
Set FeatSel = Nothing
Set currfeat = Nothing
Set segcol = Nothing
Set currseg = Nothing
Set ticPoint = Nothing
Set ticLine = Nothing
Set trans2D = Nothing
Set newTic = Nothing
Set newTicPline = Nothing

lblStatus.Caption = "Status:  Done; " & ticfeatcount & " features ticked."

End Sub
```

Appendix C: ArcMap VBA Macro: MatchFloodplainWidth



Screenshot of User Form
(the last two checkbox options unused, due to the new smoothing algorithm in Appendix D)

Source Code:

```
'Created by Thomas Dickerson (last update: 12/4/2006)
'Required references:  all the usual ESRI libraries

'comments:
'maybe the IZ or IZAware Interfaces will help?

Public pMxDoc As IMxDocument
Public pMap As IMap
Public pActiveView As IActiveView


Private Sub UserForm_Initialize()

Dim i As Long, testFeatLayer As IFeatureLayer

Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
Set pActiveView = pMap

'populate comboboxe(s)
For i = 0 To pMap.LayerCount - 1
   If TypeOf pMap.Layer(i) Is IFeatureLayer Then
      Set testFeatLayer = pMap.Layer(i)
      If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
         cbo3DXsec.AddItem testFeatLayer.Name
      End If
   End If
Next i

Set testFeatLayer = Nothing


End Sub

Private Sub cbo3DXsec_Change()

Dim i As Long, j As Long, testFeatLayer As IFeatureLayer

cboTargetWidthField.Clear
cboSolutionElevationField.Clear
cboStation.Clear
cboAdjustedElevField.Clear
cboOptimizedElevField.Clear
cboMinXSElev.Clear

For i = 0 To pMap.LayerCount - 1
   If TypeOf pMap.Layer(i) Is IFeatureLayer Then
      Set testFeatLayer = pMap.Layer(i)
      If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
         If cbo3DXsec.Text = testFeatLayer.Name Then
            For j = 0 To testFeatLayer.FeatureClass.Fields.FieldCount - 1
               cboTargetWidthField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
               cboSolutionElevationField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
```

```
            cboStation.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
            cboAdjustedElevField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
            cboOptimizedElevField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
            cboMinXSElev.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
          Next j

        Exit For
        End If

      End If
   End If
Next i

Set testFeatLayer = Nothing


End Sub


Private Sub cmdRun_Click()

Dim i As Long, j As Long, featloopCount As Long, solutionIterationCount As Long
Dim FeatLayer As IFeatureLayer, FeatClass As IFeatureClass
Dim FeatCursor As IFeatureCursor
Dim currXSfeat As IFeature, pline As IPolyline, pointCol As IPointCollection

Dim xsPrevPoint(0 To 2) As Double, prevDist As Double
Dim xsVertices()
Dim maxElev As Double, minElev As Double, topLimitElev As Double, bottomLimitElev As Double,
elevTolerance As Double
Dim proposedElev As Double, targetWidth As Double, continueSearch As Boolean
Dim dx As Double, dz As Double, mSlope As Double, Bint As Double
Dim xHit As Double
Dim HitList()
Dim hitString As String

Dim currWSstatus As String
Dim WSwidth As Double, netWSwidth As Double

Dim WSEcheckArray()
Dim LinInterpLower(0 To 1) As Double, LinInterpUpper(0 To 1) As Double
Dim LinInterpM As Double, LinInterpB As Double
Dim WSEcheckRepeat As Boolean
Dim WSEmaxdiff As Double, WSEShift As Double, SumSqDev As Double
Dim WSEminShift As Double, WSEmaxShift As Double
Dim currentOptimumShift(0 To 1) As Double
Dim overallOptimumShift(0 To 1) As Double
Dim OptimizationLoop As Boolean, optimizationIterationCount As Long

Dim fso As FileSystemObject, textOut As TextStream

'Dim TopoOp As ITopologicalOperator

lblStatusMsg.Caption = "Started"

For i = 0 To pMap.LayerCount - 1
```

```
    If TypeOf pMap.Layer(i) Is IFeatureLayer Then
        Set FeatLayer = pMap.Layer(i)
        If FeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
            If cbo3DXsec.Text = FeatLayer.Name Then
                Set FeatClass = FeatLayer.FeatureClass
                Exit For
            End If
        End If
    End If
Next i

'------------preliminary checks--------------------
If FeatLayer Is Nothing Or FeatClass Is Nothing Then
    MsgBox "Failed to set layer / feature class; aborting."
    GoTo Abort
End If
If cboTargetWidthField.Value = "" Or cboTargetWidthField.Value = Null Then
    MsgBox "Failed to set Target Width Field; aborting."
    GoTo Abort
End If
If cboSolutionElevationField.Value = "" Or cboSolutionElevationField.Value = Null Then
    MsgBox "Failed to set Solution Elevation Field; aborting."
    GoTo Abort
End If

Set fso = New FileSystemObject
Set textOut = fso.CreateTextFile(txtOutputFilePath.Text, True)

textOut.WriteLine "Output from 'Match Floodplain Width' macro started: " & Now


'Do loop for each feature in the 3d cross-section layer
Set FeatCursor = FeatClass.Search(Nothing, False)
Set currXSfeat = FeatCursor.NextFeature

featloopCount = 0

Do While Not currXSfeat Is Nothing

    'build an array of the cross-section vertices
    'this process will proceed by starting at the FromPoint, and then calculating the (x,y) distance to the next point
    '(this will "stretch out" XSections with internal wrinkes onto a flat plane)

    featloopCount = featloopCount + 1

    lblStatusMsg.Caption = "Feature #: " & featloopCount
    Me.Repaint

    textOut.WriteLine "XS feature FID: " & currXSfeat.OID
    targetWidth = currXSfeat.Value(cboTargetWidthField.ListIndex)

    Set pointCol = currXSfeat.Shape
    ReDim xsVertices(0 To 1, 0 To 0)
    For i = 0 To pointCol.PointCount - 1
        If i = 0 Then
            'this is the first point (hopefuly the "from" point?)
```

```
            'log the start point as (0,Z)
            xsVertices(0, UBound(xsVertices, 2)) = 0
            xsVertices(1, UBound(xsVertices, 2)) = pointCol.Point(i).Z
            'add the next blank slot
            ReDim Preserve xsVertices(0 To 1, 0 To UBound(xsVertices, 2) + 1)

            'record the datum point for future math
            xsPrevPoint(0) = pointCol.Point(i).X
            xsPrevPoint(1) = pointCol.Point(i).Y
            xsPrevPoint(2) = pointCol.Point(i).Z

            'record the previous distance covered (starts as zero)
            prevDist = 0

        Else
            'this is a subsequent point
            'log the point based on math from the previous point
            prevDist = prevDist + ((((pointCol.Point(i).X - xsPrevPoint(0)) ^ 2) + ((pointCol.Point(i).Y - xsPrevPoint(1))
^ 2)) ^ (1 / 2))
            xsVertices(0, UBound(xsVertices, 2)) = prevDist
            xsVertices(1, UBound(xsVertices, 2)) = pointCol.Point(i).Z

            'add the next blank slot
            ReDim Preserve xsVertices(0 To 1, 0 To UBound(xsVertices, 2) + 1)

            'record the  point for future math
            xsPrevPoint(0) = pointCol.Point(i).X
            xsPrevPoint(1) = pointCol.Point(i).Y
            xsPrevPoint(2) = pointCol.Point(i).Z

        End If

    Next i

    'chop off the last blank slot
    ReDim Preserve xsVertices(0 To 1, 0 To UBound(xsVertices, 2) - 1)
    'now we have finished the array of cross-section points
    'the XS has been flattened to an X,Z coordinate system, where X is measured in the (X,Y) plane from the starting
point,
    'and Z is the true elevation

    'Debug-type output of raw cross-section data (should be unnecessary)
    textOut.WriteLine "XSdist, Elev"
    For i = 0 To UBound(xsVertices, 2)
        textOut.WriteLine xsVertices(0, i) & "," & xsVertices(1, i)
    Next i

    'find the maximum & minimum elevation
    For i = 0 To UBound(xsVertices, 2)
        If i = 1 Then
            maxElev = xsVertices(1, i)
            minElev = xsVertices(1, i)
        Else
            If xsVertices(1, i) > maxElev Then
                maxElev = xsVertices(1, i)
            End If
```

```
        If xsVertices(1, i) < minElev Then
            minElev = xsVertices(1, i)
        End If
    End If
Next i

'if desired, report the minimum elevation at this cross-section
If chkReportMinXSElev.Value = True Then
    currXSfeat.Value(cboMinXSElev.ListIndex) = minElev
    currXSfeat.Store
End If

'topLimitElev and bottomLimitElev will hold an up-to-date elevation range within which the solution is expected
to lie
topLimitElev = maxElev
bottomLimitElev = minElev
'the first proposed elevation will be the half-way point of the range between min and max elevations
proposedElev = minElev + (0.5 * (maxElev - minElev))
textOut.WriteLine "Target Water Surface Width: " & targetWidth

solutionIterationCount = 0

Do

'the "HitList" will store the intersections' X values (distance along the XS) and a comment indicating
'whether the intersection occurred in a rising or falling manner
ReDim HitList(0 To 1, 0 To 0)

solutionIterationCount = solutionIterationCount + 1
textOut.WriteLine "Proposed Water Surface Elevation: " & proposedElev
textOut.WriteLine "Top Limit Elev: " & topLimitElev
textOut.WriteLine "Bottom Limit Elev: " & bottomLimitElev


