Intelligent Decimation of River Geometry Data for Manageable Use in Surface-Water Models

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
with a
Major in Civil Engineering
in the
College of Graduate Studies
University of Idaho
by
Charles Berenbrock

Major Professor: Peter Goodwin, Ph.D.
Committee Members: Nigel Wright, Ph.D.; Fritz Fiedler, Ph.D.; Terry Soule, Ph.D.
Department Administrator: Patricia Colberg, Ph.D.

May 2018

Authorization to Submit Dissertation

This dissertation of Charles Berenbrock, submitted for the degree of Doctor of Philosophy with a major in Civil Engineering and titled "Intelligent Decimation of River Geometry Data for Manageable Use in Surface-Water Models," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor ______________________________ Date______________
Peter Goodwin, Ph.D.

Committee
Members ______________________________ Date______________
Nigel G. Wright, Ph.D.

______________________________ Date______________
Fritz R. Fiedler, Ph.D.

______________________________ Date______________
Terrence Soule, Ph.D.

Department
Administrator ______________________________ Date______________
Patricia Colberg, Ph.D.

Abstract

Two genetic algorithms (GA) for reducing river geometry data are presented. These algorithms effectively remove “redundant” and/or “nonessential” points from large datasets. The resulting smaller, less dense datasets makes the information more manageable and easier to work with. The first genetic algorithm reduces stream channel cross section data, and the second reduces bathymetry/LiDAR data.

The cross-section genetic algorithm was used to reduce stream channel cross section data. A hypothetical example consisting of 41 data points and 10 cross sections on the Kootenai River in northern Idaho were reduced. Cross sections from the Kootenai River that are representative of meander, straight, braided, and canyon reaches were used to evaluate the reduction methods. The number of data points for the Kootenai River cross sections ranged from about 500 to more than 2,500. Results indicated that the genetic algorithm successfully reduced the data. However, the original genetic algorithm does not account for varying distances between the data points. To account for irregularly-spaced data, the fitness function was modified and used in subsequent analyses. Fitness values from the modified genetic algorithm were lower (better) than in the original genetic algorithm and those that used the standard method of reducing cross-section data. Visual and hydraulic analyses were also used to assess the methods. The genetic algorithm reduced cross sections approximated the shape of the original cross sections better than the standard-reduced cross sections. Also a greater number of cross-sectional data points were needed for reduced cross sections in the straight reach and even more in the meander reach because a greater amount of data points are needed to adequately define cross sections that have greater topographic variability.

The effects of reduced cross-sectional data points on steady flow profiles were also analyzed. A portion of the original steady-flow model of the Kootenai River was used, consisting of thirty-five cross sections. These cross sections were reduced to 10, 20, and 30 data points by the standard and modified genetic algorithm methods, that is, six test were completed for each of the thirty-five cross sections. Differences were smaller for reduced cross sections developed by the genetic algorithm (modified) method than the standard algorithm method. Generally, differences from the original water-surface elevation were smaller as the number of data points in reduced cross sections increased, but not always, especially in the braided reach.

A genetic algorithm to decimate bathymetry and Light Detection and Ranging (LiDAR) datasets was also developed. These datasets can be used in two- and three-dimensional surface-water models. A hypothetical example consisting of 961 regularly spaced data points (x, y, and z) and data taken from an actual bathymetric and LiDAR dataset (10,080 data points) were reduced. Results indicated that the genetic algorithm successfully reduced the data. Terrains produced by the genetic algorithm are fairly representative of the original data and had smaller differences (better) than standard procedures of decimating LiDAR. Hypsometric curves of volume between the GA runs and original dataset were quite similar while the curves from standard reduction methods were quite different than the original.

Other x-y data also can be reduced in a method similar to that for cross section data. Also the LiDAR/bathymetric genetic algorithm should decimate equally as well on any terrain data that is expressed in x, y, and z coordinates.

Acknowledgements

I would like to express my gratitude to everyone who supported me throughout this lengthy journey, from concept to completion of the doctorate. First off, I would like to express my deep appreciation and gratitude to my advisor, Dr. Peter Goodwin, for his patience, encouragement, and support throughout the dissertation. I would also like to thank my committee members Dr. Nigel Wright, Dr. Terrence Soule, and Dr. Fritz R. Fiedler for reading through the dissertation and providing helpful comments. In particular, Dr. Terrence Soule provided invaluable guidance in the topics of genetic algorithms. Also I would like to thank Dr. Patricia Colberg, Department Administrator, for the support and encouragement she provided.

Thanks to Maureen and Paulette, my children, their husbands, Brian and Robert, respectively, and to my grandchildren Sawyer, Evelyn, Colette, Carlene, Aurelia, Brice, and Natalia for putting up with all the “inconveniences” of having a Dad and Grandpa who frequently had to study. God has richly blessed me with all of you and I feel very humbled.

Many, many thanks to my wife Marcia—there are too many things to thank you for, suffice to say you deserve this degree as much as I do. You are “truly” my best cheerleader and number one encourager. You have persevered with me every step of the way. You had the steadfast determination to purse this dream “for the Glory of God” when I wanted to give up. You are the love of my life!

Dedication

To Marcia…

An excellent wife, who can find? For her worth is far above jewels. Proverbs 31:10

To Maureen, Paulette, Brian, Robert, Sawyer, Evelyn, Colette, Carlene, Aurelia, Brice, and Natalia

Behold, children are a gift of the Lord……….How blessed is the man whose quiver is full of them…………Psalms 127:3-5

"To God be the Glory"

Table of Contents

List of Figures

List of Tables

CHAPTER 1. INTRODUCTON

The size of digital datasets can be quite large, and as technology advances, the size in digital data usually increases too. Large datasets can cause numerous problems especially in storing, handling, transmitting, and with software. Data reduction is commonly applied to large datasets. For bathymetric and (or) Light Detection and Ranging (LiDAR) datasets, for example, it is critical that the bathymetry/terrain not be altered when points or data are removed. By decimating intelligently, large datasets can be reduced to a manageable size for surface-water models and other applications while maintaining the original geometry. This is quite important for the accuracy of surface-water models.

Surface-water hydraulic models require accurate representation of the river and (or) floodplain geometry. Accuracy of geometry is important because it could affect channel-geometry determinations and water-surface calculations, which consequently has major effects on computations of velocity, shear stress, and sediment transport. For one-dimensional surface-water models, cross-sectional data are needed and defined by a series of data points (distance and elevation) along a straight line roughly perpendicular to streamflow. For several large rivers in northern Idaho, bathymetry data for approximately 500 cross sections were collected using a global positioning system (GPS) with an echo sounder, and bank data were collected by connecting the GPS to a laser range finder equipped with an angle encoder (Moran and Berenbrock, 2003; Barton *et al*, 2004; and Berenbrock and Tranmer, 2008). The echo sounder obtained data at very close intervals, usually exceeding 500 data points, whereas about 10 bank data points were collected at each cross section. The number of data points in each cross section ranged from about 500 to more than 2,000 which is too large for most one-dimensional models. Selecting the appropriate data points among the hundreds or thousands of data points can be both challenging and tedious.

For multi-dimensional surface-water models, data such as bathymetry and (or) LiDAR are needed. Bathymetry and (or) LiDAR datasets are usually quite large. For example, a LiDAR dataset on the lower Coeur d'Alene River for a 2 kilometer (km) by 2 km area consist of about 300,000 data points (x, y, and z). If a 10 km x 2 km reach of the river and floodplain were selected to be modeled, the dataset would be composed of about 3 million data points which is too large for multi-dimensional models. Selecting the appropriate data points among the millions of data points can be both challenging and tedious and also time consuming. Standard procedures usually consist of gridding which generalizes—misses high and low points—the terrain. Another disadvantage to gridding is that the original data points might not be honored in the grid.

For flood insurance studies, the Federal Emergency Management Agency (FEMA) indicates that cross-section points should be located at breaks in the ground slope and should approximate the actual shape of the channel and (or) floodplain (FEMA, 1995). There is no point minimum as long as the actual shape of the channel and floodplain are well defined. The FEMA requirement applies to cross-section data, but is a reasonable requirement for multi-dimensional datasets such as digital elevation models (DEM) and bathymetric and LiDAR datasets.

The purpose of this dissertation is to describe application of several GAs for decimating river geometry data for manageable use in surface-water models and demonstrate that GAs are a viable approach. The first GA is for reducing the number of data points in a cross section, and the second GA is for decimating bathymetry and (or) LiDAR data. This dissertation presents the development, testing, comparisons, and 'real world' application of the GAs. Also the cross-section GA is evaluated to determine its effects of reduced cross sections on channel geometry and steady-flow profiles, and the bathymetry/LiDAR GA is evaluated to determine its effects on decimated bathymetry/terrain data. These evaluations are presented in this dissertation. Also spectral analysis will be used to investigate the spectral content of cross-section data for different channel types and for different scales of resolution.

Chapters 2 and 3 of this dissertation discuss the development, application, and evaluation of the cross-section genetic algorithm (GA). Chapter 2 is found in Berenbrock (2006), and permission is granted by the publisher (John Wiley & Sons) to publish it in this dissertation (see Appendix A). This GA, however, did not account for irregularly spaced data, and thus, it was modified and with additional evaluation is presented in Chapter 3. This chapter (3) is found in Berenbrock (2015) and is considered a public domain report, which does not require permission to publish (see Appendix B). Chapter 4 discusses the development, application and evaluation of the bathymetry and (or) LiDAR GA. This chapter (4) is found in Berenbrock (2010) and is also a public domain paper, which does not require permission to publish (see Appendix C).

Supporting materials for this dissertation are given in the appendices. The following is a short description of each appendix:

Appendix

A. Permission by publisher (John Wiley & Sons) to publish paper in Chapter 2 in dissertation.
B. The published report in Chapter 3 is in public domain and does not require permission to publish.

C. The published paper in Chapter 4 is in public domain and does not require permission to publish.
D. A generalize description of the cross-section genetic algorithm code including how to compile and run the code.
E. A listing of the computer code (main and subroutines) for the cross-section genetic algorithm.
F. Listing of the hypothetical example used for the cross-section GA.
G. Computer listing of outputs from running the hypothetical example.
H. A generalize description of the bathymetric and LiDAR genetic algorithm code including how to compile and run the code.
I. A listing of the computer code (main and subroutines) for the bathymetry and LiDAR genetic algorithm.
J. Listing of hypothetical LiDAR data used for the bathymetry/LiDAR genetic algorithm.
K. Computer listing of output from running the hypothetical LiDAR data.
L. Permission by SNLA to use the isort code.
M. Permission by author to use the locpt code.
N. Permission by GNU LGPL to use the GEOMPACK code.

1.1 References

Barton, G.J., E.H. Moran, and Charles Berenbrock, 2004. Stream Channel Cross Sections for the Kootenai River Between Libby Dam, Montana, and Kootenay Lake, British Columbia, Canada. U.S. Geological Survey Open-File Report 2004-1045, Boise, Idaho. p. 35.

Berenbrock, Charles, 2010. Decimation of River Geometry Datasets with Integrity for Use in Surface-Water Models. Proceeding of the Second Joint Federal Interagency Conference on Sedimentation and Hydrologic Modeling 2010, Fourth Federal Interagency Hydrologic Modeling Conferences, [p. 14].

Berenbrock, Charles, 2015. Reducing cross-sectional data using a genetic algorithm method and effects on cross-section geometry and steady-flow profiles. U.S. Geological Survey Scientific Investigations Report 2015–5034, p. 16.

Berenbrock, Charles, and A.W. Tranmer, 2008. Simulation of flow, sediment transport, and sediment mobility of the Lower Coeur d'Alene River, Idaho. U.S. Geological Survey Scientific Investigations Report 2008–5093, p. 164.

FEMA (Federal Emergency Management Agency), 1995. Guidelines and Specifications for Study Contractors. Federal Emergency Management Agency, Publication 37. U.S. Government Printing Office, Washington, D.C., p. 174.

Moran, E.H. and Charles Berenbrock, 2003. GPS – Time Saver and Functional. U.S. Geological Survey Western Water Watch 1(1): 6-7.

CHAPTER 2. A GENETIC ALGORITHM TO REDUCE STREAM CHANNEL CROSS SECTION DATA

2.1 Abstract

A genetic algorithm (GA) was used to reduce cross section data for a hypothetical example consisting of 41 data points and for 10 cross sections on the Kootenai River. The number of data points for the Kootenai River cross sections ranged from about 500 to more than 2,500. The GA was applied to reduce the number of data points to a manageable dataset because most models and other software require fewer than 100 data points for management, manipulation, and analysis. Results indicated that the program successfully reduced the data. Fitness values from the genetic algorithm were lower (better) than those in a previous study that used standard procedures of reducing the cross section data. On average, fitnesses were 29 percent lower, and several were about 50 percent lower. Results also showed that cross sections produced by the genetic algorithm were representative of the original section and that near-optimal results could be obtained in a single run, even for large problems. Other data also can be reduced in a method similar to that for cross section data.

2.2 Introduction

Cross sections are used to describe the channel shape of streams and most commonly are used in mathematical computer models to simulate flow hydraulics and sediment transport in a stream. A stream channel cross section is a series of data pairs (distance and elevation) along a straight line that is roughly perpendicular to streamflow. During 2002 and 2003 stream channel cross sections and longitudinal data were collected along the Kootenai River from Libby Dam, Montana, to where the river empties into Kootenay Lake near Creston, British Columbia, Canada (study area, Figure 2.1). About 250 kilometers of streambed and banks along the Kootenai River in the study area were mapped on the basis of approximately 400 cross sections (Moran and Berenbrock, 2003). Of these cross sections, only 245 were needed for use in one-dimensional hydraulic flow and sediment transport models of the Kootenai River.

For the Kootenai River, cross section data are a combination of bathymetric and bank data. Bathymetric data collection involves interfacing global positioning system (GPS) equipment with an echo sounder, and bank data are collected by connecting the GPS to a laser range finder equipped with

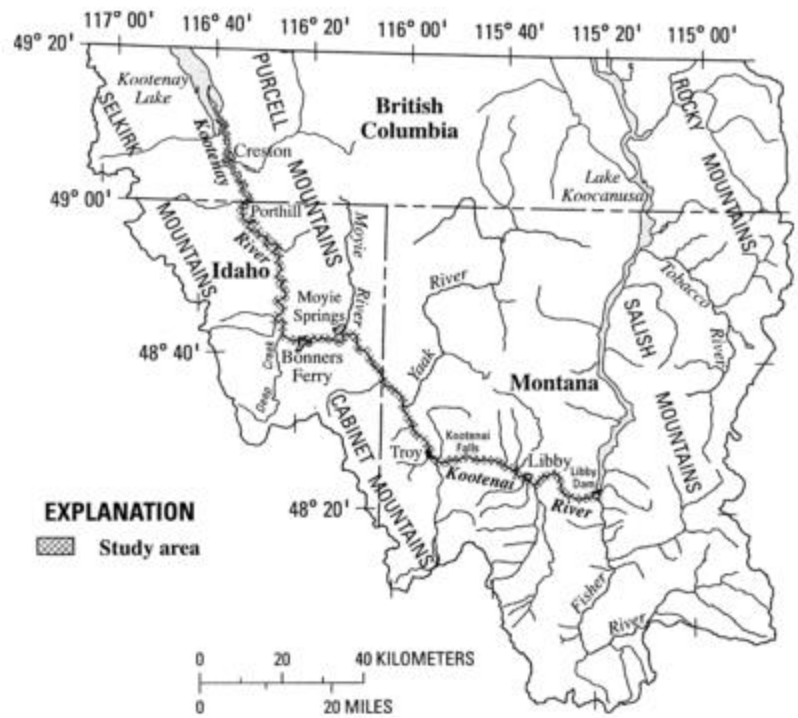


Figure 2.1. Location of the Kootenai River study area.

an angle encoder. The echo sounder obtains data at very close intervals along a section, and the number of soundings in a cross section usually exceeds 500 points, whereas about 10 bank data points are collected at each cross section. Most of the 245 cross sections have more than 1,000 data points, and about one-fourth have more than 2,000. Only a few cross sections on the Kootenai River have more than 2,500 data points.

These large datasets present a problem for use in models and software designed for data management and manipulation. Most models require fewer than 100 data points per stream channel cross section; datasets with fewer than 100 data points also are much more manageable and easier to use. For flood insurance studies, the Federal Emergency Management Agency (FEMA) has indicated that cross section points should be located at breaks in the ground slope and should approximate the actual shape of the channel and (or) floodplain (FEMA, 1995). This indicates that there is no point minimum as long as the actual shape is well defined. In a recent study, Barton *et al*. (2004) used standard procedures to reduce cross-section data by selecting a data pair every one to two meters. Because this selection process frequently misses high and low points in the data, the reduced dataset was viewed graphically, and data were added until the reduced dataset appeared to be representative of the original cross section. This procedure was done for each of the 245 cross sections; this took about a month of labor intensive work to complete.