'Cycle through the segments of the cross-section, noting the intersection points of the proposed elevation with the
xs geometry
For i = 0 To UBound(xsVertices, 2)
    If i = 0 Then
        'first time through:  don't do anything (we need to get to the point where we have a vertex behind us before
we can make a segment)
    Else
        'normal procedure:  find intersections of the segments with the proposed elevation
        'note that the i-th vertex is the end point of the segment being analyzed
        If xsVertices(1, i) < proposedElev And xsVertices(1, i - 1) > proposedElev Then
            'this segment starts above the Elev and ends below the Elev (the beginning of a section of water surface)
            dx = xsVertices(0, i) - xsVertices(0, i - 1)
            dz = xsVertices(1, i) - xsVertices(1, i - 1)

            If dx = 0 Then
                'the segment is vertical; the intersection occurs at the segment's X value
                xHit = xsVertices(0, i)
            Else
                'the segment is not vertical; follow normal intersection calculation procedure
                mSlope = dz / dx
                Bint = xsVertices(1, i) - (mSlope * xsVertices(0, i))
                xHit = (proposedElev - Bint) / mSlope
```

```
            End If

            HitList(0, UBound(HitList, 2)) = xHit
            HitList(1, UBound(HitList, 2)) = "startWS"
            ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)


        ElseIf xsVertices(1, i) > proposedElev And xsVertices(1, i - 1) < proposedElev Then
            'this segment starts below the Elev and ends above the Elev (the end of a section of water surface)
            dx = xsVertices(0, i) - xsVertices(0, i - 1)
            dz = xsVertices(1, i) - xsVertices(1, i - 1)

            If dx = 0 Then
                'the segment is vertical; the intersection occurs at the segment's X value
                xHit = xsVertices(0, i)
            Else
                'the segment is not vertical; follow normal intersection calculation procedure
                mSlope = dz / dx
                Bint = xsVertices(1, i) - (mSlope * xsVertices(0, i))
                xHit = (proposedElev - Bint) / mSlope
            End If

            HitList(0, UBound(HitList, 2)) = xHit
            HitList(1, UBound(HitList, 2)) = "endWS"
            ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)


        ElseIf xsVertices(1, i - 1) = proposedElev Or xsVertices(1, i) = proposedElev Then
            'the "start" and/or "end" point of this segment lie exactly on the proposedElev
            'we will have a special section to handle the situation where the very first vertex
            'was on the proposedElev line, but otherwise we will only handle the end vertices of the segments

            'Special Block for if the very first vertex was at the proposedElev:
            If i = 1 And xsVertices(1, i - 1) = proposedElev Then
                'this is the first time we are analyzing any segment in the XS, and the first point was on the
proposedElev
                'look at the other end of the segment to decide whether this is rising or falling?
                dx = xsVertices(0, i) - xsVertices(0, i - 1)
                dz = xsVertices(1, i) - xsVertices(1, i - 1)

                If dx = 0 And dz = 0 Then
                    Debug.Print "dx = 0 and dz = 0"
                ElseIf dz = 0 Then
                    'segment is horizontal
                    HitList(0, UBound(HitList, 2)) = xsVertices(0, i - 1)
                    'classify this as "startWS"
                    HitList(1, UBound(HitList, 2)) = "startWS"
                    'add next slot
                    ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)
                Else
                    HitList(0, UBound(HitList, 2)) = xsVertices(0, i - 1)
                    If dz > 0 Then 'the elevation at the "end" point rises above the proposedElev
                        'classify this as "endWS"
                        HitList(1, UBound(HitList, 2)) = "endWS"
                    Else
                        'classify this as "startWS"
```

```
            HitList(1, UBound(HitList, 2)) = "startWS"
          End If
          'add next slot
          ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)
        End If

      End If

      'Normal handling procedure is to ignore the start vertex and operate based only on the end vertex:
      If xsVertices(1, i) = proposedElev Then
        'the end point is on the proposedElev line
        'look at the other end of the segment to decide whether this is rising or falling?
        dx = xsVertices(0, i) - xsVertices(0, i - 1)
        dz = xsVertices(1, i) - xsVertices(1, i - 1)

        If dx = 0 And dz = 0 Then
          Debug.Print "dx = 0 and dz = 0"
        ElseIf dz = 0 Then
          'segment is horizontal
          If i = UBound(xsVertices, 2) Then
            'we are at the last vertex of the last segment (there is no next vertex)
            'do nothing (presumably we will close out any unclosed water surfaces during a subsequent
review)
          ElseIf xsVertices(1, i + 1) = xsVertices(1, i) Then
            'the next vertex is also at the same elevation (do nothing)
          Else
            'the next vertex is not at the same elevation
            If xsVertices(1, i + 1) > xsVertices(1, i) Then
              'the next vertex is higher than this one
              HitList(0, UBound(HitList, 2)) = xsVertices(0, i - 1)
              'classify this as "endWS"
              HitList(1, UBound(HitList, 2)) = "endWS"
              'add next slot
              ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)

            Else
              'the next vertex is lower than this one (do nothing, as we have already started the WS)

            End If
          End If

        Else
          'the end vertex is on the proposedElev, and the segment is not horizontal
          HitList(0, UBound(HitList, 2)) = xsVertices(0, i - 1)
          If dz > 0 Then 'the elevation at the "start" point was below the proposedElev
            'classify this as "endWS"
            HitList(1, UBound(HitList, 2)) = "endWS"
          Else
            'classify this as "startWS"
            HitList(1, UBound(HitList, 2)) = "startWS"
          End If
          'add next slot
          ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) + 1)

        End If
```

```
            End If
        End If
    End If
Next i

'chop off the last blank slot
ReDim Preserve HitList(0 To 1, 0 To UBound(HitList, 2) - 1)

'let's output the intersection points just to see what the results were
textOut.WriteLine "HitX, StartOrEnd"
For i = 0 To UBound(HitList, 2)
    textOut.WriteLine HitList(0, i) & "," & HitList(1, i)
Next i


'ToDo:
'Compare calculated width against the target width, and then propose a new WS elevation.
'Add loop back to immediately after the first guessed water surface elevation;
'keep looping until the total width of water surface comes within some tolerance of the target,
'and/or give up in the event of unexplained situations without a solution
'and/or quit if the iterations pass some maximum value (user-specified)

'Determine width:
WSwidth = 0
currWSstatus = "unspecified"
For i = 0 To UBound(HitList, 2)
    'we are going to focus mainly on the second through final results, always looking at the previous value
    If i = 0 Then
        'the only thing we need to do on the first loop is to see if we need to end an unresolved WS
        'also, if there is only one hit in the hitlist, then this is the first and last loop
        Select Case HitList(1, i)
            Case "startWS"
                'this would make sense (first loop and starting a WS),
                'we only need to take action if there is only going to be one loop
                If UBound(HitList, 2) = 0 Then
                    'unresolved; add the distance from here to the end of the whole XSec
                    WSwidth = WSwidth + (xsVertices(0, UBound(xsVertices, 2)) - HitList(0, i))
                End If

            Case "endWS"
                'this would make less sense (first loop and ending a WS), and indicates that the cross-section isn't long
enough
                WSwidth = WSwidth + HitList(0, i) '(WSwidth should be zero before we complete this line of code)

            Case Else
                'I don't know what this could be (probably an error)
                'Debug.Print "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i) " & Hitlist(1," & i
& ") = " & HitList(1, i)
                textOut.WriteLine "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i) & " &
Hitlist(1," & i & ") = " & HitList(1, i)
        End Select

    Else
        Select Case HitList(1, i)
            Case "startWS"
                'this would make sense if the previous item said "endWS"
```

```
            'if we are at the last item, then the WS is unresolved
            If i = UBound(HitList, 2) Then
               'unresolved; add the distance from here to the end of the whole XSec
               WSwidth = WSwidth + (xsVertices(0, UBound(xsVertices, 2)) - HitList(0, i))
            Else
               'normal (ignore for now; pick up when we end the WS)

            End If

         Case "endWS"
            'this would make sense if the previous item said "startWS"
            If HitList(1, i - 1) = "startWS" Then
               'good; we are ending a water surface which was started previously
               'add the distance from the previous point through this point
               WSwidth = WSwidth + (HitList(0, i) - HitList(0, i - 1))

            Else
               'we are ending a WS, but the previous item didn't say start WS
               'I don't know what this could be (probably an error)
               'Debug.Print "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i); " & Hitlist(1,"
& i & ") = " & HitList(1, i)
               textOut.WriteLine "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i) & " &
Hitlist(1," & i & ") = " & HitList(1, i)
            End If

         Case Else
            'I don't know what this could be (probably an error)
            'Debug.Print "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i) " & Hitlist(1," & i
& ") = " & HitList(1, i)
            textOut.WriteLine "Error in WS width calculation.  Hitlist(0," & i & ") = " & HitList(0, i) & " &
Hitlist(1," & i & ") = " & HitList(1, i)
      End Select


   End If

Next i

textOut.WriteLine "WSwidth = " & WSwidth


'ToDo:  continue writing code to decide how/when to loop
continueSearch = False
If Abs(targetWidth - WSwidth) < txtSolutionTolerance.Value Then
   'the current width is close enough to the target; stop the search...we have found an answer!
   textOut.WriteLine "Solution (" & proposedElev & ") Found on iteration # " & solutionIterationCount
   'It would be nice if we could output a single line containing the XS FID and the location of all the hits
   '(for future post-processing in other macros)
   'Build a string by looping through the hitlist
   hitString = ""
   For i = 0 To UBound(HitList, 2)
      hitString = hitString & HitList(0, i) & ","
   Next i
   hitString = Left(hitString, Len(hitString) - 1)
   textOut.WriteLine "*FIDandHits:" & currXSfeat.OID & "," & hitString
```

```
      currXSfeat.Value(cboSolutionElevationField.ListIndex) = proposedElev
      currXSfeat.Store
      continueSearch = False
   Else
      'keep going
      'unless we have been going way too long...(?)
      If solutionIterationCount > 100 Then
         continueSearch = False
         textOut.WriteLine "Number of Solution Iterations has exceeded the max. allowable value; quitting."
      Else
         'select the next trial solution, and update the search window
         If WSwidth > targetWidth Then
            'current result is wider than desired; try to pick a lower elevation
            If proposedElev > bottomLimitElev + txtElevationTolerance.Value Then
               topLimitElev = proposedElev
               proposedElev = bottomLimitElev + (0.5 * (topLimitElev - bottomLimitElev))
               continueSearch = True
            Else
               'we are already too close to the bottom limit; report no solution
               textOut.WriteLine "Solution Search wanted to pick a lower elevation, but we were already at the bottom
limit"
               continueSearch = False
            End If
         ElseIf WSwidth < targetWidth Then
            'current result is smaller than desired; try to pick a higher elevation
            If proposedElev < topLimitElev - txtElevationTolerance.Value Then
               bottomLimitElev = proposedElev
               proposedElev = bottomLimitElev + (0.5 * (topLimitElev - bottomLimitElev))
               continueSearch = True
            Else
               'we are already too close to the top limit; report no solution
               textOut.WriteLine "Solution Search wanted to pick a higher elevation, but we were already at the top
limit"
               continueSearch = False
            End If
         Else
            'I don't know how we could get here
            textOut.WriteLine "Solution Search encountered an unknown problem, and will stop."
            continueSearch = False
         End If
      End If
   End If

   Loop While continueSearch = True


   Set currXSfeat = FeatCursor.NextFeature

   'defuse the loop (for debugging purposes)
   'GoTo abort

Loop

'Now we have gone through all of the cross-section features

'optional check on water surface elevation consistency
```

```
If chkWSEcheck.Value = True Then
   lblStatusMsg.Caption = "Checking WSE consistency"
   Me.Repaint
   'get an array of FID, station, and elevation; loop though this array backwards to ensure
   'continually decreasing elevations downstream
   'we will make WSEcheckArray have four columns:  0=FID, 1=Station, 2=OriginalElevation,
3=AdjustedElevation
   Set currXSfeat = Nothing
   Set FeatCursor = Nothing

   ReDim WSEcheckArray(0 To 3, 0 To 0)
   Set FeatCursor = FeatClass.Search(Nothing, False)
   Set currXSfeat = FeatCursor.NextFeature
   Do While Not currXSfeat Is Nothing
      i = UBound(WSEcheckArray, 2)
      WSEcheckArray(0, i) = currXSfeat.OID
      WSEcheckArray(1, i) = currXSfeat.Value(cboStation.ListIndex)
      WSEcheckArray(2, i) = currXSfeat.Value(cboSolutionElevationField.ListIndex)
      WSEcheckArray(3, i) = currXSfeat.Value(cboSolutionElevationField.ListIndex)

      ReDim Preserve WSEcheckArray(0 To 3, 0 To i + 1)

      Set currXSfeat = FeatCursor.NextFeature
   Loop
   'chop off the last blank slot
   ReDim Preserve WSEcheckArray(0 To 3, 0 To UBound(WSEcheckArray, 2) - 1)

   'now we have all the data in the WSEcheckArray
   'be sure that the array is sorted in order of increasing station
   SelectionSort WSEcheckArray, 1, "Ascending"

   'Debug.Print "Ascending:"
   'For i = 0 To UBound(WSEcheckArray, 2)
   '    Debug.Print WSEcheckArray(0, i) & "," & WSEcheckArray(1, i) & "," & WSEcheckArray(3, i)
   'Next i


   'note that the following procedure assumes that the most upstream elevation is valid

   'loop backwards through the array, setting invalid WSEs to zero
   'ToDo: linearly interpolate water surface elevations on the invalid data X-sections
   For i = UBound(WSEcheckArray, 2) - 1 To 0 Step -1
      'be sure that the ith elevation is less than or equal to the next upstream station elevation

      'find the next valid upstream station elevation
      j = i + 1
      Do While WSEcheckArray(3, j) <= 0
         If j < UBound(WSEcheckArray, 2) Then
            j = j + 1
         Else
            j = -1
            Exit Do
         End If
      Loop

      If Not j = -1 Then
```

```
            If WSEcheckArray(3, i) > WSEcheckArray(3, j) Then
                WSEcheckArray(3, i) = 0
            End If
        End If

    Next i

    'For i = 0 To UBound(WSEcheckArray, 2)
    '    Debug.Print WSEcheckArray(0, i) & "," & WSEcheckArray(1, i) & "," & WSEcheckArray(3, i)
    'Next i


    Do

        WSEcheckRepeat = False

        For i = 0 To UBound(WSEcheckArray, 2)
            If WSEcheckArray(3, i) = 0 Then
                If i = 0 Then
                    'the first cross-section is invalid
                ElseIf i = UBound(WSEcheckArray, 2) Then
                    'the last cross-section is invalid
                Else
                    'one of the middle cross-sections is invalid
                    j = i - 1
                    Do While WSEcheckArray(3, j) = 0
                        If j > 0 Then
                            j = j - 1
                        Else
                            'we need to find a lower XS, but none are were valid
                            'find a way to abort cleanly, but for now:
                            j = -1
                            Exit Do
                        End If
                    Loop
                    LinInterpLower(0) = j

                    j = i + 1
                    Do While WSEcheckArray(3, j) = 0
                        If j < UBound(WSEcheckArray, 2) Then
                            j = j + 1
                        Else
                            'we need to find a lower XS, but none are were valid
                            'find a way to abort cleanly, but for now:
                            j = -1
                            Exit Do
                        End If
                    Loop
                    LinInterpUpper(0) = j


                    If LinInterpLower(0) <> -1 And LinInterpUpper(0) <> -1 Then
                        'we can probably proceed with interpolation, as valid lower and upper interpolation bounds were
found
                        'get real lower and upper values
                        LinInterpLower(1) = WSEcheckArray(3, LinInterpLower(0))
```

```
                LinInterpLower(0) = WSEcheckArray(1, LinInterpLower(0))
                LinInterpUpper(1) = WSEcheckArray(3, LinInterpUpper(0))
                LinInterpUpper(0) = WSEcheckArray(1, LinInterpUpper(0))

                If Not Abs(LinInterpUpper(0) - LinInterpLower(0)) = 0 Then
                   'calculate slope as dy/dx
                   LinInterpM = (LinInterpUpper(1) - LinInterpLower(1)) / (LinInterpUpper(0) - LinInterpLower(0))
                   'calculate y-intercept (B) as y-mx
                   LinInterpB = LinInterpUpper(1) - (LinInterpM * LinInterpUpper(0))

                   'interpolate an elevation for the ith cross-section, as well as any adjacent invalid cross-sections
                   WSEcheckArray(3, i) = (LinInterpM * WSEcheckArray(1, i)) + LinInterpB
                   'look for higher xs needing an interpolated WSE
                   j = i + 1
                   Do While j <= UBound(WSEcheckArray, 2)
                      If WSEcheckArray(3, j) = 0 Then
                         WSEcheckArray(3, j) = (LinInterpM * WSEcheckArray(1, j)) + LinInterpB
                         j = j + 1
                      Else
                         Exit Do
                      End If
                   Loop
                   'look for lower xs needing an interpolated WSE
                   j = i - 1
                   Do While j >= 0
                      If WSEcheckArray(3, j) = 0 Then
                         WSEcheckArray(3, j) = (LinInterpM * WSEcheckArray(1, j)) + LinInterpB
                         j = j - 1
                      Else
                         Exit Do
                      End If
                   Loop

                   'get out of this for loop so that we can go back to the beginning (via the do loop)
                   WSEcheckRepeat = True
                   Exit For
                End If

             End If

          End If

       End If
    Next i


Loop While WSEcheckRepeat = True

'go through the features and write the adjusted elevations
Set currXSfeat = Nothing
Set FeatCursor = Nothing