Many water resource problems have been solved using optimization, especially in water supply and distribution and ground water remediation problems. Optimization techniques such as linear programming, nonlinear programming, and dynamic programming have been used widely in water resources. Vink and Schot (2002) and Chen (2003) discuss the advantages, disadvantages, and appropriateness of these techniques. However, they indicate that GAs are capable of handling highly nonlinear, discontinuous, nondifferentiable, interdependent, and nonconvex problems where these other techniques cannot. Genetic algorithms have been used to solve many optimization problems, and applications in water resources are becoming more abundant. McKinney and Lin (1992) applied a single objective optimization GA to the development of a well field and aquifer remediation. Cieniawski *et al*. (1995) applied a two-objective optimization GA to ground water monitoring, and Vink and Schot (2002) applied a multi-objective optimization to a multiple well production with interdependent and nonlinear impacts. Chen (2003) applied a real coded GA to optimize rule curves for a reservoir in a water supply network system. Knaapen and Hulscher (2003) applied a GA to determine the shape, migration, and boundary variables of alternate bars from bathymetry data for input into a streambed evolution model.

The purpose of this paper is to describe applications of a GA to the reduction of cross section data and prove that the GA is successful and can complete the task within a reasonable amount of time. The GA in this paper is a single objective optimization for reducing cross section data. A hypothetical example and case study with varying amounts of cross section data are presented to validate the genetic algorithm. The hypothetical example is a cross section composed of 41 data pairs. The case study consists of data from 10 cross sections on the Kootenai River.

2.3 Genetic Algorithms

Genetic algorithms apply the principles of evolution to find the solution to a problem. They are based loosely on Darwin's theory of evolution, “survival of the fittest,” and use genetic operators such as selection, reproduction (crossover), and mutation to improve a population. Holland (1975) was the first to apply the operators of selection and mutation in a computer program. Presently, GAs have been applied successfully to many water resource problems.

In a GA, an initial population of individuals is created and is evolved until a solution is obtained or a user-specified number of generations has been met. Each individual is initially a random solution to the problem. Being random, the solutions or individuals may or may not be accurate. The fitness of an individual is a measure of its accuracy, which guides the GA. Greater accuracy increases the chances that the individual will be selected to reproduce using genetic crossover and mutation. Crossover is a random process that exchanges chromosomes between the parents to create the offspring (children). For example, two individuals `100010` and `111111` are crossed between the third and fifth elements to form two children, `10`**`111`**`0` and `11`**`001`**`1`. Mutation randomly changes some of the chromosomes in the individual. For example, the child `100110` might be mutated in the fourth element to form `100`**`0`**`10`. Some children will be better fit than the parents and some will be worse. By repetition of this process and selection of individuals with better fits in subsequent generations, the population improves. A more complete discussion on genetic algorithms is given in Grefenstette (1986), Goldberg (1989), Davis (1991), and Mitchell (2002).

2.4 Program Description

A binary GA program was written to reduce the amount of data pairs in a cross section. Binary strings of a fixed length in the program are used to represent individuals in the population, where the fixed length is equivalent to the number of data pairs (n) in the original stream channel cross section. A 0 bit represents exclusion of that particular data pair on that cross section, and a 1 represents inclusion. Also, the first and last elements are fixed to 1 (included) in all individuals to ensure that endpoints are maintained.

A limit on the number of data points is set, creating a two-conditional fitness function,

$$f(i) = \begin{cases} \sum_{j=1}^{n} d_j & \text{if } inc_i \le plimit \\ \left[\left(\sum_{j=1}^{n} d_j\right) + 1\right] \cdot 10^{(inc_i - plimit)} & \text{otherwise} \end{cases} \qquad \text{(Equation 2.1)}$$

where f(i) is the value of fitness for individual i, n is the number of data pairs in the original cross section, d_j is the vertical distance (m) between the original and reduced data pairs in the cross section for element j, inc_i is the number of included data pairs in the reduced cross section for individual i, and plimit (point limit) is the maximum number of data pairs to be included in the reduced cross section.

The first condition applies if an individual's sum of inclusions (1 bits) is less than or equal to the point limit; otherwise, the second condition applies. Fitness for the first condition is the sum of the distances between the original and reduced data pairs. At included data pairs, vertical distance (d_j) is zero. At excluded pairs, d_j is calculated by subtracting the elevation from the original data pair to an elevation on a straight line derived from two adjacent included data pairs. The vertical distance is always a positive value.

Equation 2.1 gives the single objective function for optimization. The GA minimizes the sum of distances between the original and reduced cross sections or fitness in Equation 2.1 (*Minimize* f(i)) for quantifying the optimal reduced cross section from all possible cross sections.

Figure 2.2 shows a sample calculation of fitness. Included pairs occur at Elements 1, 6, 7, and 8, and d_j for those points is 0. Elevation at excluded Elements 2, 3, 4, and 5 is calculated from a straight line connecting Elements 1 and 6 (included), and d_j is calculated to be 0 m, 10 m, 15 m, and 15 m, respectively. At Element 2, elevations from the original data pair and from the straight line are the same, which results in d_j equal to 0. Fitness for the second condition of Equation 2.1 is more complex and requires the same fitness calculation as in the first. The plus one expression prevents a 0 fitness, which will occur if all data pairs are included. The second half of this condition penalizes fitness because an individual's inc_i exceeds the point limit. The function also causes fitness to increase beyond the point limit.

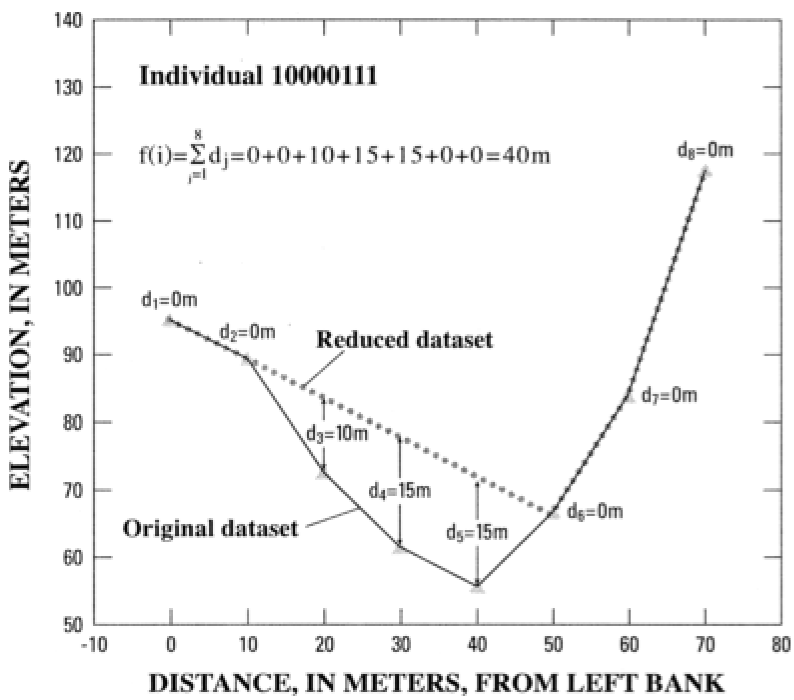


Figure 2.2. Fitness calculation for individual 10000111.

Figure 2.3 shows fitness results of a population by using Equation 2.1. Individuals with inclusions greater than the point limit have much higher (worse) fitnesses. The penalty to fitness is quite significant (Figure 2.3), which causes the GA to perform better by increasing more highly fit (lower fitness value) individuals in the population.

In most binary GAs, each bit has a 50 percent chance of being excluded (0) or included (1) during creation. Thus, on average, half of the bits in an individual will be 0s and the other half will be 1s. In these experiments, half of n is usually greater than plimit. Therefore, the probability of an element being included is set to the plimit divided by string length (n). This greatly reduces the number of generations and program run time.

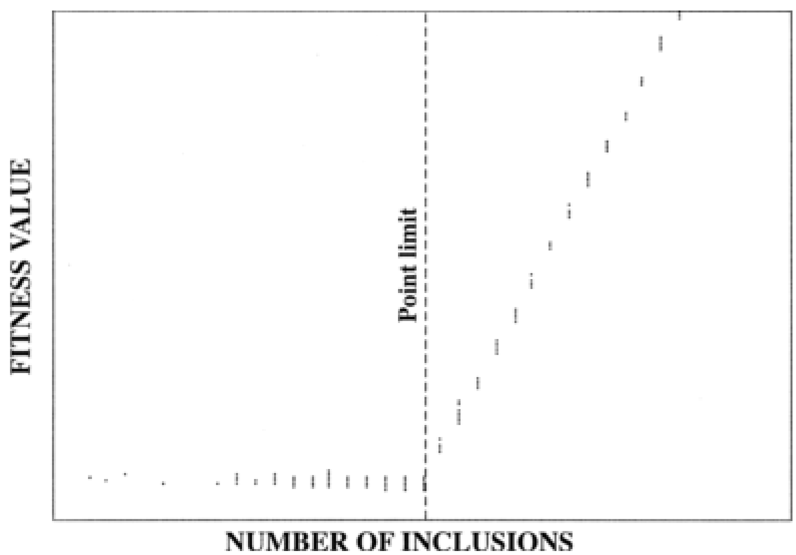


Figure 2.3. Individual fitness and inclusions of a population and point limit.

This initialization technique might reduce the search space and increase premature convergence. However, the reduced search space is still large relative to the population size. For example, the search space for the hypothetical example (discussed later in the paper) with a probability of an element being included of 50 percent (N/2) is 1.68×10^{29}. For a reduced probability of 33 percent, the search space is 5.06×10^{19}. This search space is still quite large relative to the population size and should not significantly affect convergence. By narrowing the search space, the GA then focuses its search on approximately the right number of points and does not explore unlikely solutions (including all points, including no points, and so on).

The GA is generational; two elite individuals are copied each generation. Tournament selection is used as the reproduction method, and tournament size is 3. Several different crossover rates (P_c), population sizes, and mutation rates (P_m) were tested. The mutation rate of 1/n was used, a standard rate suggested by Reed *et al*. (2000, 2003). A general description of the genetic algorithm program for cross-section reduction is given in Appendix D, and a listing of the program code is given in Appendix E.

2.5 Program Validation

The hypothetical and Kootenai River stream channel cross section data were used to validate the binary GA. The plimit for the hypothetical example was arbitrarily set to 15 and to the number of points selected by Barton *et al*. (2004) for the Kootenai River cross sections. On the basis of work by Reed *et al*. (2000, 2003) and Mitchell (2002), a P_c of 0.70 was used for the hypothetical data.

2.5.1 Hypothetical Example

The hypothetical cross section (Figure 2.4) consists of 41 data pairs (distance and elevation) (see Appendix 3 for the x-y data pairs). The GA was run 10 times with a population of 200, for 100 generations, using a crossover rate of 0.70 (Reed *et al*., 2000, 2003; Mitchell, 2002) and a plimit of 15 points. Results from the runs are shown in Table 2.1. The best fitness in all runs had 15 inclusions (the plimit), and Run 9 had the lowest (best) fitness, 1.572 m. Even though the GA relies on randomness in sampling and in creating the initial population, the range in best fitness for all runs was small compared with the ranges in average fitness and root mean squared error (RMSE). The small range in best fitness suggests that near optimal results could be obtained in a single GA run. The large range in average and RMSE fitnesses is indicative of the randomness of the GA and diversity in the population. The reduced cross section from the best fit individual in Run 9 was superimposed on the original cross section (Figure 2.4) and the result indicates that the GA reduced dataset closely represents the original.

The best, average, and RMSE fitnesses for Run 9 are shown in Figure 2.5. Initially these values decreased, but after 10 generations, average and RMSE fitnesses oscillated while best fitness continued to decrease. Similar results were observed in the other nine runs.

The run time for this example was extremely fast, less than 1 second (Table 2.1). These runs were performed on a 450-MHz personal computer (PC). An example output listing from the cross-section reduction program for the hypothetical cross section is given in Appendix 4.

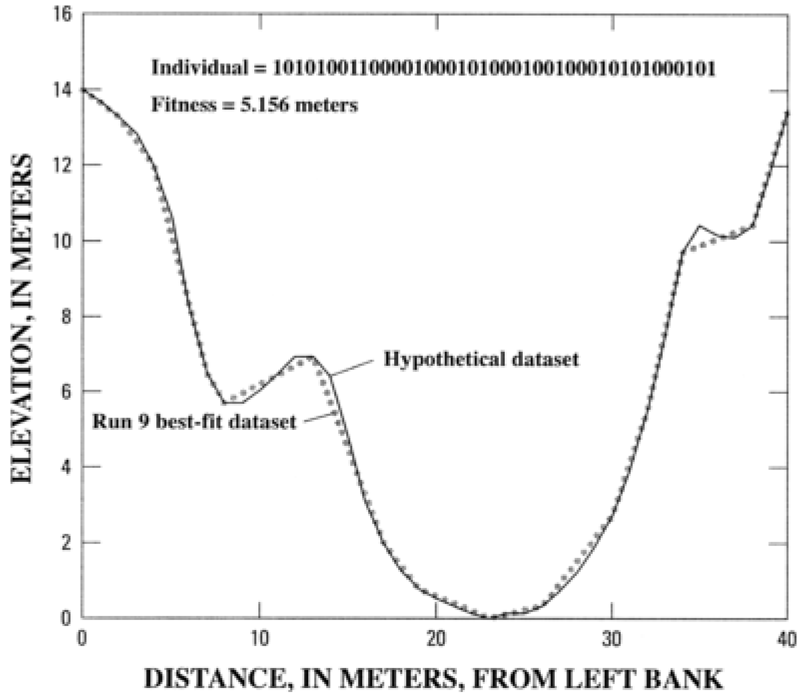


Figure 2.4. Hypothetical cross section and run 9 best-fit cross section.

Table 2.1. Fitness values, run time, and number of included points for the hypothetical example.

[RMSE, root mean squared error]

Run No.	Run Time (seconds)	Number of Included Points	Fitness (meters) Best	Average	RMSE
1	0.4707	15	1.648	187	1,380
2	0.4607	15	1.755	91	397
3	0.5007	15	1.780	116	479
4	0.4306	15	1.977	583	4,204
5	0.4607	15	1.928	62	332
6	0.4807	15	1.877	130	529
7	0.4807	15	1.709	203	1,996
8	0.4607	15	1.712	92	415
9	0.4807	15	1.572	148	646
10	0.4707	15	1.773	47	244

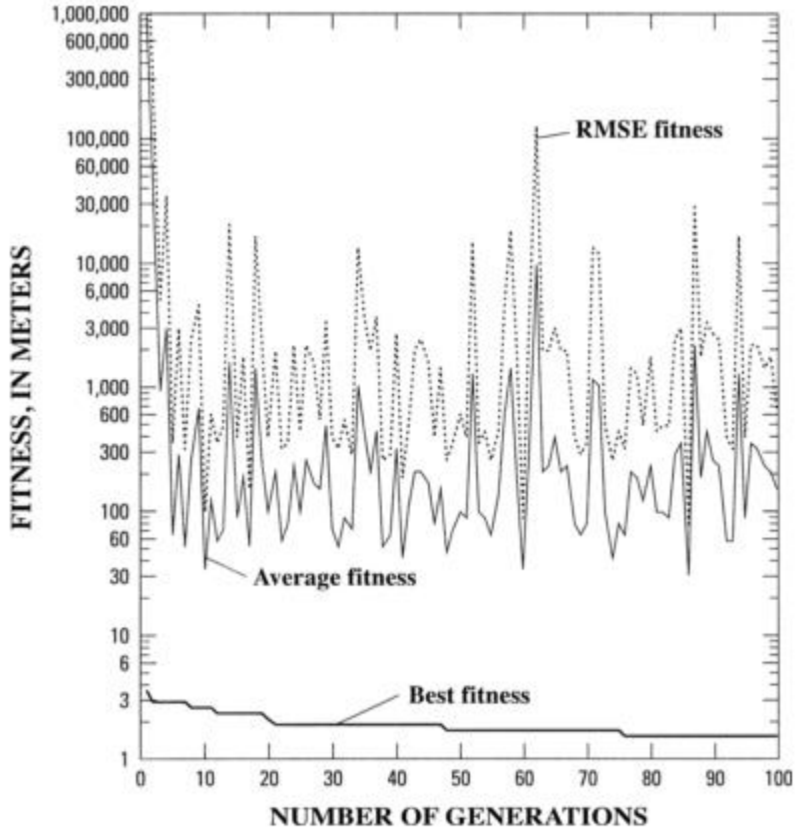


Figure 2.5. Best fitness, average fitness, and root mean squared error (RMSE) of fitness for each generation of run 9 for the hypothetical example.

2.5.2 Kootenai River Application

For the real world application, 10 cross sections out of 245 from the Kootenai River were arbitrarily selected (Table 2.2). The number of data pairs in the original cross sections ranged from 497 in cross section 152.019 to 2,521 in cross section 154.972 (Table 2.2, column 2). The GA initially was run with the same parameter values as in the hypothetical example, except plimit was set to the number of data pairs in the reduced cross sections from Barton *et al*. (2004) (see Table 2.2, column 3). Their procedure for reducing data pairs was previously discussed (see 2.2 Introduction).

Table 2.2. Comparison between best fitness values from genetic algorithm (GA) runs and values from standard procedures (Barton *et al*., 2004) for each cross section.