For i = 0 To UBound(WSEcheckArray, 2)
   Set currXSfeat = FeatClass.GetFeature(WSEcheckArray(0, i))
   currXSfeat.Value(cboAdjustedElevField.ListIndex) = WSEcheckArray(3, i)
   currXSfeat.Store
```

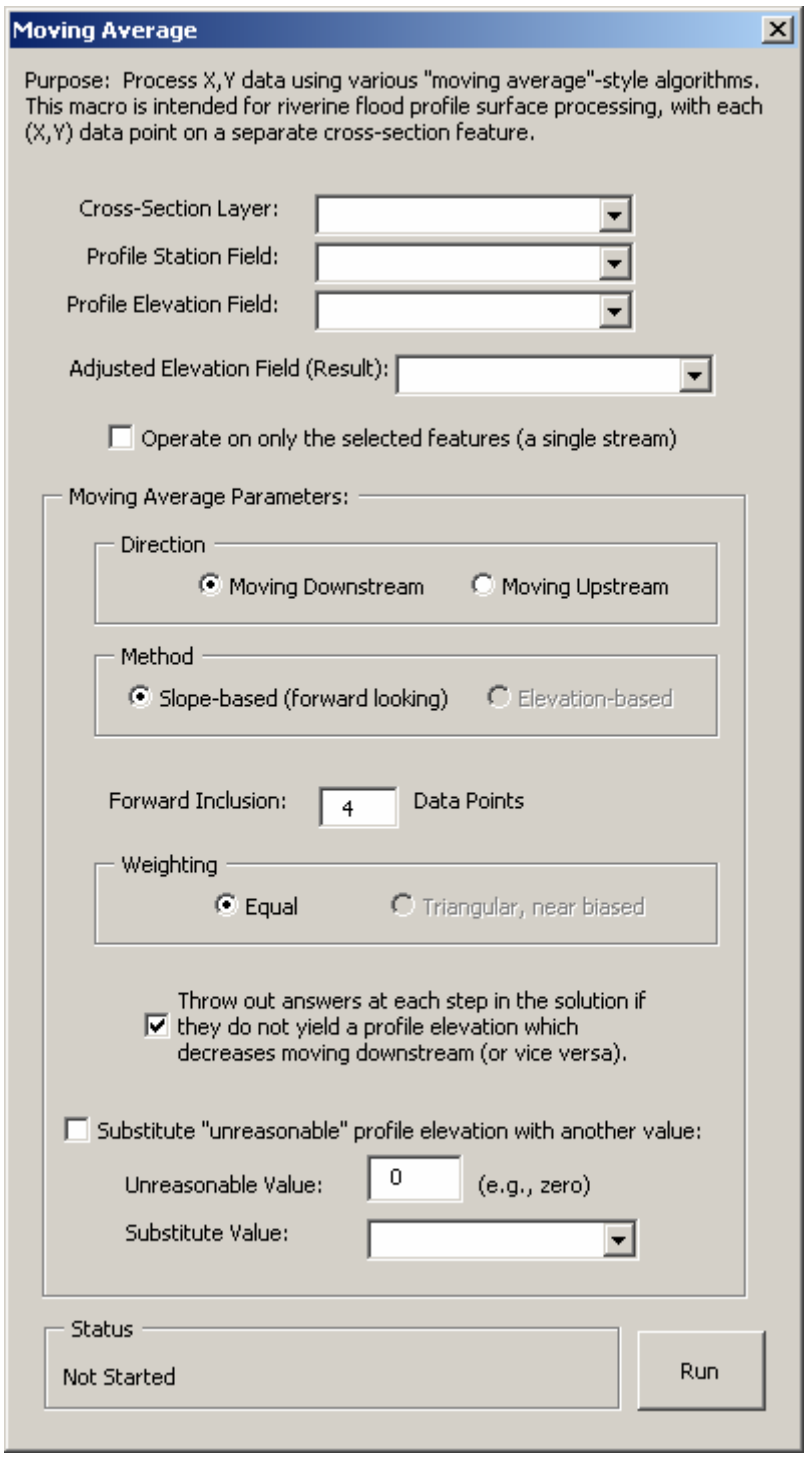
```
    Next i


End If

'try to determine optimal shift for minimal profile to minimize the overall error
'(be sure to exclude the points for which no solution was found)
If chkWSEcheck.Value = True And chkOptimalShift.Value = True Then
    lblStatusMsg.Caption = "Optimizing Profile Shift"
    Me.Repaint
    textOut.WriteLine "Starting Profile Shift Optimization"

    'WSEcheckArray has four columns:  0=FID, 1=Station, 2=OriginalElevation, 3=AdjustedElevation

    'find the max difference
    WSEmaxdiff = -1
    For i = 0 To UBound(WSEcheckArray, 2)
        If WSEcheckArray(3, i) <> 0 And WSEcheckArray(2, i) <> 0 Then
            If WSEmaxdiff = -1 Then
                WSEmaxdiff = WSEcheckArray(2, i) - WSEcheckArray(3, i)
            End If

            If WSEcheckArray(2, i) - WSEcheckArray(3, i) > WSEmaxdiff Then
                WSEmaxdiff = WSEcheckArray(2, i) - WSEcheckArray(3, i)
            End If

        End If

    Next i


    optimizationIterationCount = 0
    overallOptimumShift(0) = -1
    overallOptimumShift(1) = -1

    'The first sampling will be a spread from 0 to the max diff
    WSEminShift = 0
    WSEmaxShift = WSEmaxdiff
    WSEShift = WSEminShift

    Do
        textOut.WriteLine "WSEminShift: " & WSEminShift
        textOut.WriteLine "WSEmaxShift: " & WSEmaxShift
        textOut.WriteLine "Shift Step: " & ((WSEmaxShift - WSEminShift) / 30)

        currentOptimumShift(0) = -1
        currentOptimumShift(1) = -1

        Do While WSEShift <= WSEmaxShift

            SumSqDev = 0
            For i = 0 To UBound(WSEcheckArray, 2)
                'consider data points where the original solution was valid, or if invalid, was replaced by interpolation
                If WSEcheckArray(3, i) <> 0 And WSEcheckArray(2, i) <> 0 Then
                    SumSqDev = SumSqDev + (((WSEcheckArray(3, i) + WSEShift) - WSEcheckArray(2, i)) ^ 2)
                End If
```

```
        Next i

        If currentOptimumShift(1) = -1 Then
            currentOptimumShift(1) = SumSqDev
            currentOptimumShift(0) = WSEShift
        ElseIf SumSqDev < currentOptimumShift(1) Then
            currentOptimumShift(1) = SumSqDev
            currentOptimumShift(0) = WSEShift
        End If

        textOut.WriteLine WSEShift & "," & SumSqDev

        WSEShift = WSEShift + ((WSEmaxShift - WSEminShift) / 30)
    Loop

    'analyze the results of the current optimization spread, and decide whether to loop again with
    'a tighter spread, or to quit with the current answer
    OptimizationLoop = False
    If overallOptimumShift(1) = -1 Then
        overallOptimumShift(0) = currentOptimumShift(0)
        overallOptimumShift(1) = currentOptimumShift(1)
        OptimizationLoop = True
    Else
        If Abs(currentOptimumShift(0) - overallOptimumShift(0)) <= txtOptimizationTolerance.Value Then
            'we have converged on a solution that is acceptable (the elevation didn't change by much on this loop)
            'stop looping
            OptimizationLoop = False
            textOut.WriteLine "Optimal WSE Shift: " & currentOptimumShift(0)

            'write the results back to the Optimized Elevation Field of the shapefile
            For i = 0 To UBound(WSEcheckArray, 2)
                If Not WSEcheckArray(3, i) = 0 Then
                    Set currXSfeat = FeatClass.GetFeature(WSEcheckArray(0, i))
                    currXSfeat.Value(cboOptimizedElevField.ListIndex) = WSEcheckArray(3, i) +
currentOptimumShift(0)
                    currXSfeat.Store
                End If
            Next i


        Else
            overallOptimumShift(0) = currentOptimumShift(0)
            overallOptimumShift(1) = currentOptimumShift(1)
            OptimizationLoop = True
        End If
    End If

    If optimizationIterationCount > 100 Then
        textOut.WriteLine "Optimization Iterations have exceeded 100; aborting the optimization."
        OptimizationLoop = False
    End If

    If OptimizationLoop = True Then
        'set the spread of the next optimization sampling
        WSEminShift = WSEminShift + (0.5 * (currentOptimumShift(0) - WSEminShift))
        WSEmaxShift = WSEmaxShift - (0.5 * (WSEmaxShift - currentOptimumShift(0)))
```

```
            WSEShift = WSEminShift
        End If

    Loop While OptimizationLoop = True


End If


Abort:
Set currXSfeat = Nothing
Set FeatCursor = Nothing
Set FeatClass = Nothing
Set FeatLayer = Nothing

lblStatusMsg.Caption = "Done"
textOut.WriteLine "End of macro execution."

End Sub


Public Sub SelectionSort(ByRef TwoDimensionalArray, SortIndexOfFirstDimension As Long,
AscendingOrDescending As String)
'intended to be an implementation of a selection sort algorithm
'the input TwoDimensionalArray must be set up so that the first dimension has various indexes (columns),
'and the second dimension functions as the rows

Dim i As Long, j As Long, k As Long
Dim cutRecord()
ReDim cutRecord(0 To UBound(TwoDimensionalArray, 1))
Dim cutLoc As Long


For i = 0 To UBound(TwoDimensionalArray, 2)
    'copy the current row
    cutLoc = i
    For k = 0 To UBound(cutRecord)
        cutRecord(k) = TwoDimensionalArray(k, i)
    Next k

    'note:  when we are at the last i value of the array; the following j loop should not execute
    For j = i + 1 To UBound(TwoDimensionalArray, 2)
        Select Case AscendingOrDescending
            Case "Descending"
                If TwoDimensionalArray(SortIndexOfFirstDimension, j) > cutRecord(SortIndexOfFirstDimension) Then
                    cutLoc = j
                    For k = 0 To UBound(cutRecord)
                        cutRecord(k) = TwoDimensionalArray(k, j)
                    Next k
                End If
            Case "Ascending"
                If TwoDimensionalArray(SortIndexOfFirstDimension, j) < cutRecord(SortIndexOfFirstDimension) Then
                    cutLoc = j
                    For k = 0 To UBound(cutRecord)
                        cutRecord(k) = TwoDimensionalArray(k, j)
                    Next k
```

```
            End If
        Case Else
            Debug.Print "Invalid AscendingOrDescending argument passed to SelectionSort subroutine"
    End Select
  Next j
  'place values
  For k = 0 To UBound(cutRecord)
    TwoDimensionalArray(k, cutLoc) = TwoDimensionalArray(k, i)
  Next k
  For k = 0 To UBound(cutRecord)
    TwoDimensionalArray(k, i) = cutRecord(k)
  Next k

Next i

End Sub
```

Appendix D: ArcMap VBA Macro: Moving Average



Screenshot of User Form

Source Code:

```
'Created by Thomas Dickerson (last update: 1/10/2007)
'Required references:  all the usual ESRI libraries

'Note:  US = "upstream", DS = "downstream"

Public pMxDoc As IMxDocument
Public pMap As IMap
Public pActiveView As IActiveView

Private Sub UserForm_Initialize()

Dim i As Long, testFeatLayer As IFeatureLayer

Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
Set pActiveView = pMap

'populate comboboxe(s)
For i = 0 To pMap.LayerCount - 1
   If TypeOf pMap.Layer(i) Is IFeatureLayer Then
      Set testFeatLayer = pMap.Layer(i)
      If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
         cboXsec.AddItem testFeatLayer.Name
      End If
   End If
Next i

Set testFeatLayer = Nothing


End Sub

Private Sub cboXsec_Change()

Dim i As Long, j As Long, testFeatLayer As IFeatureLayer

cboStation.Clear
cboElevationField.Clear
cboAdjustedElevField.Clear
cboSubstitute.Clear

For i = 0 To pMap.LayerCount - 1
   If TypeOf pMap.Layer(i) Is IFeatureLayer Then
      Set testFeatLayer = pMap.Layer(i)
      If testFeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
         If cboXsec.Text = testFeatLayer.Name Then
            For j = 0 To testFeatLayer.FeatureClass.Fields.FieldCount - 1
               cboStation.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
               cboElevationField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
               cboAdjustedElevField.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
               cboSubstitute.AddItem testFeatLayer.FeatureClass.Fields.Field(j).Name
            Next j
```

```
            Exit For
            End If

        End If
    End If
Next i

Set testFeatLayer = Nothing

End Sub

Private Sub cmdRun_Click()

Dim i As Long, j As Long
Dim FeatLayer As IFeatureLayer, FeatClass As IFeatureClass
Dim FeatCursor As IFeatureCursor
Dim currXSfeat As IFeature
Dim FeatSel As IFeatureSelection

Dim WSEarray()
Dim DSforecast()
Dim USforecast()

Dim maxStepsDS As Long, DSvalue As Double
Dim maxStepsUS As Long, USvalue As Double

lblStatusMsg.Caption = "Started"

For i = 0 To pMap.LayerCount - 1
    If TypeOf pMap.Layer(i) Is IFeatureLayer Then
        Set FeatLayer = pMap.Layer(i)
        If FeatLayer.FeatureClass.ShapeType = esriGeometryPolyline Then
            If cboXsec.Text = FeatLayer.Name Then
                Set FeatClass = FeatLayer.FeatureClass
                Exit For
            End If
        End If
    End If
Next i

'------------preliminary checks--------------------
If FeatLayer Is Nothing Or FeatClass Is Nothing Then
    MsgBox "Failed to set layer / feature class; aborting."
    GoTo Abort
End If
If cboElevationField.Value = "" Or cboElevationField.Value = Null Then
    MsgBox "Failed to set Elevation Field; aborting."
    GoTo Abort
End If


'get an array of FID, station, and elevation; loop though this array from US to DS, using a
'forward-looking moving average
'we will make WSEcheckArray have four columns:  0=FID, 1=Station, 2=OriginalElevation, 3=AdjustedElevation
Set currXSfeat = Nothing
Set FeatCursor = Nothing
```

```
ReDim WSEarray(0 To 3, 0 To 0)

'optionally, operate on only the selected features
If chkSelectedOnly.Value = True Then
   'get a cursor of only the selected features
   Set FeatSel = FeatLayer
   FeatSel.SelectionSet.Search Nothing, False, FeatCursor
Else
   Set FeatCursor = FeatClass.Search(Nothing, False)
End If

Set currXSfeat = FeatCursor.NextFeature
Do While Not currXSfeat Is Nothing
   i = UBound(WSEarray, 2)
   WSEarray(0, i) = currXSfeat.OID
   WSEarray(1, i) = currXSfeat.Value(cboStation.ListIndex)
   If currXSfeat.Value(cboElevationField.ListIndex) = txtUnreasonable.Text Then
      'use substitute value instead
      WSEarray(2, i) = currXSfeat.Value(cboSubstitute.ListIndex)
      WSEarray(3, i) = currXSfeat.Value(cboSubstitute.ListIndex)
   Else
      WSEarray(2, i) = currXSfeat.Value(cboElevationField.ListIndex)
      WSEarray(3, i) = currXSfeat.Value(cboElevationField.ListIndex)
   End If

   ReDim Preserve WSEarray(0 To 3, 0 To i + 1)

   Set currXSfeat = FeatCursor.NextFeature
Loop
'chop off the last blank slot
ReDim Preserve WSEarray(0 To 3, 0 To UBound(WSEarray, 2) - 1)
'now we have all the data in the WSEcheckArray
'be sure that the array is sorted in order of increasing station
SelectionSort WSEarray, 1, "Ascending"


If optMoveDownstream.Value = True Then

   '(note that the following procedure assumes that the most upstream elevation is valid)
   For i = UBound(WSEarray, 2) To 0 Step -1

      'we will have an array, "DSforecast", which contains either the slopes or elevations that will figure into the
forecast
      'this will be a 1-based array (contrary to almost all of my other arrays)
      'the second dimension will be used to indicate whether the value is to be included in the averaging
      ReDim DSforecast(1 To txtForwardInclusion.Text, 0 To 1)

      maxStepsDS = UBound(DSforecast, 1)
      If i - maxStepsDS < 0 Then
         Do While i - maxStepsDS < 0
            Debug.Print "Reducing the maxStepsDS..."
            maxStepsDS = maxStepsDS - 1
         Loop
         If maxStepsDS < 1 Then
            Debug.Print "Warning:  we have reached the most DS point, and dont' know what to do"
```

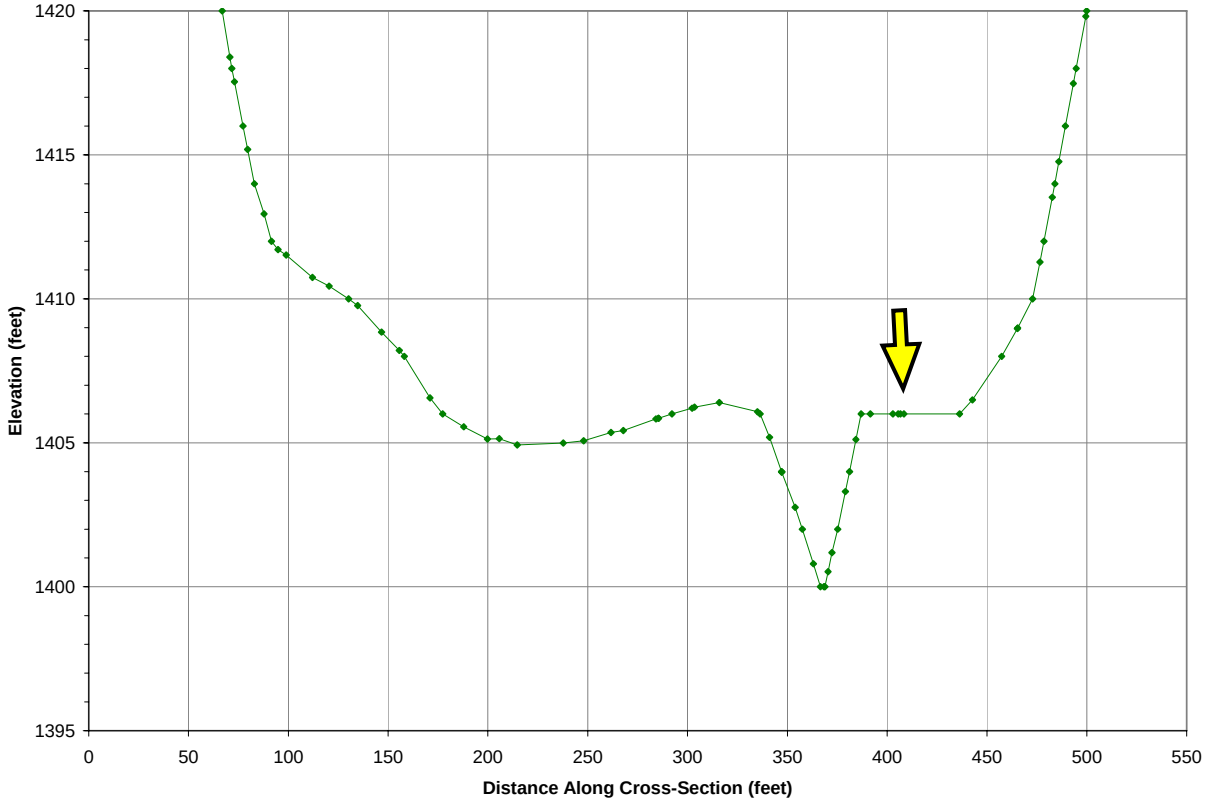
```
        End If
    End If

    If maxStepsDS >= 1 Then
        For j = 1 To maxStepsDS
            If optMethodElev.Value = True Then
                'for elevation method:
                DSforecast(j, 0) = WSEarray(2, i - j)
                'mark whether elevation is valid (or at least, probably valid, based on the idea that since we are in
SWVA, an elevation of zero would not be possible)
                If WSEarray(2, i - j) = 0 Then
                    DSforecast(j, 1) = 0
                Else
                    DSforecast(j, 1) = 1
                End If
            ElseIf optMethodSlope.Value = True Then
                'for slope method:
                '(calculate slope from where were are right now (not necessarily the original elevation value at this
station):
                DSforecast(j, 0) = (WSEarray(2, i - j) - WSEarray(3, i)) / (WSEarray(1, i) - WSEarray(1, i - j))
                Debug.Print "Slope: " & DSforecast(j, 0)
                If WSEarray(2, i - j) = 0 Or WSEarray(3, i) = 0 Or WSEarray(1, i) = 0 Or WSEarray(1, i - j) = 0 Then
                    DSforecast(j, 1) = 0
                Else
                    DSforecast(j, 1) = 1
                End If
            End If
        Next j