Cross Section	Number of Data Pairs in Original Dataset (n)	Number of Included Points	GA Run Time (seconds)	Best Fitness (meters)		
				Barton *et al*. (2004)	GA	Percent Lower
219.881	2,024	130	183.7	23.9	20.2	15.7
216.622	696	86	31.8	18.6	11.6	38.0
212.227	1,534	126	70.0	13.6	8.7	35.5
199.727	762	81	34.6	8.4	4.9	51.8
185.394	548	94	51.7	6.6	3.4	47.9
163.027	1,723	123	49.5	22.5	1.9	91.8
154.972	2,521	444	224.9	37.9	33.6	11.3
152.019	497	117	23.3	22.2	16.6	25.4
151.438	987	181	96.4	26.8	19.4	27.6
107.658	886	126	125.3	28.0	19.9	28.9
					Average =	29.0

Results with the parameters (P_c and P_m) used for the hypothetical case were somewhat poor, so a trial-and-error approach was used to determine the appropriate parameter values for each cross section. Ten runs usually were necessary to determine the appropriate values for the GA parameters (Table 2.3), which represent approximately 10 minutes of user time. This process probably could be further automated so that the GA automatically tests several standard parameter choices and uses the best one.

After parameter values were determined, the GA was run five times for each cross section. The best fitness results are shown in Table 2.2 (column 6). Best fitness values in all GA runs were consistently lower (better) than those of Barton *et al*. (2004), and several values were as much as 50 percent lower. On average, the GA fitness was 29 percent lower. These results indicate that GA cross sections are significantly more defined or representative of the original cross section than those generated by Barton *et al*. (2004).

Table 2.3. Genetic Algorithm (GA) Parameters Used in Runs for Each Cross Section.

Cross Section	Number of Generations	Population Size	Cross Over Rate (P_c)	Mutation Rate (P_m)
219.881	400	400	0.60	1/2.5n
216.622	400	200	0.70	1/2.5n
212.227	400	200	0.70	1/2.5n
199.727	400	200	0.70	1/5n
185.394	400	200	0.30	1/10n
163.027	400	200	0.60	1/3.5n
154.972	400	200	0.50	1/5n
152.019	400	200	0.70	1/10n
151.438	400	400	0.50	1/n
107.658	600	400	0.70	1/2.5n

The range of best fitness for cross section 199.727 was small. Best fitness for the five runs were 5.2 m (Run 1), 4.9 m (Run 2), 5.0 m (Run 3), 5.4 m (Run 4), and 5.8 m (Run 5). In fact, the range was small for all cross sections, and this suggests that near optimal results can be obtained in a single GA run. For cross section 199.727, the best fit individual (4.9 m for Run 2) was superimposed on the original cross section (Figure 2.6A) and matched the original cross section quite well. Even a detailed portion of the cross section showed a good match (Figure 2.6B). The other GA generated cross sections were similarly representative of the original cross section.

Again, as expected, the fitness for a run in cross section 199.727 decreased as the number of generations increased (Figure 2.7), indicating that the program functions correctly for large datasets. Best fitness decreased from 10.1 m to 5.2 m, about half at generation 400 from its initial fitness. Also, average fitness and RMSE generally decreased throughout the run. Similar results were observed in other runs from this section and in runs from other cross sections.

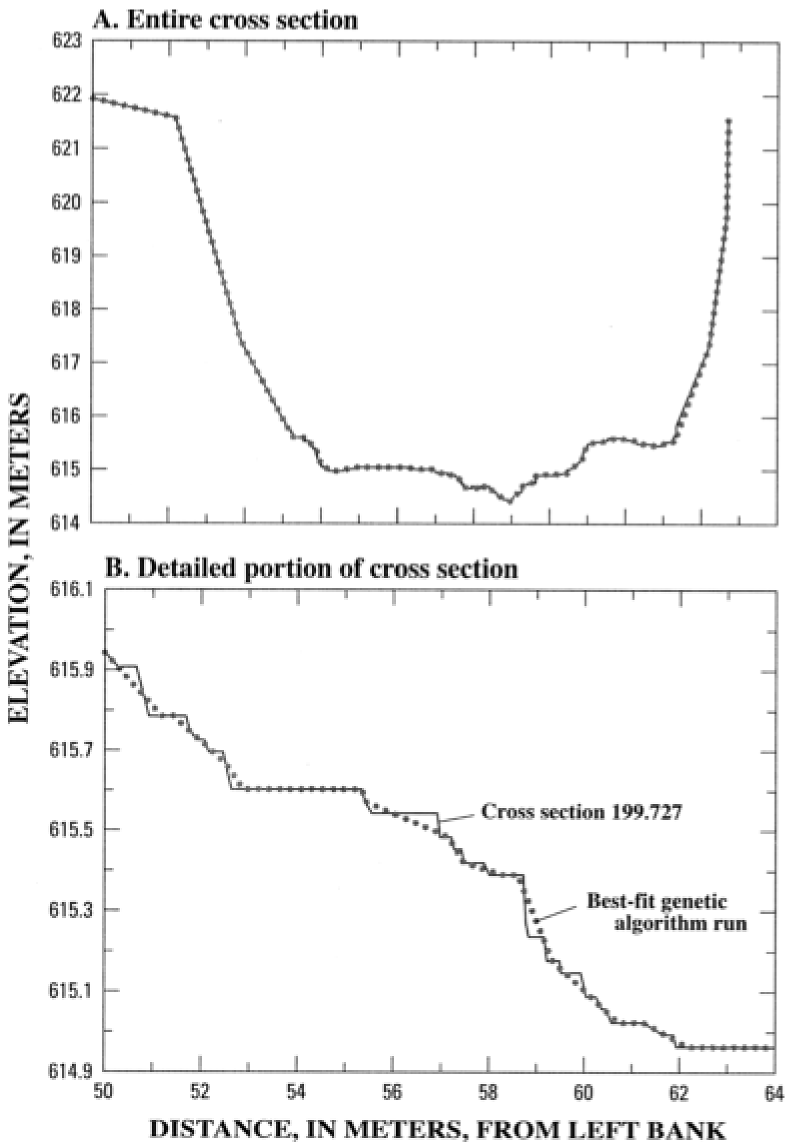


Figure 2.6. Cross section 199.727 and best-fit genetic algorithm run.

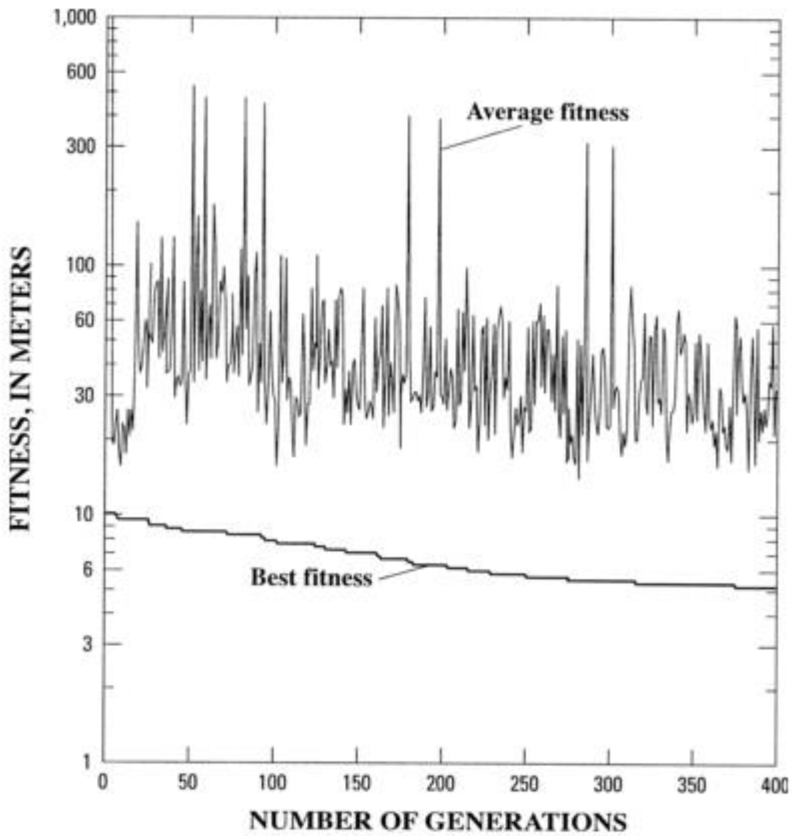


Figure 2.7. Best fitness and average fitness for each generation in cross section 199.727.

Genetic algorithm run time for these cross sections was very fast (Table 2.2, column 4) and was faster than the standard procedure used by Barton *et al*. (2004). It is estimated that it would take one week to perform all 245 cross sections by the GA, compared with the month required for Barton *et al*.'s procedure, a time savings of 75 percent. Run times were less than one minute for four cross sections, greater than one minute and less than two minutes for three cross sections, and greater than two minutes for three cross sections. These runs were performed on a 450 MHz PC. Run times would be faster using higher speed PCs.

2.6 Discussion and Conclusions

This paper demonstrates that a genetic algorithm can successfully solve the stream channel cross section reduction problem. Because the value of plimit is much less than the total number of data pairs in the cross section, initial populations created with a probability of plimit divided by string length (n) for inclusion significantly shortened the number of generations needed for successful results, especially for the larger Kootenai River datasets.

Fitness values for the GA run were all lower than those of Barton *et al*. (2004). Several GA cross sections had fitness values about 50 percent lower (better) than those of Barton *et al*. (2004). Results showed that GA cross sections closely represented the original cross section. Results also demonstrated that near-optimal results could be obtained in a single GA run, even for large problems.

Genetic algorithm run times for the hypothetical example and Kootenai River cross sections were much faster than the standard procedures used by Barton *et al*. (2004). Estimates indicate that it would take one week to complete the 245 Kootenai River cross sections using the GA, a time savings of 75 percent over the standard procedure.

To prevent a trial-and-error approach, an automated process is needed in the program that will determine the appropriate parameter values for each cross section. Also, a method is needed to select a plimit without relying on previous work, as in this study. The current fitness function always selects the plimit as the number of inclusions. A fitness function that gradually penalizes more points and gradually gives credit to fewer points might cause the GA to select a good minimum number of points. These improvements are left for future investigations.

Using this GA, other water resource, ecological, and biological data can be reduced in a method similar to that used for cross section data. For example, a dataset containing the location along a river (river mile) or highway (mileage) and the number of nonnative plants at the location can be reduced to a smaller and representative dataset. Time series data likewise can be reduced by transforming date values to a single numeric value such as Julian date.

2.7 References

Barton, G.J., E.H. Moran, and Charles Berenbrock, 2004. Stream Channel Cross Sections for the Kootenai River Between Libby Dam, Montana, and Kootenay Lake, British Columbia, Canada. U.S. Geological Survey Open-File Report 2004-1045, Boise, Idaho. p. 35.

Chen, Li, 2003. Real Coded Genetic Algorithm Optimization of Long Term Reservoir Operation. Journal of the American Water Resources Association (JAWRA) 39(5):1157-1165.

Cieniawski, S.E., J.W. Eheart, and S. Ranjithan, 1995. Using Genetic Algorithms to Solve a Multiobjective Groundwater Monitoring Problem. Water Resources Research 31(2):399-409.

Davis, L., 1991. Hybridization and Numerical Representation. In: Handbook of Genetic Algorithms, L. Davis (Editor). Van Nostrand Reinhold, United Kingdom, pp. 61-71.

FEMA (Federal Emergency Management Agency), 1995. Guidelines and Specifications for Study Contractors. Federal Emergency Management Agency, Publication 37, U.S. Government Printing Office, Washington, D.C., p. 174.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, Massachusetts.

Grefenstette, J.J., 1990. Genetic Algorithms and Their Applications. In: Encyclopaedia of Computer Science and Technology, A. Kent and J.G. Williams (Editors). Marcel Dekker, New York, New York, pp. 139-152.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan.

Knaapen, M.A..F. and S.J.M.H. Hulscher, 2003. Use of a Genetic Algorithm to Improve Predictions of Alternate Bar Dynamics. Journal of Water Resources Research 39(9):1231, doi 10.1029/2002WR001793, 2003.

McKinney, D.C. and M.D. Lin, 1992. Design Methodology for Efficient Aquifer Remediation Using Pump and Treat Systems. In: Mathematical Modeling in Water Resources, T. Russel et al. (Editors). Elsevier Science, New York, New York, pp. 695-702.

Mitchell, M., 2002. An Introduction to Genetic Algorithms. The MIT Press, Cambridge, Massachusetts, (8th printing), p. 209.

Moran, E.H. and Charles Berenbrock, 2003. GPS – Time Saver and Functional. U.S. Geological Survey Western Water Watch 1(1): 6-7.

Reed, P., B. Minsker, and D.E. Goldberg, 2000. Designing a Competent Simple Genetic Algorithm for Search and Optimization. Journal of Water Resources Research 36(12):3757-3761.

Reed, P., B. Minsker, and D.E. Goldberg, 2003. Simplifying Multiobjective Optimization – An Automated Design Methodology for the Nondominated Sorted Genetic Algorithm-II. Journal of Water Resources Research 39(7):1196-1206.

Vink, K. and P. Schot, 2002. Multiple-Objective Optimization of Drinking Water Production Strategies Using a Genetic Algorithm. Journal of Water Resources Research 38(9):1157-1165.

CHAPTER 3. REDUCING CROSS-SECTIONAL DATA USING A GENETIC ALGORITHM METHOD AND EFFECTS ON CROSS-SECTION GEOMETRY AND STEADY-FLOW PROFILES

3.1 Abstract

Reduction of cross-sectional data using a genetic algorithm method, and the effects of data reduction on channel geometry and steady-flow profiles, were analyzed. Two reduction methods─standard and genetic algorithms─were used to reduce cross-sectional data from the Kootenai River in northern Idaho. Cross sections that are representative of meander, straight, braided, and canyon reaches were used to evaluate the reduction methods. Visual and hydraulic analyses were used to assess the methods. The genetic algorithm-reduced cross sections approximated the shape of the original cross sections better than the standard-reduced cross sections. A greater number of cross-sectional data points were needed for reduced cross sections in the straight reach, and even more in the braided reach, because a greater amount of data points are needed to adequately define cross sections that have greater topographic variability. For the genetic algorithm-reduction method, about 40 data points were needed to adequately define the shape of a reduced cross section in the braided reach compared to 10 to 20 data points in the meander and canyon reaches. The standard-reduction method needed about 70 data points for the braided reach and more than 30 points for the meander and canyon reaches. The genetic algorithm can effectively reduce data while staying within the threshold set by the maximum number of points to be included in the reduced dataset.

The effects of reduced cross-sectional data points on steady-flow profiles were also determined. Thirty-five cross sections of the original steady-flow model of the Kootenai River were used. These two methods were tested for all cross sections with each cross section resolution reduced to 10, 20 and 30 data points, that is, six tests were completed for each of the thirty-five cross sections. Generally, differences from the original water-surface elevation were smaller as the number of data points in reduced cross sections increased, but this was not always the case, especially in the braided reach. Differences were smaller for reduced cross sections developed by the genetic algorithm method than the standard algorithm method.

3.2 Introduction

Cross-sectional data are used for many purposes, such as the investigations of flood plain delineation, flow patterns, shear stress, sediment mobility and transport, channel evolution, and aquatic habitat conditions. Accuracy of cross-section data is important because it could affect channel-geometry determinations and water-surface profile calculations. For example, the consequence of errors in the water-surface elevation has a major effect on computations of velocity, shear stress, and sediment transport. Water-surface profiles in many studies are computed by using one-dimensional (1-D) step-backwater models such as HEC-RAS (Brunner, 2010), which uses the standard step method for steady flow. The standard step method uses the energy, continuity, and flow resistance (for example, Manning's) equations between cross sections to compute the water-surface elevation and streamflow velocity (Chow, 1959).

All models have a limit to the number of points allowed in a cross section. For example, the HEC-RAS version 4 model has a 500-point limit. This limit might seem large enough, but when cross sections are computer generated from Light Detection and Ranging (LiDAR) or Digital Elevation Models (DEMs), or data are collected with equipment such as an echo sounder, the number of cross-sectional data points tends to be quite substantial. In an earlier study (Barton *et al*, 2004), for example, approximately 400 cross sections were surveyed on the Kootenai River in northern Idaho in order to understand the hydraulic characteristics of the river and to promote hydraulic conditions that improve spawning conditions for the endangered Kootenai River white sturgeon. The number of data points for each cross section ranged from about 500 to more than 2,500 points (Moran and Berenbrock, 2003; Barton *et al.*, 2004). Only a few cross sections had more than 2,000 data points. More than half of the surveyed cross sections were included in a HEC-RAS model of the Kootenai River (Berenbrock, 2005, 2006a), and most cross sections were reduced to less than 150 data points.

Large datasets must be reduced to the meet the limitations of the programs being used. Reduced datasets improve run-time performance and facilitate data transmission and storage. Selecting the appropriate data points to keep from among the hundreds or thousands of data points can be both challenging and tedious. However, reducing the number of cross-sectional data points can result in significant changes to the reduced cross section, which could affect computed water-surface elevations, streamflow velocity, shear stress, and sediment transport. Considerable care must be taken when reducing data so that computed errors and uncertainties remain small or within acceptable limits. It is important to understand the effects that reduced cross sections can have on National Flood Insurance Program, flood-inundation, habitat, and sediment-transport studies. The Federal Emergency Management Agency (FEMA) has indicated that there is no point minimum—the number of data points for defining a cross section—as long as the actual shape of the cross section is well defined (Federal Emergency Management Agency, 2007).