        'make the downstream forecast (average of values in the DSforecast array, up to the max steps ds permitted)
        DSvalue = 0
        For j = 1 To maxStepsDS
            If DSforecast(j, 1) = 1 Then
                'value was marked as valid for inclusion
                DSvalue = DSforecast(j, 0) + DSvalue
            End If
        Next j
        DSvalue = DSvalue / maxStepsDS

        'write this value to the DS station
        If optMethodElev.Value = True Then
            'just write the value
            WSEarray(3, i - 1) = DSvalue
        ElseIf optMethodSlope.Value = True Then
            'for slope method: do some math based on our current position
            WSEarray(3, i - 1) = WSEarray(3, i) + (DSvalue * (WSEarray(1, i) - WSEarray(1, i - 1)))
        End If

        'throw in the check to make sure we aren't trying to go uphill moving DS
        If chkRejectNonDS.Value = True Then
            If WSEarray(3, i - 1) > WSEarray(3, i) Then
                'we will just carry the US value DS
                WSEarray(3, i - 1) = WSEarray(3, i)
            End If
        End If
```

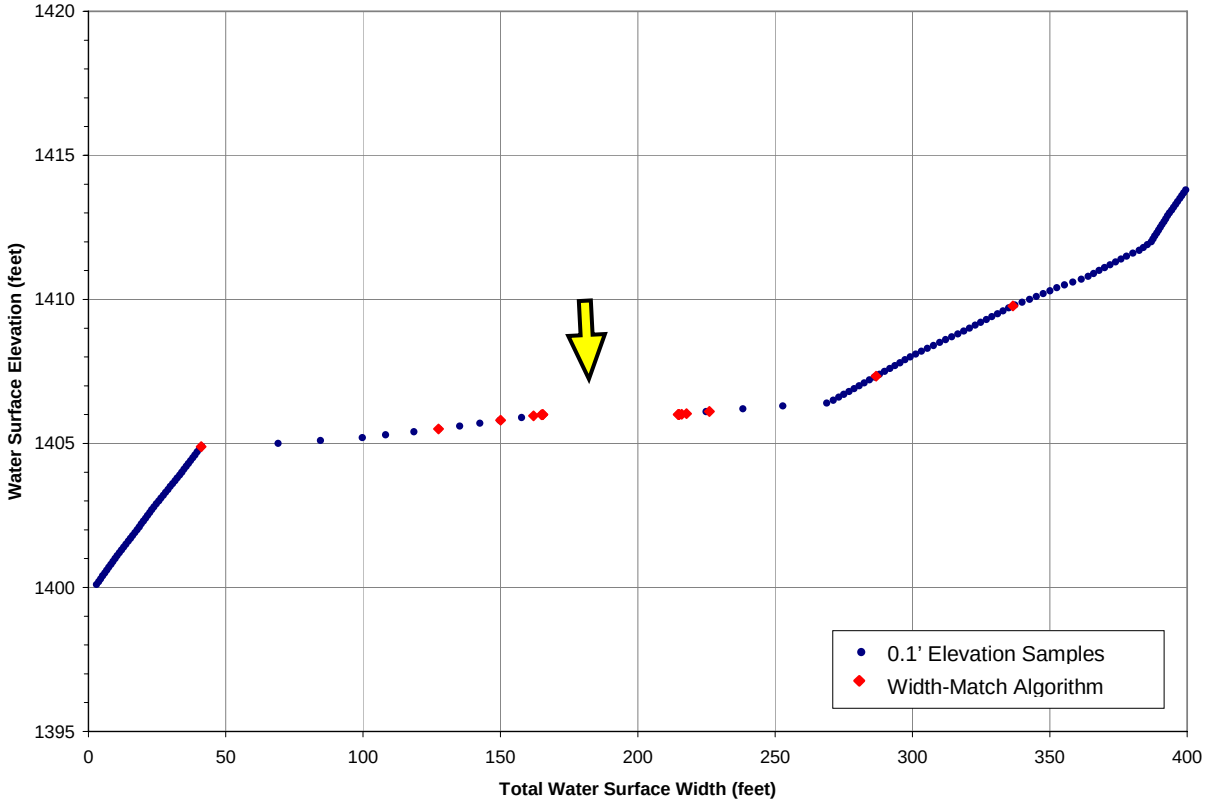
```
      Else '(maxStepsDS < 1)
         'we are already at the most DS point (i=0), and can't forecast downstream any further
         'this is not a problem, just quit (do nothing)
      End If

   Next i

ElseIf optMoveUpstream.Value = True Then

   '(note that the following procedure assumes that the most upstream elevation is valid)
   For i = 0 To UBound(WSEarray, 2)

      'we will have an array, "USforecast", which contains either the slopes or elevations that will figure into the
forecast
      'this will be a 1-based array (contrary to almost all of my other arrays)
      ReDim USforecast(1 To txtForwardInclusion.Text, 0 To 1)

      maxStepsUS = UBound(USforecast, 1)
      If i + maxStepsUS > UBound(WSEarray, 2) Then
         Do While i + maxStepsUS > UBound(WSEarray, 2)
            Debug.Print "Reducing the maxStepsUS..."
            maxStepsUS = maxStepsUS - 1
         Loop
         If maxStepsUS < 1 Then
            Debug.Print "Warning:  we have reached the most US point, and dont' know what to do"
         End If
      End If

      If maxStepsUS >= 1 Then
         For j = 1 To maxStepsUS
            If optMethodElev.Value = True Then
               'for elevation method:
               USforecast(j, 0) = WSEarray(2, i + j)
               'mark whether elevation is valid (or at least, probably valid, based on the idea that since we are in
SWVA, an elevation of zero would not be possible)
               If WSEarray(2, i + j) = 0 Then
                  USforecast(j, 1) = 0
               Else
                  USforecast(j, 1) = 1
               End If
            ElseIf optMethodSlope.Value = True Then
               'for slope method:
               '(calculate slope from where were are right now (not necessarily the original elevation value at this
station):
               USforecast(j, 0) = (WSEarray(2, i + j) - WSEarray(3, i)) / (WSEarray(1, i + j) - WSEarray(1, i))
               Debug.Print "Slope: " & USforecast(j, 0)
               If WSEarray(2, i + j) = 0 Or WSEarray(3, i) = 0 Or WSEarray(1, i) = 0 Or WSEarray(1, i + j) = 0 Then
                  USforecast(j, 1) = 0
               Else
                  USforecast(j, 1) = 1
               End If
            End If
         Next j

         'make the upstream forecast (average of values in the USforecast array, up to the max steps us permitted)
         USvalue = 0
```

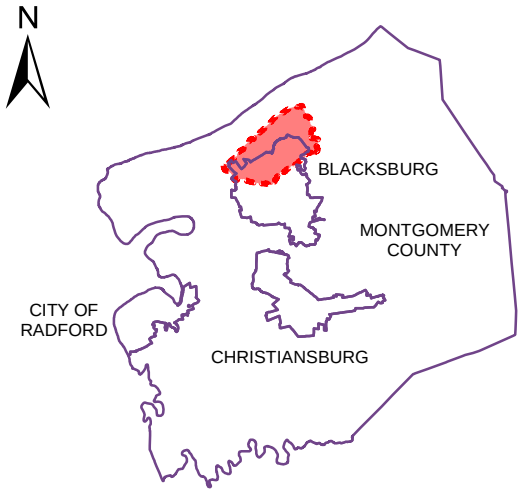
```
        For j = 1 To maxStepsUS
            If USforecast(j, 1) = 1 Then
                'value was marked as valid for inclusion
                USvalue = USforecast(j, 0) + USvalue
            End If
        Next j
        USvalue = USvalue / maxStepsUS

        'write this value to the US station
        If optMethodElev.Value = True Then
            'just write the value
            WSEarray(3, i + 1) = USvalue
        ElseIf optMethodSlope.Value = True Then
            'for slope method: do some math based on our current position
            WSEarray(3, i + 1) = WSEarray(3, i) + (USvalue * (WSEarray(1, i + 1) - WSEarray(1, i)))
        End If

        'throw in the check to make sure we aren't trying to go downhill moving US
        If chkRejectNonDS.Value = True Then
            If WSEarray(3, i + 1) < WSEarray(3, i) Then
                'we will just carry the DS value US
                WSEarray(3, i + 1) = WSEarray(3, i)
            End If
        End If

    Else '(maxStepsDS < 1)
        'we are already at the most US point (i=Ubound), and can't forecast uptream any further
        'this is not a problem, just quit (do nothing)
    End If

  Next i


Else
  'this should not be possible; we will abort
  MsgBox "No direction selected; aborting"
  GoTo Abort

End If

'go through the features and write the adjusted elevations
Set currXSfeat = Nothing
Set FeatCursor = Nothing

For i = 0 To UBound(WSEarray, 2)
  Set currXSfeat = FeatClass.GetFeature(WSEarray(0, i))
  currXSfeat.Value(cboAdjustedElevField.ListIndex) = WSEarray(3, i)
  currXSfeat.Store
Next i


Abort:

Set currXSfeat = Nothing
Set FeatCursor = Nothing
Set FeatClass = Nothing
```

```
Set FeatLayer = Nothing

lblStatusMsg.Caption = "Finished"

End Sub

Public Sub SelectionSort(ByRef TwoDimensionalArray, SortIndexOfFirstDimension As Long,
AscendingOrDescending As String)
'intended to be an implementation of a selection sort algorithm
'the input TwoDimensionalArray must be set up so that the first dimension has various indexes (columns),
'and the second dimension functions as the rows

Dim i As Long, j As Long, k As Long
Dim cutRecord()
ReDim cutRecord(0 To UBound(TwoDimensionalArray, 1))
Dim cutLoc As Long

For i = 0 To UBound(TwoDimensionalArray, 2)
   'copy the current row
   cutLoc = i
   For k = 0 To UBound(cutRecord)
      cutRecord(k) = TwoDimensionalArray(k, i)
   Next k

   'note:  when we are at the last i value of the array; the following j loop should not execute
   For j = i + 1 To UBound(TwoDimensionalArray, 2)
      Select Case AscendingOrDescending
         Case "Descending"
            If TwoDimensionalArray(SortIndexOfFirstDimension, j) > cutRecord(SortIndexOfFirstDimension) Then
               cutLoc = j
               For k = 0 To UBound(cutRecord)
                  cutRecord(k) = TwoDimensionalArray(k, j)
               Next k
            End If
         Case "Ascending"
            If TwoDimensionalArray(SortIndexOfFirstDimension, j) < cutRecord(SortIndexOfFirstDimension) Then
               cutLoc = j
               For k = 0 To UBound(cutRecord)
                  cutRecord(k) = TwoDimensionalArray(k, j)
               Next k
            End If
         Case Else
            Debug.Print "Invalid AscendingOrDescending argument passed to SelectionSort subroutine"
      End Select
   Next j
   'place values
   For k = 0 To UBound(cutRecord)
      TwoDimensionalArray(k, cutLoc) = TwoDimensionalArray(k, i)
   Next k
   For k = 0 To UBound(cutRecord)
      TwoDimensionalArray(k, i) = cutRecord(k)
   Next k