Much research has been done on error and uncertainty analysis in surface-water hydraulics. Research has been carried out on determining the optimal spacing between cross sections (Samuels, 1989; Castellarin *et al.*, 2009); developing cross sections from topographic maps, LiDAR, and DEM data (Burnham and Davis, 1990; Pasternack *et al.*, 2004; Cook and Merwade, 2009); and interpolating cross sections between known cross sections (Traver and Miller, 1993). Travis and Lokey (1999) developed a method to reduce cross-section data to 100 data points, the maximum limit of the HEC-2 model. Berenbrock (2006b) developed a genetic algorithm (GA) computer program that reduces the number of data points in a cross section to any size. He compared the GA-reduced cross sections to cross sections developed by standard reduction methods—selecting every 10th, 20th, or *n*th point and omitting the rest—for the same number of data points in a cross section. Reduced cross sections developed from standard and GA methods were compared to the original cross-sectional data, and results showed that the GA method produced smaller differences from the original cross-sectional data than those obtained by using standard procedures. Unfortunately, no research to date has been done to determine the optimal number of points that are needed in a cross section or the effects on cross-section geometry and steady-flow profiles. The optimal number of points depends on the degree of topographic variability and the scale of topography that is of interest. It also depends on laws, regulations, or the requirements of the funding party—for example, FEMA (Federal Emergency Management Agency, 2007).

The purpose of this report is to describe an application of a GA to the reduction of cross-sectional data points, demonstrate that the GA is a viable approach, and to evaluate the effects of reduced cross sections on channel geometry and steady-flow profiles. First, the study compared the accuracy of two reduction approaches, standard and genetic algorithm methods. Data from 10 cross sections covering 4 different channel types on a river were reduced by standard and genetic algorithm methods. These reduction methods were employed because the raw (original) data are preserved, not averaged, interpolated, or extrapolated. Second, the study identified the sources and spatial distribution of error in different channel types and determined the requisite sample size for different scales of resolution and application. Third, the study examined the effects of data reduction on steady-flow profiles. From these components, the amount of reduction can be tailored to the goals of an application.

3.3 Reduction Methods

There are many data-reduction methods available. For this study, only methods that preserve the original data were considered. The advantages of preserving the original data are that the original features, including vertical banks and discontinuities, are maintained. The original cross section—whether it contains 2,000 data points or 20 data points—is more accurate than anything generated in part from those data. Stream-channel cross-section data from the Kootenai River (Barton *et al.*, 2004) were used in the data-point reduction methods. Data for the streambed part of these cross sections were collected by connecting continuous Real-Time Kinematic (RTK) Global Positioning System (GPS) equipment to an echo sounder, and bank data were collected by connecting a RTK GPS to a laser rangefinder equipped with an angle encoder (Moran and Berenbrock, 2003). Berenbrock (2006b) specifically used 10 cross sections from the Kootenai River to substantiate a genetic algorithm (GA) for data-point reduction. Data points from these 10 cross sections were also used in this study because the original data were still available. The cross sections are 107.658, 151.438, 152.019, 154.972, 163.027, 185.394, 199.727, 212.227, 216.622, and 219.881, which are defined by a station number in river miles[1] that corresponds to its location on the river. The number of data points in these sections ranged from 497 in cross-section 152.019 to 2,521 in cross-section 154.972. Original data from the other cross sections on the Kootenai River were not available. However, there were still enough data points in the reduced cross sections by Barton *et al.* (2004)—usually more than 100 data points per cross section, with some sections containing several hundred data points—for further reduction. For this study, cross-section data were reduced to as few as 10 data points per cross section.

[1]River mile locations are based on Columbia Basin Inter-Agency Committee (1965) river-mile index for the Kootenai/Kootenay River. River mile 0 is at the confluence of the Kootenay River and Columbia River near Castlegar, British Columbia, Canada, and river mile 152 is upstream on the Kootenai River near Bonners Ferry, Idaho.

3.3.1 Standard Reduction

The standard-reduction method to reduce data points is to keep every 10th, 20th, 30th, or nth point and discard (omit) the rest. This procedure was used in this study because of the standard-reduction method's simplicity, ease, and quickness. The value of the nth data point for each cross section was different because the total number of data points in the cross sections was different. For example, cross-section 152.019 has 497 data points. If the number of points was reduced to 20, then every 26th point would be selected with 2 points remaining ($1+(26\times19)=495$; then $497-495=2$). The first data point in the cross section is always kept, so a value of 1 is added to the number of intervals (19 for this example). Fewer points exist along the banks than on the streambed for this dataset because the bank data were collected manually with a laser rangefinder. The remaining two points in this example were inserted into two different intervals that spanned the streambed. Thus, the size of those intervals was increased to 27. This procedure ensures that the last point in the cross section is kept and, for this example, the 497th point (last point) was kept ($1+(26\times17)+(27\times2)=497$).

3.3.2 Genetic Algorithm Reduction

Reducing the number of data points in a cross section is a non-linear combinatorial problem and, therefore, is well suited for heuristic algorithms such as genetic algorithm (GA). GAs mimic the natural selection and survival of the fittest and are well suited for solving combinatorial optimization problems in which there is a large set of candidate solutions (Fisher, 2013). Koza (1992, p. 18) provides the following definition of a GA:

> The genetic algorithm is a highly parallel mathematical algorithm that transforms a set (population) of individual mathematical objects (typically fixed-length character strings patterned after chromosome strings), each with an associated fitness value, into a new population (i.e., the next generation) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (notably sexual recombination).

In a GA, a population is represented by a number of individuals called genes (strings of chromosomes). Individuals are produced by 'mating' (crossover of chromosomes) two individuals together and 'mutating' a chromosome. The fittest individuals in the new population are selected to breed and mutate again, passing their genetic information to their children to create a newer population, and the least fit individuals are discarded. The newer population is then used in the next iteration of the algorithm. This process is repeated until a number of iterations has been reached or the maximum number of consecutive iterations without any improvement to the best fit individual is exceeded. Note that each individual is a solution to the problem. In essence, the GA represents an "intelligent" exploitation of the search space in a random fashion to solve a problem. A more complete discussion on genetic algorithms is given in Goldberg (1989), Grefenstette (1990), Davis (1991), and Mitchell (2002).

The GA developed by Berenbrock (2006b) [Chapter 2] was used in this study to reduce cross-section point data because it is easy to use, fast, and preserves the original data. The fitness function in Berenbrock's (2006b) [Chapter 2, Equation 2.1] GA, however, is biased because the x-value (distance) for each data point does not contribute to the fitness function—only the y-values (elevation) do. The function does not account for the varying distances between the data points (irregularly spaced data) and, thus, data points need to be regularly spaced along a cross section. However, most cross sections are composed of irregularly spaced points. To account for irregularly spaced points, the fitness function was modified to calculate the area between the original and reduced cross sections–noted as the area between the cross sections (ABC). Thus, ABC accounts for the contribution from both x and y values. The value of ABC is always positive, regardless of how the cross sections cross one another. To solve this mathematically, the absolute value of ABC is employed and is denoted as |ABC|. The modified two-conditional fitness function is as follows:

$$f(i) = \begin{cases} \sum_{j=1}^{n} \left|ABC_j\right| & \text{if } inc_i \le plimit \\ \left[\left(\sum_{j=1}^{n} \left|ABC_j\right|\right) + 1\right] \times 10^{(inc_i - plimit)} & \text{otherwise} \end{cases} \qquad \text{(Equation 3.1)}$$

where

$f(i)$	is the value of fitness for individual i;
n	is the number of data points in the original dataset;
ABC_j	is the area between the cross sections, original and reduced, for trait j;
inc_i	is the number of included data points in individual i; and
plimit	or point limit is the maximum number of points to be included in the reduced dataset.

Each individual in the GA represents a reduced cross section, and the fitness of an individual (thus, a fitness of traits or data-point combinations) is represented as a value from the fitness function. The GA minimizes the fitness function, *Minimize* f(i), to identify the best-fit or optimal individual from all possible data-point combinations.

A sample calculation of fitness for a hypothetical cross section is shown in Figure 3.1. Reduced cross sections are composed of included and excluded points. Included points are data points that are kept from the original cross section, and excluded points are data points that are discarded from the original cross section. For the reduced cross section shown in Figure 3.1, the included data points are at points 1, 2, 5, 6, 9, and 11, and excluded data points are at points 3, 4, 7, 8, and 10. The area between the cross-sections (ABC), original and reduced, is calculated for each closest pair of points. For the first pair of points 1–2, ABC is calculated to be 0 square meters (m^2) because the data points for both cross sections are the same. For the second pair (2–3), ABC is calculated to be 53.8 m^2. For point pairs 3–4 and 4–5, ABC is calculated to be 40.3 m^2 and 0.4 m^2, respectively. For point pairs 5–6, ABC is calculated to be 0 m^2 because the data points for both cross sections are the same. For point pairs 6–7, 7–8, 8–9, 9–10, and 10–11, ABC is calculated to be 0.5 m^2, 25.2 m^2, 34.3 m^2, 13.4 m^2,

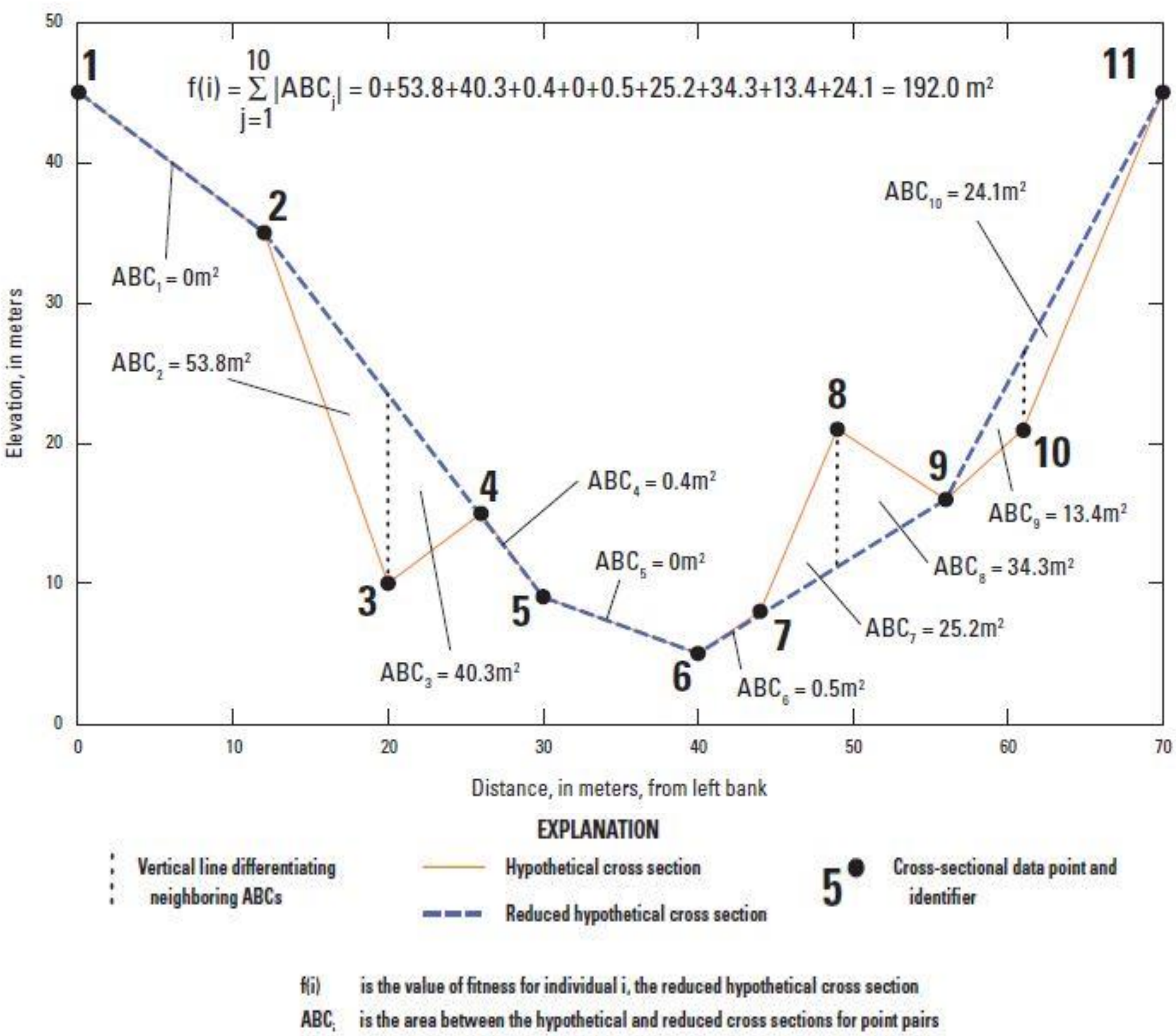


Figure 3.1. Fitness calculations for a hypothetical cross section. Fitness is calculated according to equation 3.1 as shown at the top of the figure. m^2, square meters.

and 24.1 m^2, respectively. The total ABC is 192.0 m^2, which is the fitness value for this reduced hypothetical cross section. Fitness serves to aggregate the errors of an individual into a single measure. It is a good measure of accuracy, but only between other individuals in the population, as it is scale dependent.

To validate the modifications made to the GA, the 10 cross sections that were used to validate the original GA (Berenbrock, 2006b) were used. The GA was run with the same parameter values as in the original GA. The best fitness results are shown in Table 3.1. The reduction method used by Barton *et al.* (2004), however, was based on selecting points every 1 to 2 meters (m), plus additional user-specified points to capture any important missing topography as determined from visual inspection (Berenbrock, 2006b, p. 388); this results in a variable number of reduced data points per cross section (Table 3.1, column 3), rather than a fixed number of points, as described by the standard-reduction method. To compare these methods, the fitness for the best fit reduced cross sections from Barton *et al.* (2004) and original GA (Berenbrock, 2006b) were recalculated by using Equation 3.1 and presented in Table 3.1.

The best fitness values for the GA runs were consistently less (better performance, more accurate) than those from Barton *et al.* (2004) and original GA (Berenbrock, 2006b) methods. On average, the GA fitness was 39.2 percent and 57.0 percent less than Barton *et al.* (2004) and original GA methods, respectively. These results indicate that the GA cross sections are significantly more (better) defined or representative of the original cross section than cross sections by the other two reduction methods. Note that the original GA fitness values were usually greater (lower performance) than Barton *et al* (2004) fitness values because the original GA fitness function was optimized for regularly spaced data, not irregularly spaced data as constitute most cross sections.

The GA-reduction method preserves the detailed character of the original cross section better than Barton *et al.* (2004) and original GA (Berenbrock 2006b) methods. The GA cross sections were generally more defined where the original cross section had more topographic variability. The GA cross section matched the original cross section quite well, especially in the relatively smooth areas. The point density in the relatively smooth areas was far more reduced in the GA section than in Barton *et al.* (2004) and original GA sections. Conversely, the point density in the relatively rough areas (topographic variability) was increased more in the GA section than in Barton *et al.* (2004) and original GA sections. For the most part, the detailed character of the original cross section was better preserved by using the GA method than using the other two reduction methods. Therefore, the

Table 3.1. Comparison of best fitness between several reduction methods for 10 cross sections on the Kootenai River, Idaho.

[GA, genetic algorithm]

Cross sections	[1]Number of data points in original dataset	[1]Number of reduced data points (npr)	Percentage Reduction Of data points	[2]Best fitness, in square meters			GA method percentage lower than	
				[3]Barton *et al.* (2004) method	[3]Original GA (Berenbrock, 2006b) method	GA method	Barton *et al.* (2004) Method	Original GA (Berenbrock, 2006b) method
107.658	886	126	85.8	4.26	3.87	2.52	40.8	34.9
151.438	987	181	81.7	7.02	14.08	3.53	49.7	74.9
152.019	497	117	76.5	6.37	8.27	3.47	45.5	58.0
154.972	2,521	444	82.4	8.72	10.27	5.69	34.7	44.6
163.027	1,723	123	92.9	1.34	3.57	1.05	21.6	70.6
185.394	548	94	82.8	1.10	0.92	0.56	49.1	39.1
199.727	762	81	89.4	0.96	1.37	0.51	46.9	62.8
212.227	1,534	126	91.8	1.12	1.15	0.60	46.4	47.8
216.622	696	86	87.6	1.96	2.99	1.16	40.8	61.2
219.881	2,024	130	93.6	1.63	5.63	1.36	16.6	75.8
Average =			86.4				39.2	57.0

[1]Barton *et al.* (2004).

[2]Calculated from Equation 3.1, which is based on the area between the original and reduced cross sections (ABC).

[3]To compare these methods, the fitness for each best fit reduced cross section from Barton *et al.* (2004) and original GA (Berenbrock, 2006b) [Chapter 2] were recalculated using Equation 3.1.

modification that was made to the GA method is the more appropriate genetic algorithm-reduction method and was used throughout this study.

3.4 Comparison of Reduction Results

To evaluate the effect on cross-section geometry and steady flow profiles, comparisons from the standard-reduction method and GA-reduction method were conducted. The comparisons included the 10 cross sections from the Kootenai River in Table 3.1. The original cross sections were reduced in size to 10, 20, and 30 data points using both standard-reduction and GA-reduction methods. Harrelson *et al.* (1994) determined that at least 20 data points are needed in a cross section to accurately describe the character of the channel. The reduction to 30 data points was selected because a greater amount of data points are needed if the cross section is quite broad or complex, such as the case with braided channels. The reduction to 10 data points was selected because it is one-half of the recommended minimum (Harrelson *et al.*, 1994). Results from the reductions' visual and hydraulic analyses are presented in the following sections. The practical consequences from both reduction methods are investigated in the "Hydraulic Modeling Analysis of Reduction Methods" section by the use of one-dimensional (1-D) steady-flow profiles.

3.4.1 Visual Analysis of Cross-Section Reductions

After reducing the cross-section data, the reduced datasets were viewed graphically and compared to the original data. Although the analysis of fitness is useful as given in Table 3.1, it is a black-box approach that reports aggregate results without providing an understanding of the spatial details and does not distinguish which parts of the cross section are causing the error. Visual analysis provides further insight regarding the spatial distribution of error for the two reduction methods.

Synder and Minshall (1996) identified three geomorphic reaches in the Kootenai River—a meander reach, a braided reach, and a canyon reach. Barton *et al.* (2005) defined a fourth geomorphic reach—a straight reach. The meander reach is a single channel with gentle bends. The streambed consists primarily of fine sand. Water depths usually exceed 12 m, and the water-surface slope is about 2×10^{-5} meters per meter (m/m), less than one-twentieth the slope in the braided reach. Sand dunes—as high as 1.4 m and as long as about 23 m (Barton *et al.*, 2005)—also occur throughout the meander reach. The straight reach is a transitional reach between the meander and braided reaches, and its streambed consists primarily of sand, gravel, and cobbles. The braided reach usually consists of multiple channels, and the streambed is composed primarily of gravel and cobbles. Water depths usually are less than 2 m, and water-surface slope is about 4.6×10^{-4} m/m. The canyon reach consists of a long, straight single channel with steep canyon walls and is incised into bedrock. The streambed consists primarily of cobbles and boulders. Water depths are usually about 6 m, and water-surface slope is about 3×10^{-4} m/m.

Examples for each geomorphic or channel type—meander, straight, braided, and canyon (Czuba and Barton, 2011)—are shown in Figure 3.2. Cross-section 107.658 is in the meander reach, cross-section 152.019 is in a straight reach, cross-section 154.972 is in a braided reach, and cross-section 163.027 is in a canyon reach. The effects of standard data-point reduction on cross-sectional shape are shown in Figures 3.2A, 3.2C, 3.2E, and 3.2G, and effects of GA reduction on cross-sectional shape are shown in Figures 3.2B, 3.2D, 3.2F, and 3.2H.

At first glance, the reduced cross sections composed of 10 data points showed that it was the worst shaped case for both reduction methods. The graphs in Figure 3.2 indicate that more than 30 points were needed when using the standard-reduction method, whereas 20 to 30 data points were adequate when using the GA-reduction method, except for cross-section 154.972 (Figure 3.2D). The GA-reduced cross sections approximated the shape of the original cross sections better than the standard-reduced cross sections for the same number of reduced data points (npr). The standard-reduction method produced greater bank errors than the GA method; also there were greater streambed errors in the braided and straight reaches. At cross-section 154.972 (braided reach), a greater number of data points were needed for both reduction methods because there was much more topographic variability in this cross section than in the other cross sections. Additional analysis indicated (not shown in Figure 3.2) that about 70 data points were needed to adequately define the shape of cross-section 154.972 when using the standard-reduction method and about 40 points when using the GA-reduction method.

A visual comparison was performed on cross-sectional area plots. Area was used instead of conveyance because the performance of conveyance can be confounded by the uncertainly in coefficients in Manning's flow equation (Chow, 1959). The effects of data-point reduction on cross-sectional area for the cross sections in Figure 3.2 are shown in Figure 3.3. The cross-sectional area curves for the reduced cross sections composed of 10 points were furthest from the original area curves for both reduction methods, and were closest to the original curves for the 30-point cross sections for both methods. Cross-sectional areas for the GA-reduced cross sections approximated the shape of the original cross-sectional areas better than the standard-reduced cross sections for respective data-point reduction. But for cross-section 154.972 (braided reach), none of the reduced

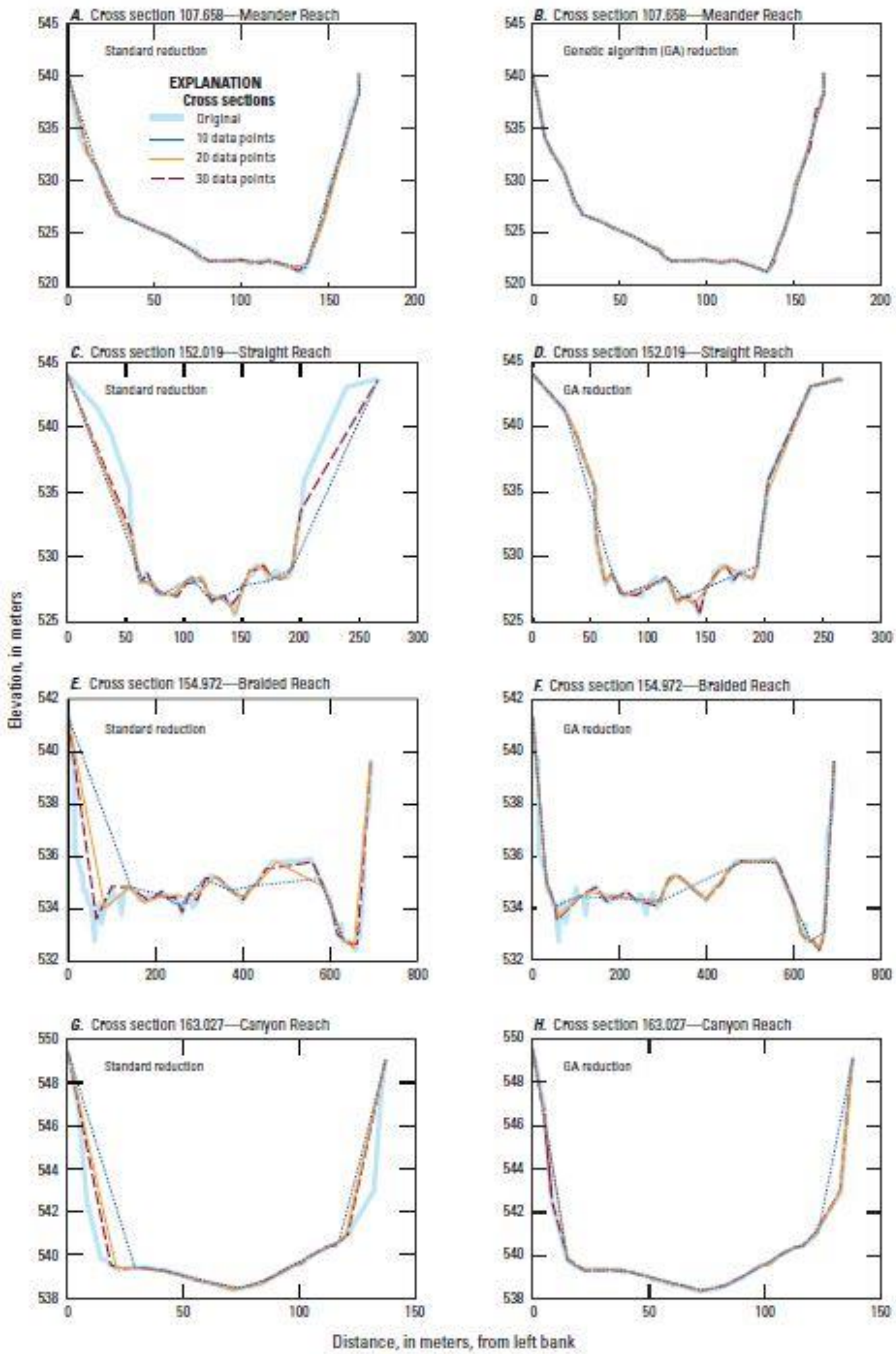


Figure 3.2. Effects of data-point reduction on cross-sectional shape.

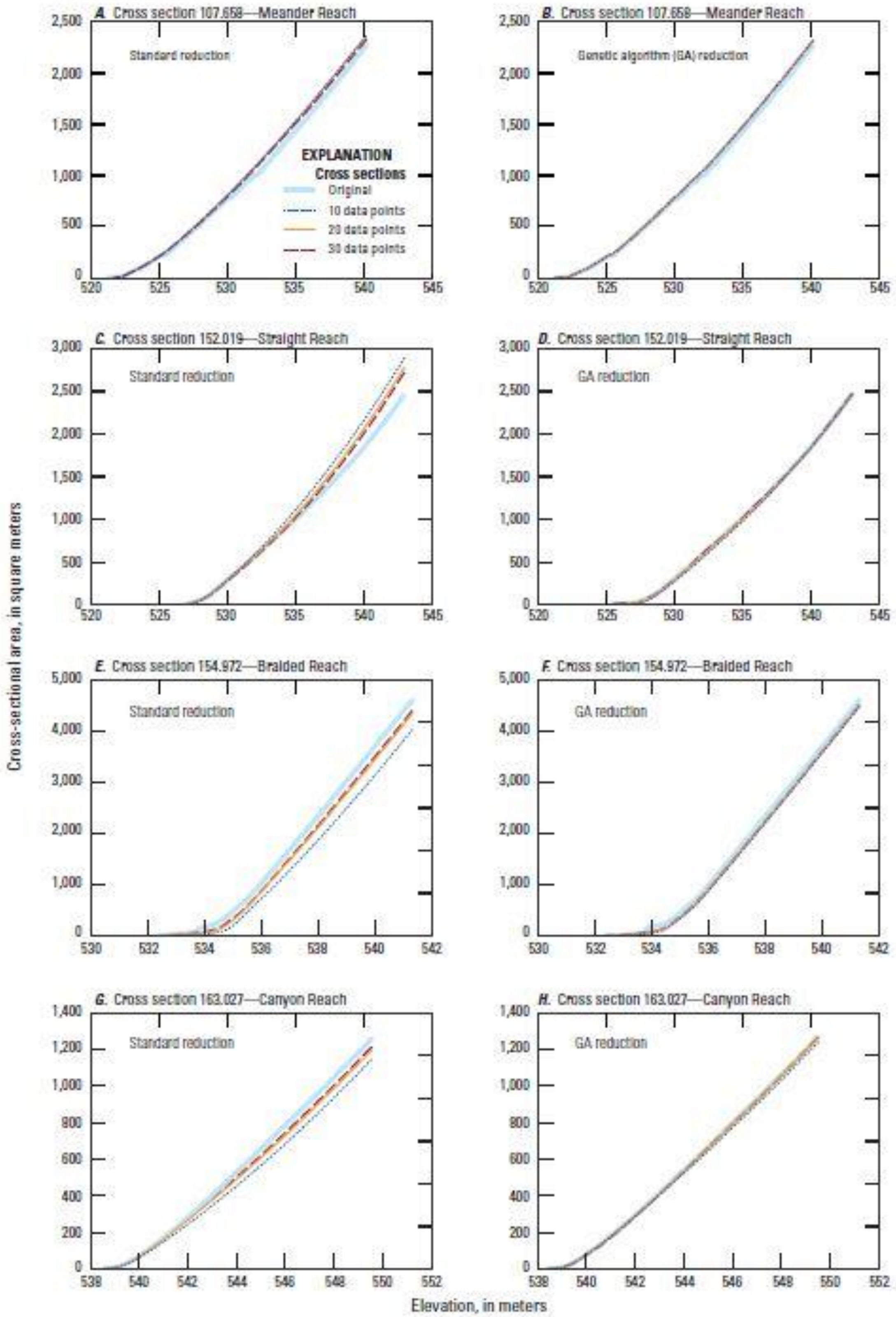


Figure 3.3. Effects of data-point reduction on cross-sectional area.

area curves for both methods closely agreed with the original area curves (Figures 3.3E and 3.3F). The original area curves show a break near an elevation of 534 m that is probably caused by a greater amount of topographic variability in the original cross section (Figures 3.2E and 3.2F) compared to the reduced cross sections. For the reduced cross sections, this departure at the break in slope indicates that there are not enough data points to adequately define the topographic variability in this area.

The performance of the cross-section reduction is quantified by the measured error in the reduced cross section. This error is designated as reduction error (RE) and is defined as the total area between the original and reduced cross-section curves (ABC) divided by the number of reduced data points (npr) and is expressed as follows:

$$RE=\frac{ABC}{npr} \qquad \text{(Equation 3.2)}$$

The RE normalizes the reduction methods with respect to the number of npr and allows for direct comparison of different cross sections and reduction methods. The RE has units of area per point; for the datasets used in this study, it was square meters per reduced data point (m^2/point). The value for RE depends on the number of points in the original and reduced cross sections and the topography of the cross section. The smaller the RE, the greater similarity between the original and reduced cross sections. The larger the RE, the greater the disparity between the topography of the original and reduced cross sections. Equation 3.2 represents the gain in cross-sectional likeness due to the reduction of the area between the cross-section curves with the increase in the number of reduced data points. The RE values are unique and depend on cross-section topography and the number of data points in the original and reduced cross sections. The accuracy of the cross-section reduction is likely to be sensitive to the quality and quantity of the original data and how it represents the different scales of topography that are present. For example, the performance of the cross-section could differ if a topographically complex channel was surveyed by 40 data points as opposed to 400 data points. This is not an issue for the Kootenai data, given the high density of original data points, but it could be an issue in other studies.

Values of RE for reduced cross sections that contained 10, 20, and 30 data points were calculated for the 10 cross sections on the Kootenai River by using the standard- and GA-reduction methods (Table 3.2). For the GA-reduced cross sections, fitness or ABC was calculated by the GA program (Appendix E). For the standard-reduced cross sections, ABC was calculated using the first part of Equation 3.1, similar to the calculation shown in Figure 3.1. Then, Equation 3.2 was used to calculate the RE for both methods. As shown in Table 3.2, as the number of reduced data points increase, the RE decreases. Also the RE values for the GA-reduced cross sections were always less than the RE values for the standard-reduced cross sections, indicating that the GA-reduced cross sections are more representative of the original data than the standard-reduced cross sections. The RE values for the canyon-reach cross sections were usually consistent with one another for the number of data points for both methods. For the GA method, the canyon cross sections had the lowest RE values, and cross-section 154.972 (braided reach) had the highest RE values. For the standard-reduction method, cross-section 107.658 (meander reach) had the lowest RE values because the banks in this cross section were gently sloping, thus, reducing its ABC value (Figure 3.2A). Also, for the standard-reduction methods, cross-section 152.019 (straight reach) had the highest RE values because the method selected only a few points on the banks, thus, causing ABC to be quite large (Figure 3.2C). Similarly, that is the reason for the large RE values for cross-section 151.438 (straight reach).

The GA program was used to develop RE curves (Figure 3.4) for the 10 cross sections listed in Table 3.2. Each curve was developed by running the program for a selected number of reduced data points (npr) starting at 10 data points and incrementing by 10 until reaching 100 data points. The program was run 10 times at every npr to ensure that near optimal results were reached. The lowest RE value (calculated from Equation 3.2) at each npr was retained and used to develop the RE curve for every cross section. Also, the ABC and npr values from Table 3.1 for the modified GA were used, and, for seven cross sections, they were used to extend the RE curves beyond an npr of 100. These curves represent near optimal solutions. All RE curves in Figure 3.4 are concave upward to the right. The RE curve for cross-section 154.972 (braided reach) is more upward toward the right than all the other RE curves because its cross section was more complex (greater topographic variability). In contrast, cross sections in the meander and canyon reaches were less complex (less topography variability), causing the RE curves to be in the lower part of the plot. The RE curves for cross sections in the straight reach were between the canyon and braided curves. The RE plot (Figure 3.4) shows that as topographic variability in a cross section increases, the RE curve will be more upward toward the right in the plot, thereby indicating that increasing the number of points in a cross section needs to be increased to adequately represent the original cross section. The RE curve also allows one to visually judge where an increase in npr does not result in significant RE reduction (called the point of diminishing returns). This location on a RE curve is at the point of diminishing returns (breakpoint). For this study, the breakpoint's location was determined by a two-phase linear regression where two straight lines are fitted to the data by minimizing the residual sum of squares. Above the breakpoint (to the left on the curve), the RE value increases quite rapidly as the number of data points decrease; below the breakpoint (to the right on the curve), the increase in the number of data points does not

Table 3.2. Reduction error (RE) values for 10 cross sections resulting from the standard and genetic algorithm (GA) reduction methods, Kootenai River, Idaho.