Next i

End Sub
```

Appendix E: Example of Cross-Section Which May Result in No Width-Match Solution







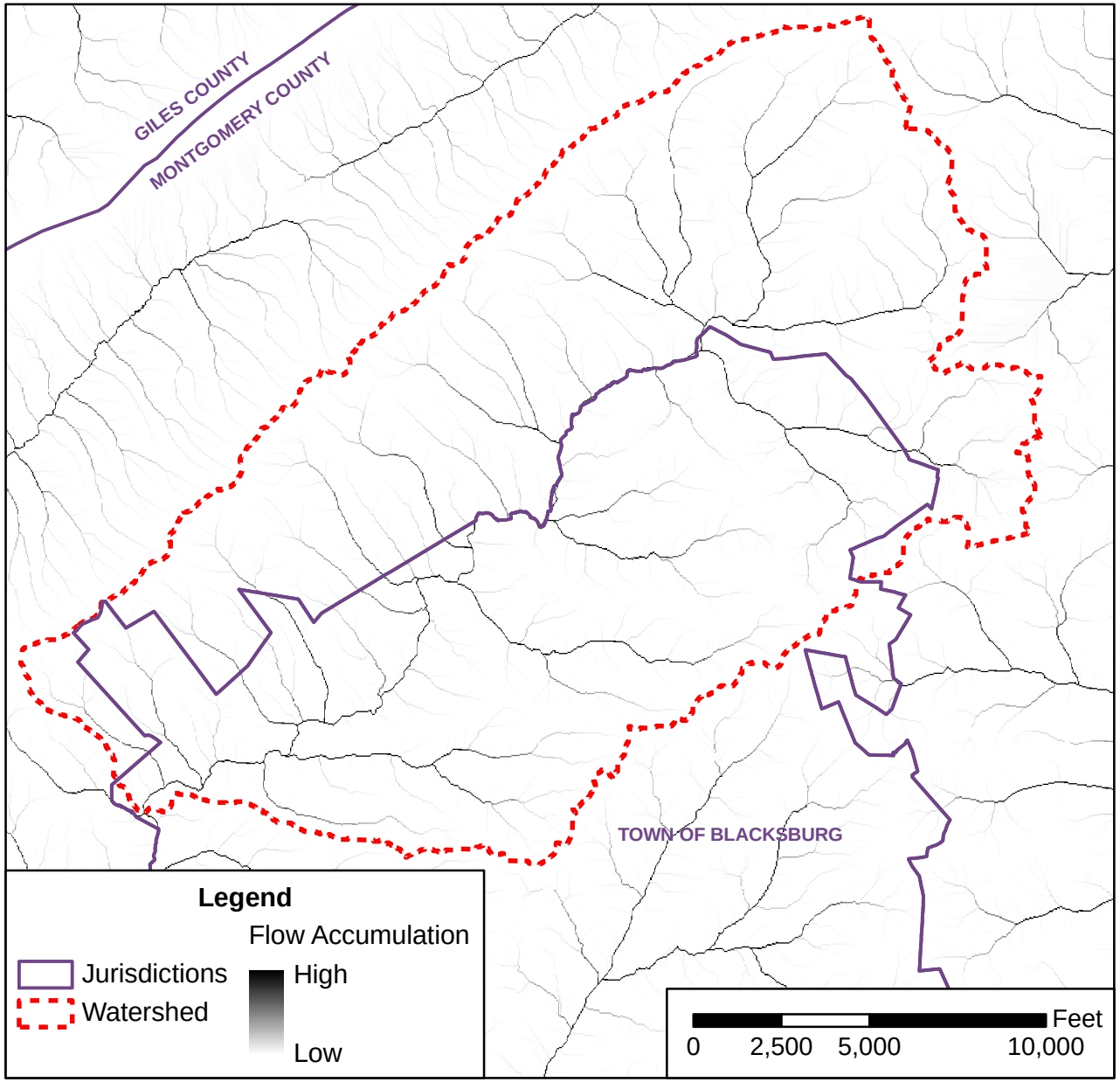
Index Map (1:700,000)

Notes:

Toms Creek flows west from Blacksburg until it joins the New River on the western border of Montgomery County.

The watershed shown on this map in red is delineated to the point where the creek crosses out of Blacksburg and into Montgomery County.

Major flow accumulation pathways are shown in shades of gray and black.

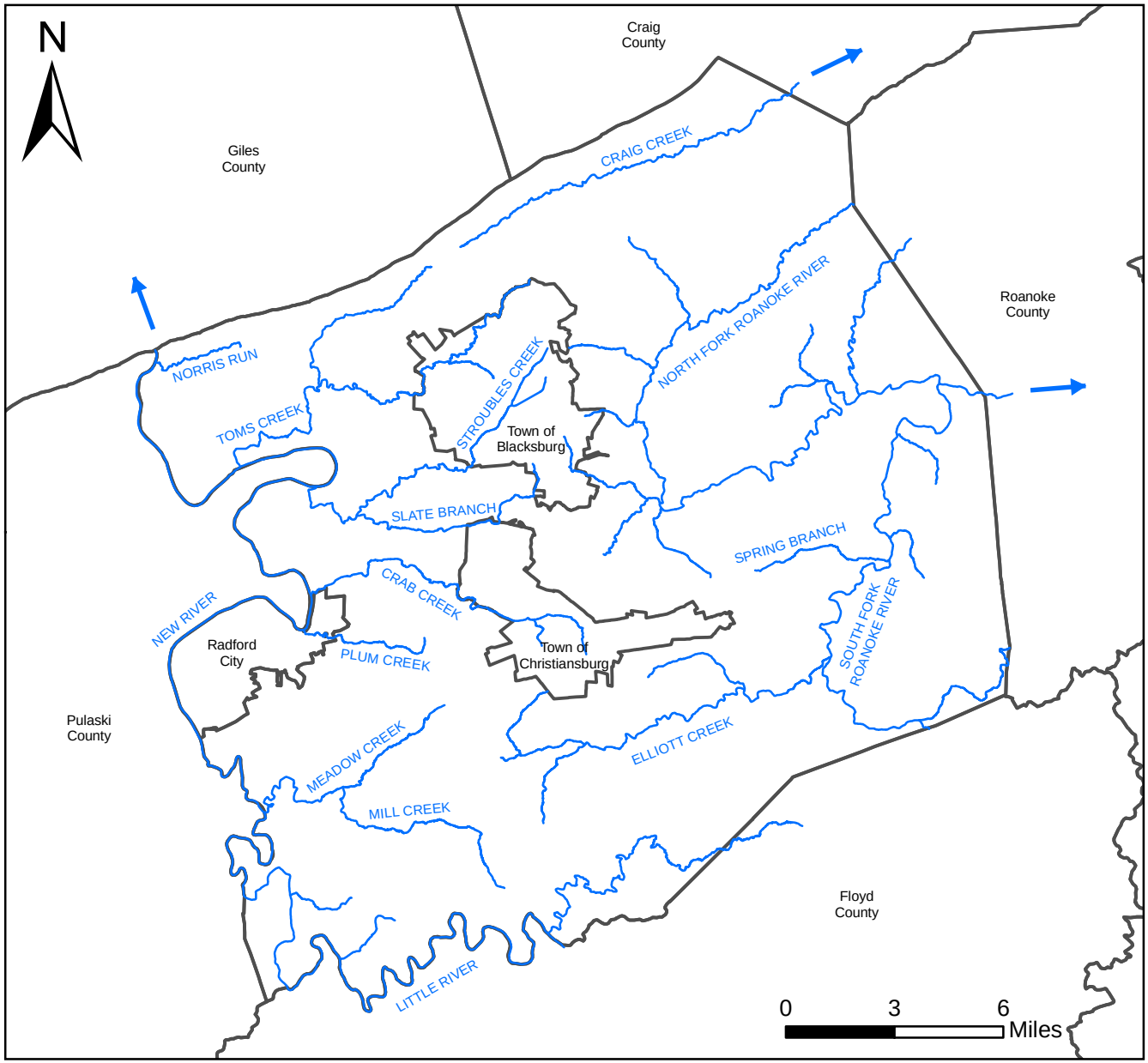


Appendix F: Toms Creek Watershed in Blacksburg, Virginia

Thomas Dickerson, 2007

Flow paths based on USGS 10m NED

Jurisdictional boundaries provided by Montgomery County



Index Map

Streams
Jurisdictions
States

Legend

The New River flows north through Giles County, eventually joining the Mississippi River, flowing into the Gulf of Mexico.

The North and South Forks of the Roanoke River come together and flow east through Roanoke County, and then turns south into North Carolina before joining the Atlantic Ocean.

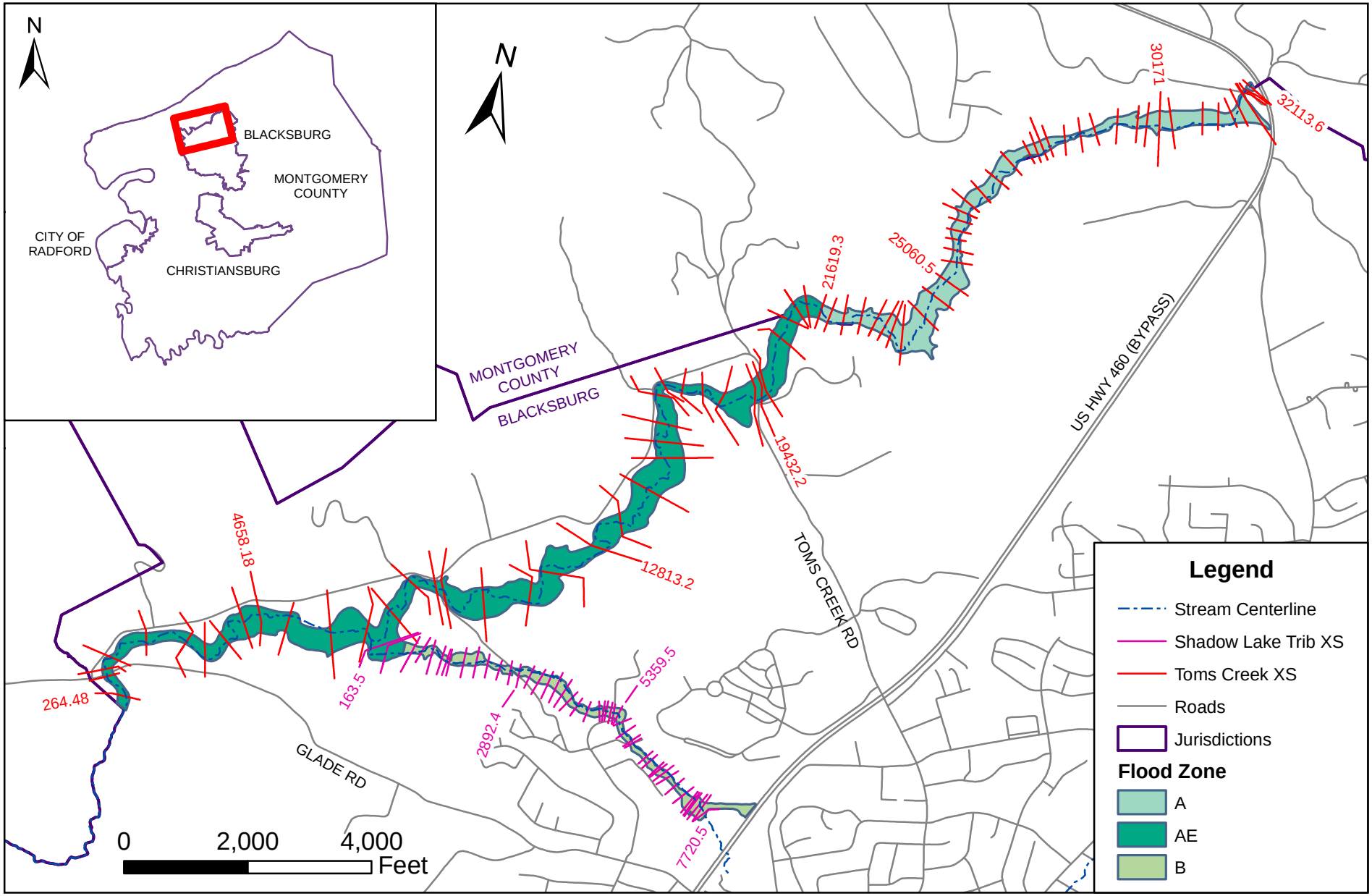
Craig Creek flows northeast through Craig County, joining the James River which discharges into the Chesapeake Bay.

Notes

Appendix G: Streams of Montgomery County, Virginia

Thomas Dickerson, 2007

County boundaries from VBMP2002
State boundaries from U.S. National Atlas
Streams compiled by the Center for Geospatial Information Technology at Virginia Tech

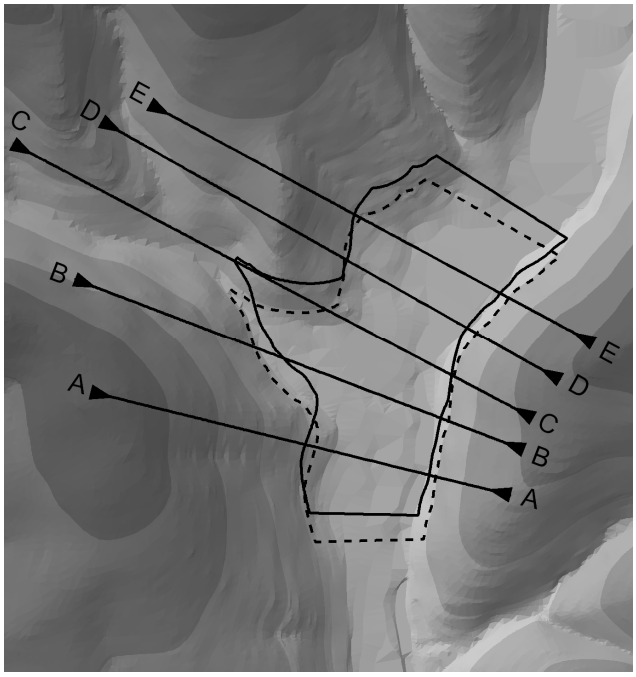


Appendix H: Stream Stationing on Toms Creek and Shadow Lakes Tributary

Thomas Dickerson, 2007

Basemap data provided by Montgomery County

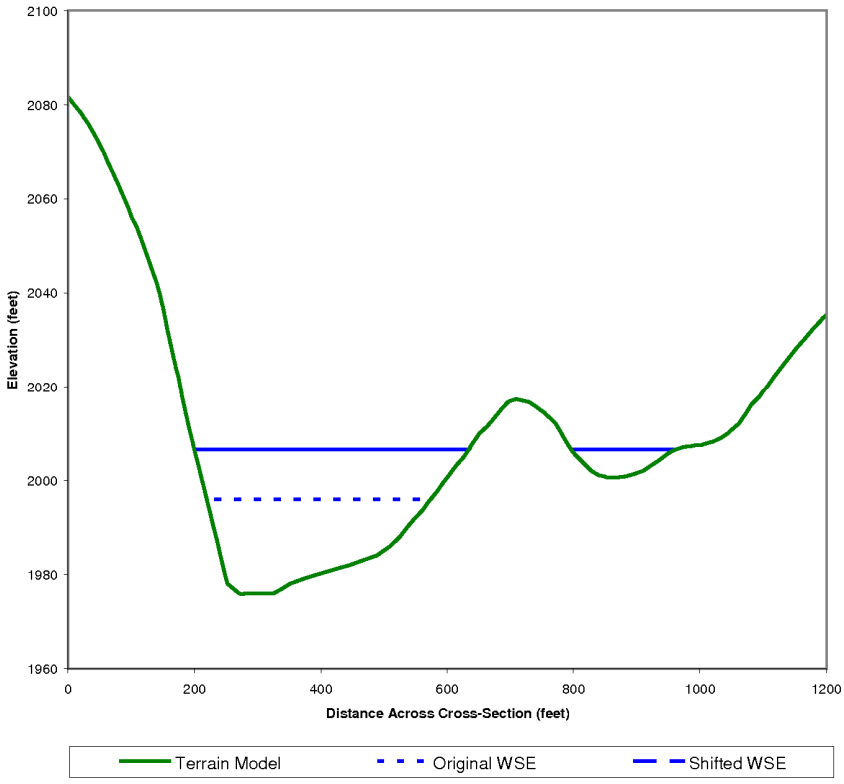
Appendix I: Example of Floodplain Boundaries Incompatible with Terrain Model



The example at right shows an area where a smaller tributary stream on the left joins the main floodplain running vertically through the image. This creates a noticeable triangular extension to the floodplain. The original, accurately delineated floodplain is shown with a dashed line, while a shifted copy of this floodplain is shown with a solid line. The solid line represents the sort of error that could occur due to poor georeferencing, or due to significant errors in the original terrain model.

In this example, the original width of the floodplain at cross-section C is about 350 feet, but the width measured across the shifted floodplain is about 600 feet.

Cross-section C is shown below in profile view. In the cross-sectional view, the dashed lined is the water surface elevation associated with the original, accurately delineated floodplain. The solid line represents the width-match solution that would be obtained if the shifted floodplain were used as the basis for redelineation. Specifically, the width-matching algorithm would search for a water surface elevation resulting in a floodplain width of 600 feet. As shown, the error associated with the shifted floodplain is significant.



Provided that such errors are rare, their effects may be handled adequately by the profile smoothing algorithm described in this thesis.