	Reduction error, in square meters per reduced point					
	Number of reduced data points in cross sections					
Cross	Standard method			GA method		
Section	**10**	**20**	**30**	**10**	**20**	**30**
Meander Reach						
107.658	4.9	1.7	0.8	2.0	0.6	0.2
Straight Reach						
151.438	33.9	9.0	4.3	11.6	2.1	0.9
152.019	49.8	18.1	10.4	10.9	1.7	0.6
Braided Reach						
154.972	41.1	10.4	4.2	18.2	4.7	2.6
Canyon Reach						
163.027	11.8	6.1	3.9	3.5	0.7	0.1
185.394	13.0	3.4	1.5	1.8	0.2	0.1
199.727	11.1	4.0	2.5	3.4	0.1	<0.1
212.227	9.7	2.5	1.5	4.1	0.9	0.2
216.622	7.6	2.7	1.3	3.5	0.6	0.2
219.881	17.0	4.6	2.1	2.7	1.2	0.6

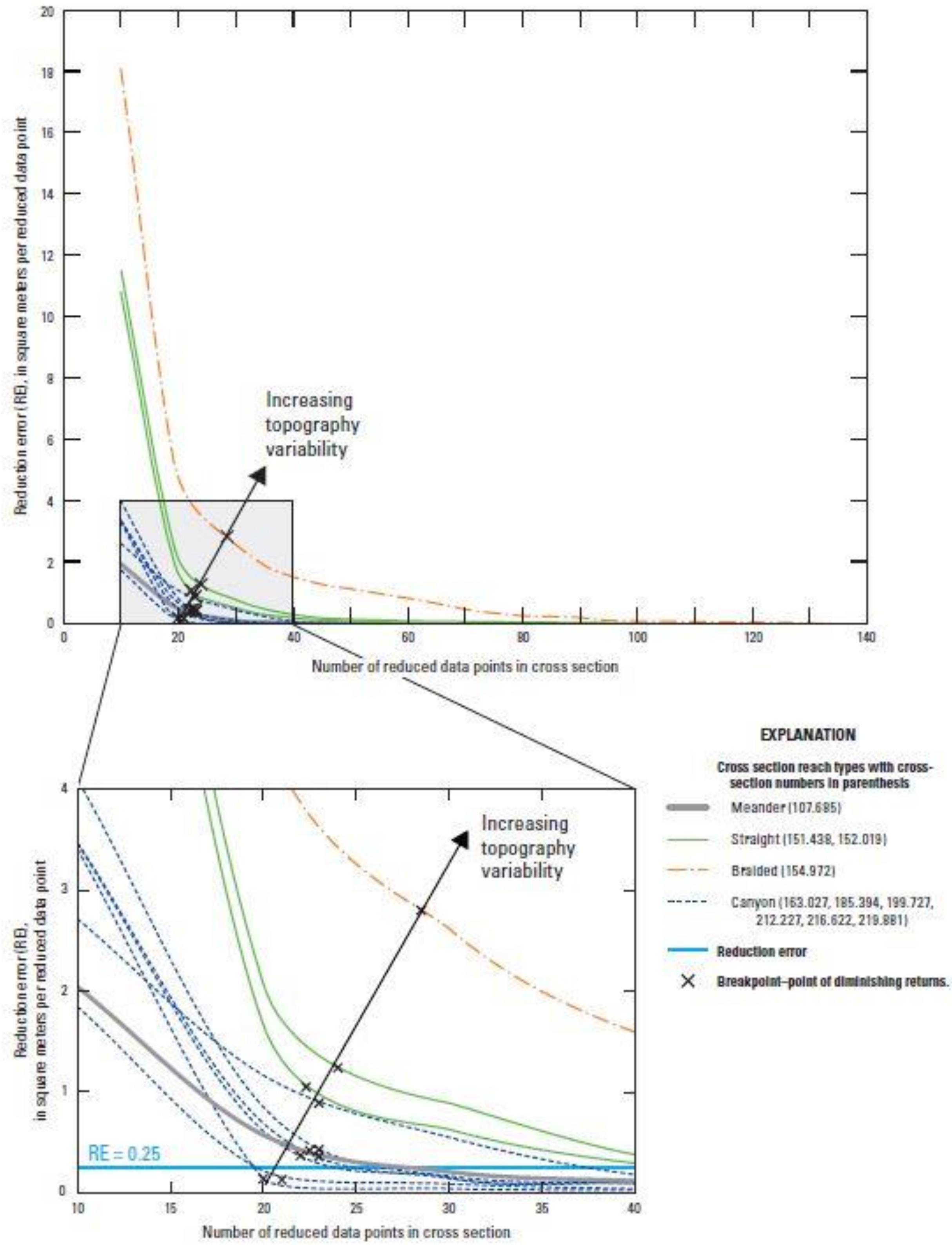


Figure 3.4. Reduction-error (RE) curves for 10 cross sections resulting from the genetic algorithm (GA) reduction method, Kootenai River, Idaho.

lower the RE value as rapidly. The break points for the 10 RE curves are shown in Figure 3.4. The location of the breakpoint for cross-section 107.658 (meander reach) is at an RE of about 0.4 m^2/point and at a number of reduced data points of 22 points, about 1 m^2/point and 22 points for cross-section 152.019 (straight reach), about 3 m^2/point and 29 points for cross-section 154.972 (braided reach), and about 0.4 m^2/point and 23 points for cross-section 163.027 (canyon reach). Breakpoints (points of diminishing returns) for cross sections in the meander and canyon reaches were less than those in the straight and braided reaches because their topography is less varied (Figure 3.4). By 50 data points, all cross sections except 154.972 (braided reach) had RE values equal to or less than 0.25 m^2/point (Figure 3.4). At this value, the RE curves for the meander, straight, and canyon cross sections are nearly asymptotic to the x-axis. Cross-section 154.972 (braided reach) did not reach an RE value of 0.25 m^2/point until 93 data points. This again indicates that cross sections having varied topography such as the braided reach, require more data points to define them adequately.

3.4.2 Hydraulic Modeling Analysis of Reduction Methods

The effects of cross-sectional data-point reduction along a reach were examined on steady-flow water-surface profiles. The 1-D hydraulic-flow model of the Kootenai River in Idaho (Berenbrock, 2005, 2006a) was used to evaluate these effects. The HEC-RAS model, version 4.1 (Brunner, 2010), was used to compute the steady flow profiles (water-surface elevations at cross sections). Only part of the original model (164 cross sections) was used. Cross-sectional data points for only 35 cross sections were reduced—starting at cross-section 149.910 (meander reach) and stopping at cross-section 156.861 (braided reach). This reach was selected because it is the focus of a habitat-restoration project for the recovery of the endangered Kootenai River white sturgeon (Acipenser transmontanus) population, and misrepresentation of cross-sectional data could have major effects on computed steady-flow profiles. This reach included meander, straight, and braided reaches. The model was not extended into the canyon reach because reduction in cross-sectional data points had little effect on the computed cross sections, as shown by the low RE values (Table 3.2 and Figure 3.4). The transition from meander to straight occurs near river mile 151, straight to braided at river mile 153.3, and braided to canyon near river mile 161. A total of 30 simulations were run using the combinations of 5 discharges, 3 data-point reduction levels, and 2 reduction methods. The five discharges (170, 283, 850, 1,416, and 1,982 cubic meters per second, or m^3/s) represent the objective discharges, with respect to habitat restoration (Berenbrock, 2006a), and cross sections were reduced to 10, 20, and 30 data points because they span the breakpoint values for cross sections in figure 3.4. The two data-point reduction methods used were standard and GA.

Results for water-surface elevations are given as differences from the original (Figure 3.5). Generally, results from the simulations showed that the standard-reduced cross sections had greater water-surface differences from the original than did simulations from GA-reduced cross sections. Also differences were greater for the 10-point simulations than for the 20-point and 30-point simulations in respective methods and discharges. For all simulations, the greatest water-surface elevation differences were at cross sections in the braided reach, probably because the reduced cross sections in that reach were not as accurate in representing the original, as shown by the higher RE values. For the standard-reduction simulations, effects from the reduced cross section were seen upstream of river mile 161 (not shown on Figure 3.5), and for the GA-reduced simulations, no effects were seen upstream of river mile 159 (Figure 3.5). In simulations where the RE value for every cross section was equal to or less than 0.25 m^2/point, differences in water-surface elevations (steady flow profile) from the original were very small.

Some cross sections had greater differences in water-surface elevation when more data points were used (Figure 3.5). This is contrary to the paradigm that more data are better. For example, as seen in Figure 3.5A, model results for the standard-reduction method showed that water-surface differences for the 20-point simulation were less than for the 30-point simulation. This occurred in and around cross-section 154.575 (braided reach). The error depends on which cross-sectional data points are captured during reduction and their importance in defining the cross section. The original cross section of 154.575 has four braided channels—three shallow secondary channels and one deep main channel—when total discharge is 170 m^3/s (Figure 3.6). The left most secondary channel was characterized by 6 and 10 data points in the 20-point and 30-point standard-reduced cross sections, respectively (Figure 3.6A). However, water-surface differences were less for the 20-point simulation than for the 30-point simulation (Figure 3.5A). This also occurred at other cross sections, at other discharges, and for both reduction methods. Figure 3.6A shows that both point-reduction levels poorly fit the braided bar landforms in the middle of the cross section, but both point-reduction levels fit the original quite well in the deep main channel (right). The braided bar landforms were reduced in size in the 20-point and 30-point standard reduced cross sections. For the 10-point standard reduced cross section, only two channels were seen—a shallow and wide secondary channel (left), and one deep main channel (right) in which the braided bar landforms from the original cross section are missing.

At discharges of 850 m^3/s and greater, model results for the standard-reduction method showed differences in water-surface elevation were less for the 20-point simulations than for the 30-point simulations in the braided reach, specifically in and around cross-section 156.604 (Figures 3.5C–E). At this cross section, the 20-point and 30-point standard-reduced cross sections excluded data

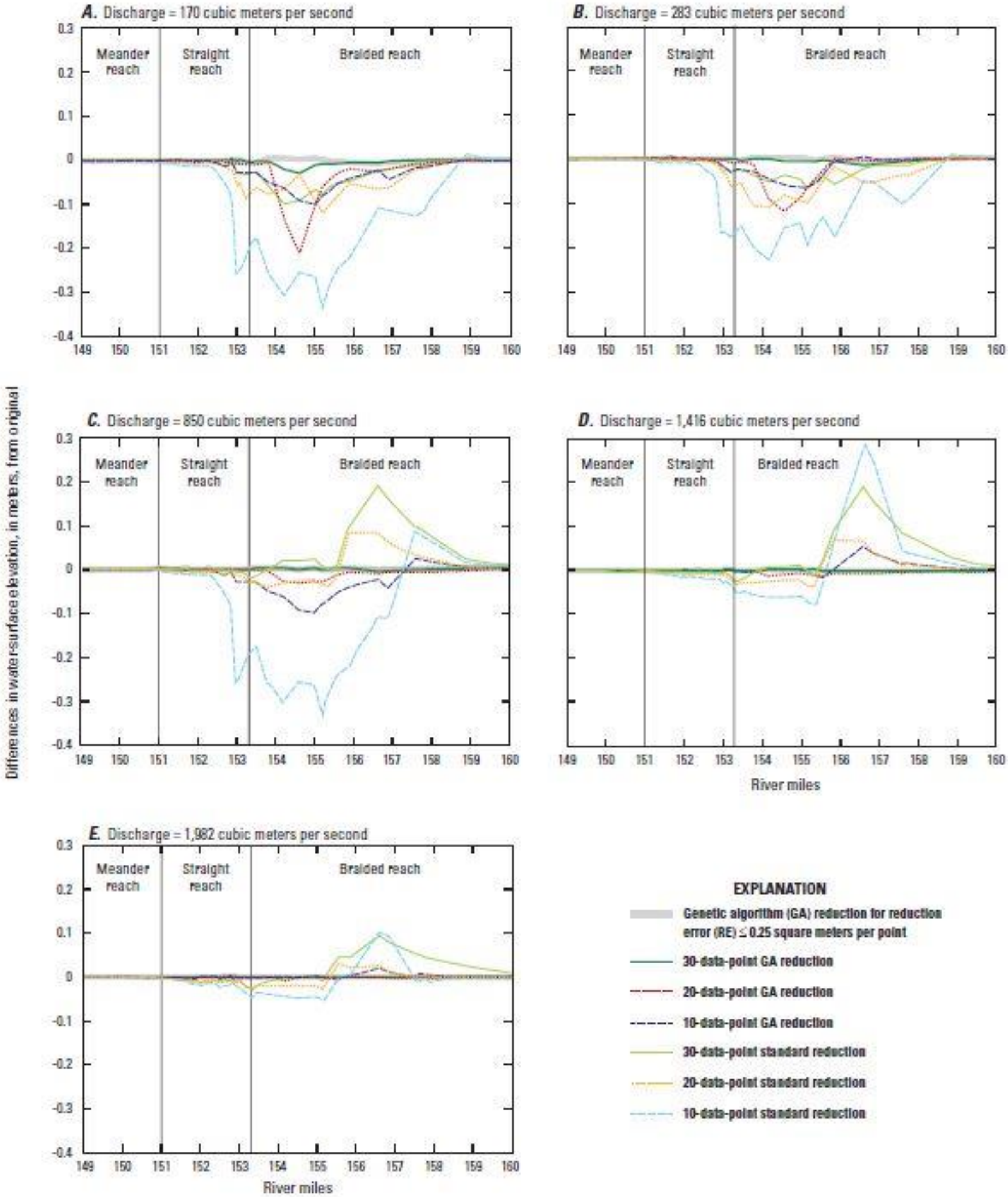
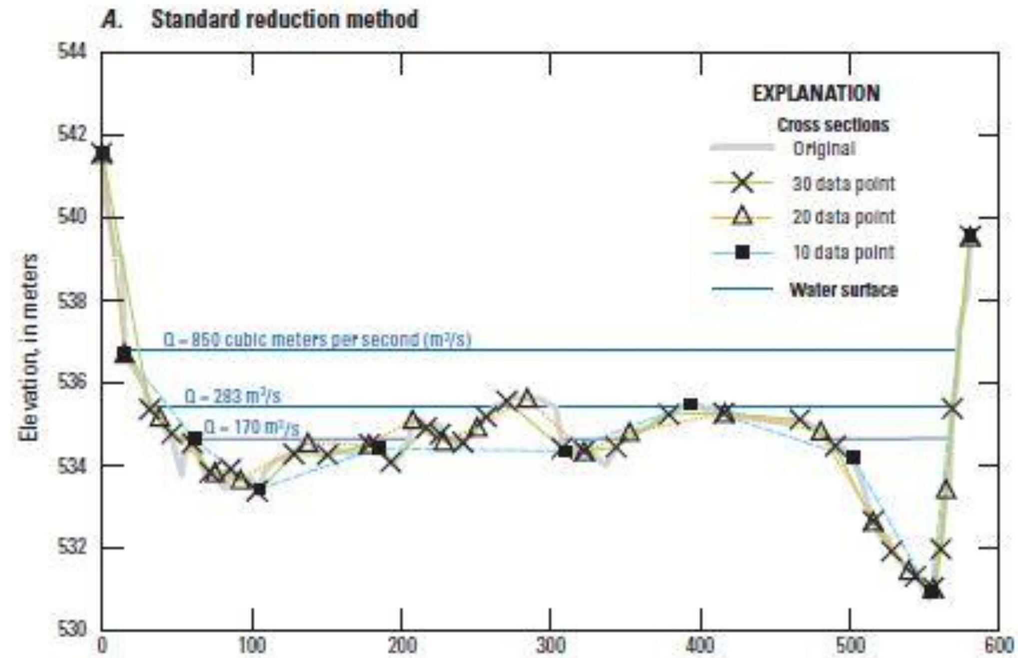


Figure 3.5. Effects of reduced cross sections on simulated water-surface elevation at five river discharges.



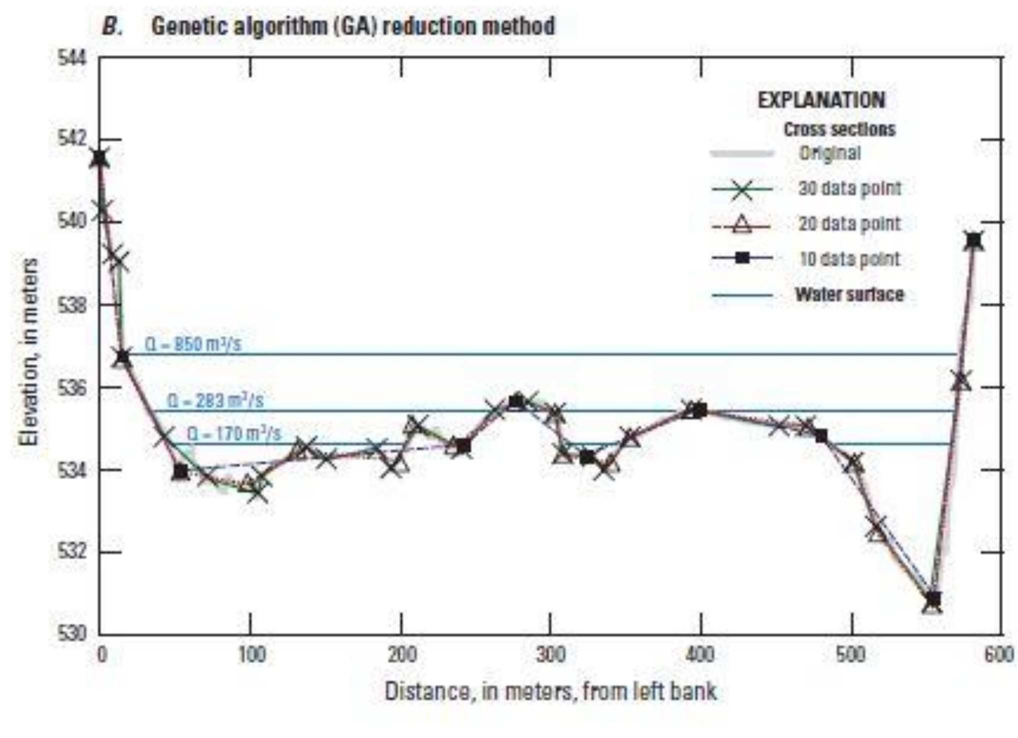


Figure 3.6. Comparisons between the original and 10 point, 20 point, and 30 point reduced cross sections produced by two reduction methods for cross-section 154.575.

points in the thalweg of the secondary channel, which caused the elevation of the reduced cross section in this area to be 0.5 m higher than the original. However, the 10-point standard-reduced cross section included it, but excluded other secondary thalweg points in the cross section. Even though the 30-point standard-reduced cross sections had more points than the 20-point cross section, the 20-point standard-reduced cross section had points in locations that represent the cross-section topography more accurately.

Generally, the GA-reduced cross sections incorporated the shallow channel thalweg and better represented the original cross section. However, at discharges of 170 and 283 m^3/s, water-surface differences for the 10-point GA-reduced simulation were less than the 20-point simulation in and around cross-section 154.575 (Figures 3.5A–B). Although the 20-point reduced cross section resembled the original cross section better than the 10-point reduced cross section (Figure 3.6B), it did worse when it came to step-backwater analysis because the water-surface elevation at cross-section 154.178, a reduced cross section just downstream of this section, contained large errors from the original. At cross-section 154.178, there were too few data points in the secondary channels to define the topographic variability adequately, especially the thalwegs in the secondary channels.

Differences from the original water-surface elevation in the steady-flow profiles were small in the GA-reduced cross sections when an RE of less than or equal to 0.25 m^2/point was used (Figure 3.5). The number of reduced data points for this condition ranged from 20 to 40 points, but several cross sections in the braided reach contained more. Cross-section 154.972 (braided reach), for example, had the most reduced data points (93) in order to meet this condition. This value is less than 100, the limit of most hydraulic and sediment transport models such as HEC-2 and HEC-6. Note that the level of acceptable error in the water-surface elevation or flow depth depends on one's intended use (for example, floodplain maps, sediment transport, or fish habitat).

3.5 Summary

The genetic algorithm (GA) method is a viable approach for reducing data points in a cross section and produced better results than the standard method it was tested against in this study. The original GA by Berenbrock (2006b) [Chapter 2] did not account for varying distances between cross-sectional data points. To account for irregularly spaced data points, the fitness function was modified to calculate the area between the original and reduced-cross-section curves. By using 10 cross sections from the Kootenai River, best fitness values were consistently lower (demonstrating better performance) for the GA runs than for the standard-method and original GA runs. On average, the GA fitness was 39.2 percent lower than the standard method, and for several cross sections was nearly 50 percent lower. The GA-reduced cross sections approximated the shape of the original cross sections better than the standard method and, thus, the GA-reduction method should be used over the standard method.

To provide further insight regarding the spatial distribution of error for the two approaches, visual and hydraulic analyses were completed. Visual analysis (graphs) demonstrated that GA-reduced cross sections approximated the shape of the original cross section better than the standard-reduced cross sections. This was also true for the cross-sectional area. An reduction error (RE) was developed to quantify the difference between the original and reduced cross sections. The RE values decreased as the number of reduced data points increased for both reduction methods, and as expected, RE values were lower (better) for the GA-reduced cross sections than for the standard method. The RE curves were developed for the 10 cross sections on the Kootenai River by using the GA-reduction method, and the breakpoints (points of diminishing returns) found. For the canyon and meander reaches, the breakpoints (about 20 data points) represent the optimal number of points needed in a cross section. However, many more cross-sectional data points were needed for cross sections in the braided and straight reaches as compared to cross sections in the canyon and meander reaches. Also, additional cross sections from other study areas are needed to draw consistent conclusions regarding the number of cross-sectional data points needed for each reach type. The GA-reduced cross sections matched the shape of the original cross section quite well when the RE was equal or less than 0.25 m^2/point, the point at which the RE curves become approximately asymptotic to the x-axis. Most cross sections reached this value at 20 to 40 data points, but cross-section 154.972 (braided reach) did not reach it until 93 data points. More complexly shaped cross sections need greater amounts of data points to define them adequately. Depending on the intended use of a cross section, the number of data points depends on the degree of topographic variability of the cross section and the scale of interest.

This study also investigated the practical consequences of errors due to cross-section reduction on steady-flow profiles. Thirty-five cross sections from the original steady-flow surface-water model of the Kootenai River were used. Cross-sectional data in these cross sections were reduced to 10, 20, and 30 data points for both reduction methods. Results generally indicated that differences were less for cross sections developed by the GA-reduction method than by the standard-reduction method. Also, differences from the original water-surface elevation were usually less as the number of data points in cross sections increased—except for some of the reduced cross sections in the braided and straight reaches. The exception is contrary to the paradigm that more data points are better, and is the result of the standard and GA methods not always having enough points in the secondary channels (braided) to define them adequately. The GA method did not select enough points in the secondary channels because fitness was not bettered (lower value) by doing so. To rectify this problem, the GA needs to be modified so that thalweg points in all channels, the main as well as secondary, are selected. Although the GA method is clearly a major advancement in the reduction of cross-sectional data, there are, of course, limits to its performance. In particular, cross sections having multiple channels, such as braided channels, can be problematic.

3.6 References Cited

Barton, G.J., McDonald, R.R., Nelson, J.M., and Dinehart, R.L., 2005, Simulation of flow and sediment mobility using a multidimensional flow model for the white sturgeon critical-habitat reach, Kootenai River near Bonners Ferry, Idaho: U.S. Geological Survey Scientific Investigations Report 2005–5230, 54 p.

Barton, G.J., Moran, E.H., and Berenbrock, Charles, 2004, Surveying cross sections of Kootenai River between Libby Dam, Montana, and Kootenay Lake, British Columbia, Canada: U.S. Geological Survey Open-File Report 2004–1045, 35 p.

Berenbrock, Charles, 2005, Simulations of hydraulic characteristics in the white sturgeon spawning habitat of the Kootenai River near Bonners Ferry, Idaho: U.S. Geological Survey Scientific Investigations Report 2005–5110, 30 p.

Berenbrock, Charles, 2006a, Simulations of hydraulic characteristics for an upstream extension of the white sturgeon habitat of the Kootenai River near Bonners Ferry, Idaho—A supplement to Scientific Investigations Report 2005–5110: U.S. Geological Survey Scientific Investigations Report 2006–5019, 17 p.

Berenbrock, Charles, 2006b, A genetic algorithm to reduce stream channel cross section data: Journal of the American Water Resources Association, v. 42, no. 2, p. 387–394.

Brunner, G.W., 2010, HEC-RAS, River analysis system hydraulic reference manual: U.S. Army Corps of Engineers Hydrologic Engineering Center, CPD-69, January 2010, version 4.1, 417 p.

Burnham, M.W., and Davis, D.W., 1990, Effects of data errors on computed steady-flow profiles: American Society of Civil Engineers Journal of Hydraulic Engineering, v. 116, no. 7, p. 914–929, DOI: http://dx.doi.org/10.1061/(ASCE)0733-9429(1990)116:7(914).

Castellarin, A., Di Baldassarre, G., Bates, P.D., and Brath, A., 2009, Optimal cross-sectional spacing in Preissmann Scheme 1D hydrodynamic models: Journal of Hydraulic Engineering, v. 135, no. 2, p. 96–105.

Chow, V.T., 1959, Open-channel hydraulics: New York, McGraw-Hill, 680 p.

Columbia Basin Inter-Agency Committee, 1965, River mile index, Kootenai River, United States, Kootenay River, Canada, Columbia River Basin, Idaho, Montana, British Columbia: Columbia Basin Inter-Agency Committee, Hydrology Subcommittee, November 1965, 49p [114].

Cook, Aaron, and Merwade, Venkatesh, 2009, Effect of topographic data, geometric configuration and modeling approach on flood inundation mapping: Journal of Hydrology, v. 377, p. 131–142.

Czuba, C.R., and Barton, G.J., 2011, Updated one-dimensional hydraulic model of the Kootenai River, Idaho—A supplement to Scientific Investigations Report 2005–5110: U.S. Geological Survey Scientific Investigations Report 2011–5128, 36 p.

Davis, L., 1991, Hybridization and numerical representation, in Davis, L., ed.,The handbook of genetic algorithms: United Kingdom, Van Nostrand Reinhold, p. 61–71.

Federal Emergency Management Agency, 2007, Guidelines and specifications for study contractors: Federal Emergency Management Agency, Publication 37, Washington, D.C., U.S. Government Printing Office, p. 174, accessed August 25, 2011, at http://www.fema.gov/library/viewRecord.do?id=2238.

Fisher, J.C., 2013, Optimization of water-level monitoring networks in the eastern Snake River Plain aquifer using a kriging-based genetic algorithm method: U.S. Geological Survey Scientific Investigations Report 2013-5120 (DOE/ID-22224), 74 p., http://pubs.usgs.gov/sir/2013/5120.

Goldberg, D.E., 1989, Genetic algorithms in search, optimization and machine learning: Reading, Mass., Addison Wesley, 412 p.

Grefenstette, J.J., 1990, Genetic algorithms and their applications, in Kent, A., and Williams, J.G., eds., The encyclopedia of computer science and technology, 21 (Supp. 6): New York, Marcel Dekker, p. 139–152.

Harrelson, C.C., Rawlins, C.L., and Potyondy, J.P., 1994, Stream channel reference sites—An illustrated guide to field technique: U.S. Department of Agriculture, Forest Service, Rocky

Mountain Forest and Range Experiment Station, Gen. Tech. Rep. RM-245, Fort Collins, Colo., 61 p.

Koza, J.R., 1992, Genetic programming—Vol. 1, on the programming of computers by means of natural selection (complex adaptive systems): London, A Bradford Book, 819 p.

Mitchell, M., 2002, An introduction to genetic algorithms: Cambridge, Mass., The MIT Press, (8th printing), p. 209.

Moran, E.H., and Berenbrock, Charles, 2003, GPS—Time saver and functional: U.S. Geological Survey Western Water Watch, v. 1, no. 1, p. 6–7.

Pasternack, G.B., Wang, C.L., and Merz, J.E., 2004, Application of a 2D hydrodynamic model to design of reach-scale spawning gravel replenishment on the Mokelumne River, California: Journal of River Research and Applications, v. 20, p. 205–225.

Samuels, P.G., 1989, Backwater length in rivers: Proceedings of Institution of Civil Engineers, pt. 2, no. 87, December, p. 571–581.

Synder, E.B., and Minshall, G.W., 1996, Ecosystem metabolism and nutrient dynamics in the Kootenai River in relation to impoundment and flow enhancement of fisheries management: Idaho State University, Stream Ecology Center, variously paginated.

Traver, R.G., and Miller, A.C., 1993, Open channel interpolation of cross sectional properties: Journal of the American Water Resources Association, v. 29, no. 5, p. 767–776. DOI: http://dx.doi.org/10.1111/j.1752-1688.1993.tb03236.x.

Travis, Quentin, and Lokey, Burke, 1999, Minimizing errors due to cross-section point reduction, in American Society of Civil Engineers Proceedings of the 26th Annual Water Resources Planning and Management Conference (WRPMD), Tempe, Ariz., June 6–9, 1999, Wilson, E.M., ed.: Reston, Va., ASCE, 978-0-7844-0430-0 or 0-7844-0430-5, 1999, chap. 3G64, p. 1-13, DOI: http://dx.doi.org/10.1061/40430(1999)142.

CHAPTER 4. DECIMATION OF RIVER GEOMETRY DATASETS USING GENETIC ALGORITHMS FOR USE IN SURFACE-WATER MODELS

4.1 Abstract

Surface-water hydraulic models require accurate representation of the river and (or) floodplain geometry, and the resulting dataset can be too large for most one-dimensional models. Selecting the appropriate data points to use in the model from among the hundreds or thousands of data points can be both challenging and tedious.

The problem is even more challenging for multi-dimensional datasets such as bathymetry or datasets produced from using Light Detection and Ranging (LiDAR). These datasets are typically used in two- and three-dimensional surface-water models. The standard procedure usually consists of gridding, which generalizes the terrain—missing the high and low elevations. To more effectively perform this task, the Genetic Algorithm (GA) computer program was modified to decimate (i.e. reduce) multi-dimensional datasets. The program was then used to decimate data for a hypothetical example and data taken from an actual bathymetric and LiDAR dataset. Results indicated that the program successfully reduced the data. Terrains produced by the GA are fairly representative of the original data, and volumetric differences from the original terrain were smaller for the GA produced terrain than standard procedures of decimating LiDAR. Results also showed that near-optimal results could be obtained in a single GA run.

4.2 Introduction

Traditionally, surveyed stream profiles and cross sections and (or) Light Detection and Ranging (LiDAR) scans are used to obtain data that describe the channel shape of streams and floodplains and are used in mathematical computer models to simulate flow hydraulics and sediment transport in a stream. A cross section is a series of data pairs (distance and elevation) along a straight line that is roughly perpendicular to streamflow. These datasets can be large. For example, data for approximately 500 cross sections were collected on the Kootenai River in northern Idaho. The number of data points for each cross section ranged from about 500 to more than 2,000 points (Barton *et al*., 2004; and Moran and Berenbrock, 2003). LiDAR and bathymetric datasets present an even larger problem. These datasets are usually used in two- and three-dimensional surface-water models. For example, a raw LiDAR dataset from the Lower Coeur d’Alene River for a 1 kilometer (km) by 1 km area consisted of more than 350,000 data points (x, y, and z). If a 10 km x 2 km reach of this river and floodplain were selected to be modeled, the dataset would consist of more than 6 million data points.

For flood insurance studies, the Federal Emergency Management Agency (FEMA) indicates that cross-section points should be located at breaks in the ground slope and should approximate the actual shape of the channel and (or) floodplain (FEMA, 1995). There is no point minimum as long as the actual shape of the channel and floodplain are well defined. The FEMA requirement applies to cross-section data, but is a reasonable requirement for multi-dimensional datasets such as digital elevation models (DEM) and bathymetric and LiDAR datasets.

These large datasets can be reduced to smaller, less dense datasets that are easier to work with—a process called decimation. Previous investigators have developed automatic decimation procedures. Chen and Guevara (1987) presented an automatic point selection procedure called “very important points” (VIP) for selecting points directly from DEMs or triangulated irregular networks (TIN). VIP is essentially a high-pass filter that selects data based on the distance a point is from the 4 lines connecting its diametrically opposed neighbors. Factors such as slope and proximity guide the selection of data points. This procedure has several potential problems. First, the peak of a small, sharp hill will be considered more important than a peak of one that is large, yet slopes gently. Secondly, the VIP procedure chooses nearly all of the points along valleys and ridges. It is desirable to capture these important features. However, if a ridge or valley follows a straight line, that feature may be represented by two or a few points instead of by many points. Another disadvantage is that the VIP procedure chooses nearly all of the points along the boundary, thereby overly defining the margins of the study area.

Berenbrock (2006) developed a genetic algorithm (GA) for decimating cross-section data while ensuring the integrity of the cross-section geometry. The program successfully decimated cross-section data and fit better to the original data than standard procedures—selection of a data point a set distance apart or selecting every 10th, 20th, or nth point. On average, differences between the original cross sections and the GA-produced cross sections were about 30 percent less than cross sections obtained using standard procedures.

The purpose of this paper is to describe application of a GA to the decimation of LiDAR data and demonstrate that the GA is a viable approach. The GA described in this paper uses a single objective optimization scheme for decimating LiDAR and bathymetric data. A hypothetical example and a case study with actual data are presented to validate the genetic algorithm. The hypothetical example is a square dataset composed of 961 x, y, and z data points. The case study, Coeur d’Alene

River and Floodplain Application, consists of bathymetric data from the river and LiDAR data from the floodplain.

4.3 Genetic Algorithm

Genetic algorithms apply the ideas of Darwin's theory of evolution: individuals more adapted to the environment have a better chance to survive ("survival of the fittest"). Genetic operators such as selection, reproduction (crossover), and mutation are used to improve a population. Holland (1975) was the first to apply these operators. Since then, GAs have been applied successfully to many water resource problems (McKinney and Lin, 1992; Cieniawski *et al*., 1995; Vink and Schot, 2002; Chen, 2003; Knaapen and Hulscher, 2003). Vink and Schot (2002) and Chen (2003) indicated that GAs are capable of handling highly nonlinear, discontinuous, nondifferentiable, interdependent, and nonconvex problems where many other techniques such as linear and nonlinear programming, heuristic, etc. cannot. Simulated annealing relies on a weighted objective function that only finds one optimal solution per iteration, whereas, GAs are able to find multiple convex or nonconvex solutions in a single iteration (Cieniawski *et al*., 1995).

In a GA, a population of individuals is created and evolved until an individual is obtained that best represents the salient features of the dataset or until a specified number of generations is met. Individuals exhibit traits that can be inherited. For the problem at hand, traits are combinations of bathymetry and LiDAR data (terrain) points that are either included or excluded (removed) from the dataset to create individuals. The inclusion or exclusion of traits might be represented as a string of numbers or chromosomes. For example, the binary string "`10101111`" might represent an individual where the second and fourth chromosomes (data points) are excluded (a value of 0) and the other chromosomes included (a value of 1). The fitness of an individual (thus a fitness of traits or data point combinations) is represented as a value from the fitness or objective function.

Chromosomes are passed from parents to offspring through a process called "crossover" in which randomly selected chromosomes of the parents are combined or swamped to create children. For example, two individuals (parents) with traits `10101010` and `11111111` are crossed at the sixth through the eighth chromosome to produce two children, `10101`**`111`** and `11111`**`010`**.

Individuals with superior fitness values are more likely to be allowed to reproduced, although this rule is often relaxed or altered to increase the genetic diversity of the population. Additional diversity is also introduced through mutation in which randomly selected chromosomes of the children are changed (reassigned) to create a combination of traits not present in either parent. A child that has been mutated may exhibit better or poorer fitness than the parents. Mutation rates are set very low.

Repeating the selection, crossover, and mutation processes over many generations under conditions controlled by a fitness function results in better-fit individuals and the overall population improves. The concept is described as pseudo code instructions in Figure 4.1. Instructions within the “do-loop” are repeated until some time (number of generations) has elapsed, a threshold criterion has been met, or best individual fitness has reached a plateau. Goldberg (1989), Grefenstette (1990), Davis (1991), and Mitchell (2002) provide more complete discussions of the GA concept.

- Generate a random population, a collection of strings
- Do for some time
    - Evaluate each individual (string) in the population using the fitness function [f(i)]
    - Select two individuals (parents) from the population according to their fitness (the better the fitness, the greater the chance to be selected)
    - Allow for crossover–the parents form two new individuals (children). If no crossover was performed, the children are exact copies of the parents
    - Allow for mutation
    - Place children in new population
- End do

Figure 4.1. Pseudo code for a simple genetic algorithm.

4.4 Program Description

The cross section GA program (Berenbrock, 2006) was generalized and modified to create a new binary GA program capable of decimating LiDAR and (or) bathymetric data. The program uses strings of ones (1s) and zeros (0s) to represent the traits of individuals (combinations of present or omitted data points) from the dataset. The position of chromosomes within the strings and the lengths of the strings are fixed on the number of original data points (n). Ones (1s) are inserted into chromosome positions corresponding to data points that define the convex hull for the dataset. A convex hull is a set of points that define the extent or boundary of the dataset in n-space.

An individual represents a collection of data points with values of x, y, and z on the terrain. The GA program uses TINs to represent or describe the terrain because data points can be irregularly spaced, whereas in a DEM, points must be regularly spaced. A TIN is composed of three points to form a triangular arrangement. An individual's volume is the volume beneath the terrain or volume of the TINs, which is calculated by computing the volume in each TIN and summing the TIN volumes.

The program computes an individual's fitness as the difference between the individual's volume and the original volume. A limit on the number of included data points was accomplished by imposing a two-conditional fitness function (Berenbrock, 2006). The two-conditional fitness function is:

$$f(i) = \begin{cases} \left| \left(\sum_{j=1}^{n} v_j \right) - v_{original} \right| & \text{if } inc_i \leq plimit \\ \left| \left(\sum_{j=1}^{n} v_j \right) - v_{original} \right| \cdot 10^{(inc_i - plimit)} & \text{otherwise} \end{cases} \qquad \text{(Equation 4.1)}$$

where f is the value of the fitness for individual i, n is the number of TINs in individual i, v_j is the volume of the TIN for trait j, $v_{original}$ is the volume of the original dataset, inc_i is the number of included data points in individual i, and plimit (point limit) is the maximum number of points to be included in the decimated dataset. The first condition applies if the number of included points in an individual is less than or equal to the point limit; otherwise, the second condition applies. The GA then minimizes the fitness function (*Minimize* f(i)) to identify the best-fit individual or optimal dataset.

To reduce the number of generations and program run time, an initialization technique was used to generate the initial population. This technique sets the probability of a data point being included to the plimit divided by the string length (n). By narrowing the search space, the GA then focuses its search on approximately the right number of points and does not explore unlikely solutions (including all points, including no points, and so on) (Berenbrock, 2006).

The GA is generational; two elite individuals are copied each generation. Tournament selection is used as the reproduction methods, and tournament size is 3. In the validation test, several different crossover rates (P_c), population sizes, and number of generations at first were used. The mutation rate (P_m) was set to 1/n, a standard rate suggested by Reed *et al*. (2000, 2003). A general description of the genetic algorithm program for LiDAR and bathymetric decimation is given in Appendix H, and a listing of the program code is given in Appendix I.

4.5 Program Validation

Data representing a hypothetical example and data taken from actual bathymetric and LiDAR datasets collected on the Coeur d'Alene River and Floodplain were used to validate the binary GA. The plimit for the hypothetical example was arbitrarily set to 15 percent of the hypothetical data (144 data points) and 10 percent for the Coeur d'Alene data. A mutation rate of 1/n was used (Reed *et al*., 2000, 2003).

4.5.1 Hypothetical Example

The hypothetical dataset consists of 961 regularly spaced points (x, y, z) on a 31 x 31 grid, spaced 16 m apart (Figure 4.2A) (see Appendix J for the x-y-z data points). The GA was run 10 times with a population size of 80 individuals, for 100 generations, using a crossover rate of 0.30 and plimit of 15 percent (144 data points). The number of included data points ranged from 139 to 144 (Table 4.1). Because this GA seeks to minimize the fitness function (Equation 4.1), the run with the lowest fitness value is the superior or "best" run for the given fitness function and domain. In this case, Run 8 had the superior fitness value (166 m^3) given the 10 runs. Even though the GA relies on randomness in sampling and in creating the initial population, the range in best fitness for all runs was small as compared to the range in average fitness and root mean square error (RMSE) fitness. The small range in best fitness suggests that near optimal results could be obtained in a single GA run. An example output listing from the cross-section reduction program for the hypothetical dataset is given in Appendix K.

Figure 4.2 shows a colorized relief terrain representation of the TIN produced by the hypothetical (original), VIP, LATTICETIN and 10 GA runs. VIP and LATTICETIN are two commonly used TIN procedures. The number of points used in VIP and LATTICETIN was not allowed to exceed the plimit (144 points). From a qualitative viewpoint, the VIP run (Figure 4.2B) is a very poor representation of the original dataset under this condition; only 20 points were used to define the interior terrain while 120 points were used to define the boundary. The terrain is under emphasized in the interior and over emphasized at the boundary. As discussed earlier, VIP retains most if not all of the boundary points in its solution. The LATTICETIN run (Figure 4.2C) preserved the major features fairly well but also had inaccurate features near the boundaries especially near the stream. For example, the stream near the southern boundary extended too wide probably because only a few data points are used to define the terrain in this area. For the LATTICETIN run, 37 data points

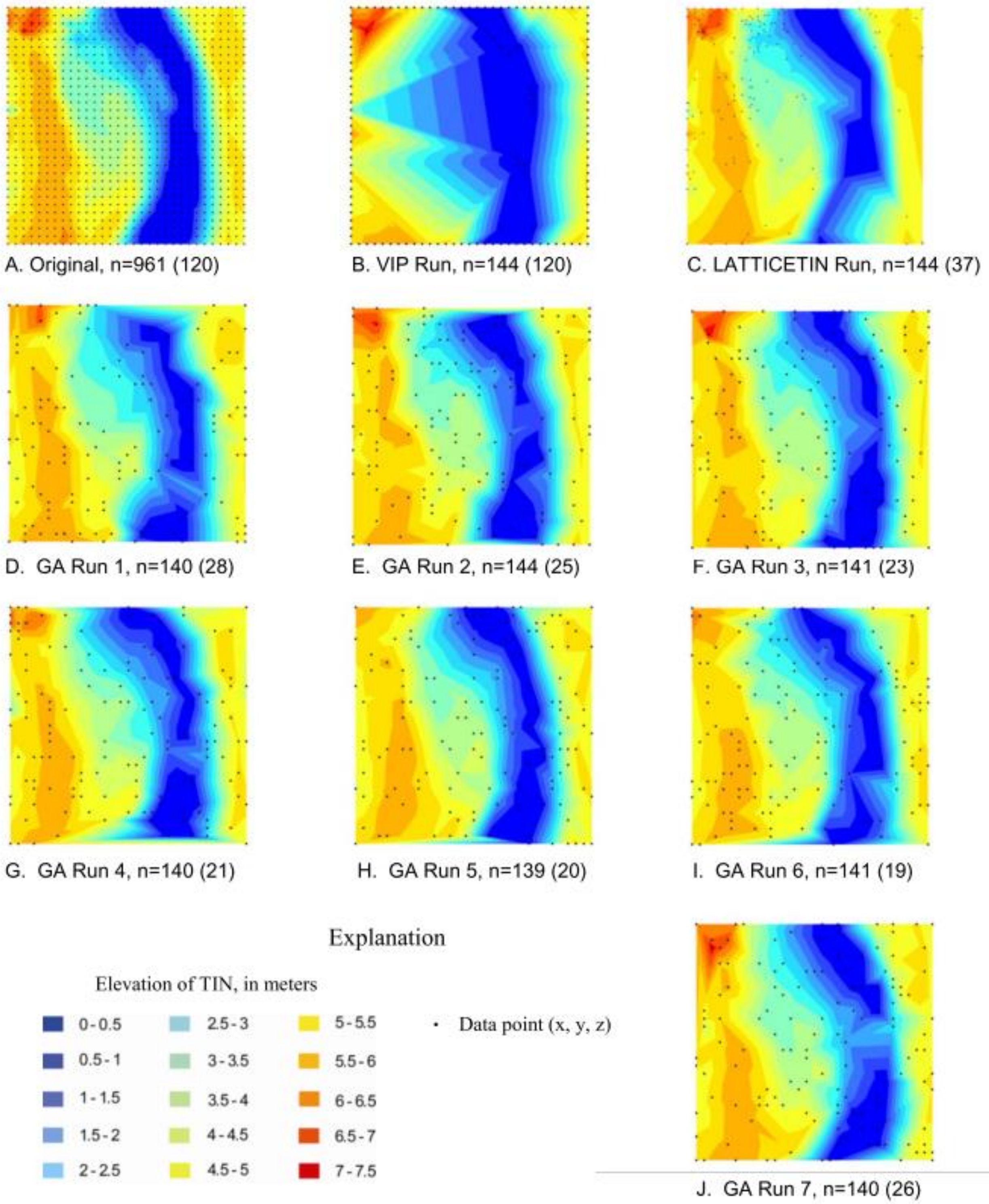
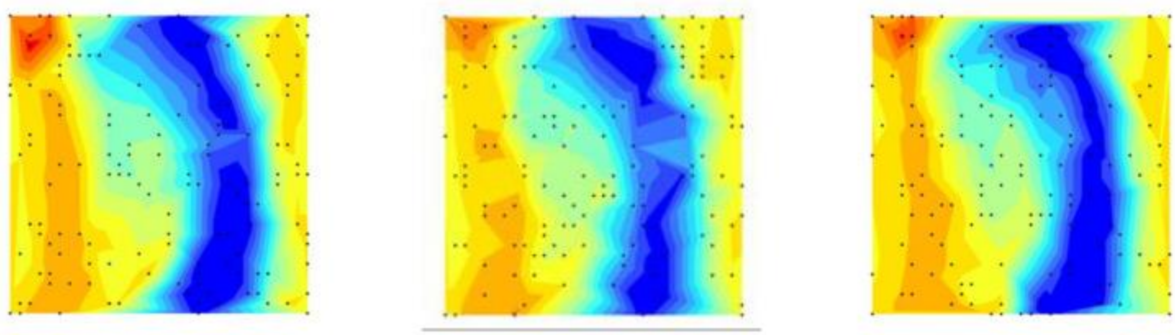


Figure 4.2. Terrain from the original dataset, VIP run, LATTICETIN run, and GA runs for the hypothetical example. (n is the number of data points, and value inside the parenthesis is the number of points located on the boundary)



K. GA Run 8, n=139 (21) L. GA Run 9, n=143 (19) M. GA Run 10, n=140 (20)

Figure 4.2.—Continued.

Table 4.1. Best fitness value, number of data points, and number of points on the boundary for the hypothetical example.

Run	Number of data points	Best fitness value (cubic meters)	Number of points on boundary	Figure no.
Original	961	--	120	2A
VIP Run	144	--	120	2B
LATTICETIN Run	144	--	37	2C
[1]GA Run 1	140	361	28	2D
[1]GA Run 2	144	355	25	2E
[1]GA Run 2	141	771	23	2F
[1]GA Run 2	140	518	21	2G
[1]GA Run 2	139	271	20	2H
[1]GA Run 2	141	260	19	2I
[1]GA Run 2	140	332	26	2J
[1]GA Run 2	139	166	21	2K
[1]GA Run 2	143	1,110	19	2L
[1]GA Run 2	140	460	20	2M

[1]The crossover rate (P_c) was set to 30 percent, the mutation rate (P_m) was set to 1/n, plimit was set to 15 percent or 144 points, and n is the number of data points in the original data (961).

are located on the boundary. In the 10 GA runs, the major topographic features are preserved fairly well (Figures 4.2D through 4.2M). Inaccurate features are also seen near the boundaries especially near the stream. This is probably due to the GA runs containing fewer points on the boundary (ranging from 19 to 28 points) which is less than the LATTICETIN run. The fewer points especially near the stream caused the stream to not fully extend to the boundary of the study area or caused the stream to extend too widely, similar to the LATTICETIN run. For example, the stream in runs 1 (Figure 4.2D), 2 (Figure 4.2E), and 10 (Figure 4.2M) did not fully extend to the northern boundary of the study area, and the stream in run 4 (Figure 4.2G) did not fully extend to the southern boundary.

The best, average, and RMSE fitness values for Run 8 are shown in Figure 4.3. The best fitness decreased as the number of generations increased indicating that the program functioned correctly for this large dataset; best fitness decreased from 1,540 m^3 to 518 m^3. To measure the distribution of the population, the average and RMSE fitness values are calculated for each generation. If the average and RMSE fitness values are large, then the diversity between individuals is high; if average and RMSE fitness values are small, the diversity is low. The genetic algorithm might not perform well if the diversity is too high or too low. For example, if the population's diversity is too low, the genetic operator "crossover" becomes almost ineffective and the population probably will have a hard time escaping from the local optimum where the population has converged too. On the other hand, if the population's diversity is too high, an optimal solution might not be reach or take a long time (number of generations) to reach. The large range in average and RMSE fitness values shown in Figure 4.3 indicates that the population is highly diverse. These values also fluctuated from one generation to the next indicating changes in the population. Similar results were observed in the other nine runs.

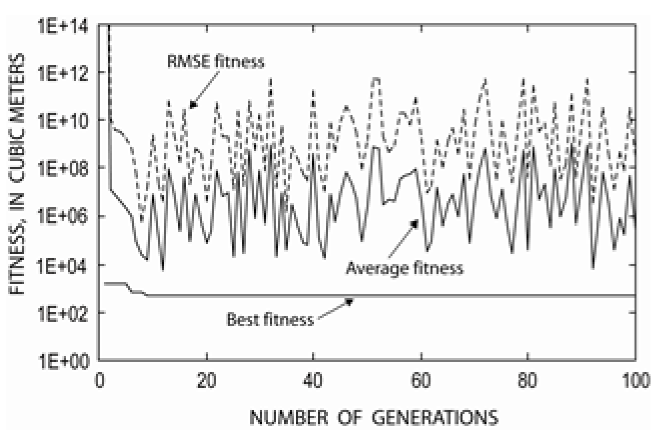


Figure 4.3. Best fitness, average fitness, and root mean squared error (RMSE) fitness for each generation of GA Run 8 for the hypothetical example.

Volumetric results [hypsometric curves of volume] from the VIP, LATTICETIN, and GA runs and the original are shown in Figure 4.4. The volume for the original and each run is calculated by computing the volume for each TIN in the dataset and then summing those TIN volumes. Results for the VIP run were much lower than the original indicating that the VIP-produced terrain is inaccurate in this example. The volumetric differences from the original for this run are quite large (Figure 4.4B). The number of interior data points (20 points) in this run was not enough to produce a reasonable volumetric representation of the original. Volumetric results [hypsometric curves for volume] for the LATTICETIN run closely tracked the original (Figure 4.4A). Differences in volume from the original and the LATTICETIN run are small for heights greater than 1 m (Figure 4.4B); differences increase when heights are less than 1 m, which can be seen in the unnatural shape of the stream (Figure 4.2C). Results for the GA runs showed that volumes closely track the original and are bunched together in a narrow band (Figure 4.4A), which also supports that near optimal results could be obtained in a single GA run.

Differences in volume from the original in the GA runs are also small (Figure 4.4B). The greatest differences occurred in the middle heights and are probably cause by the fitness function in the GA basing its volumetric calculations only at a 0 m height. These differences are small but are evidence of the deficiency in the GA. A fitness function is needed that integrates the entire volume along the curve.

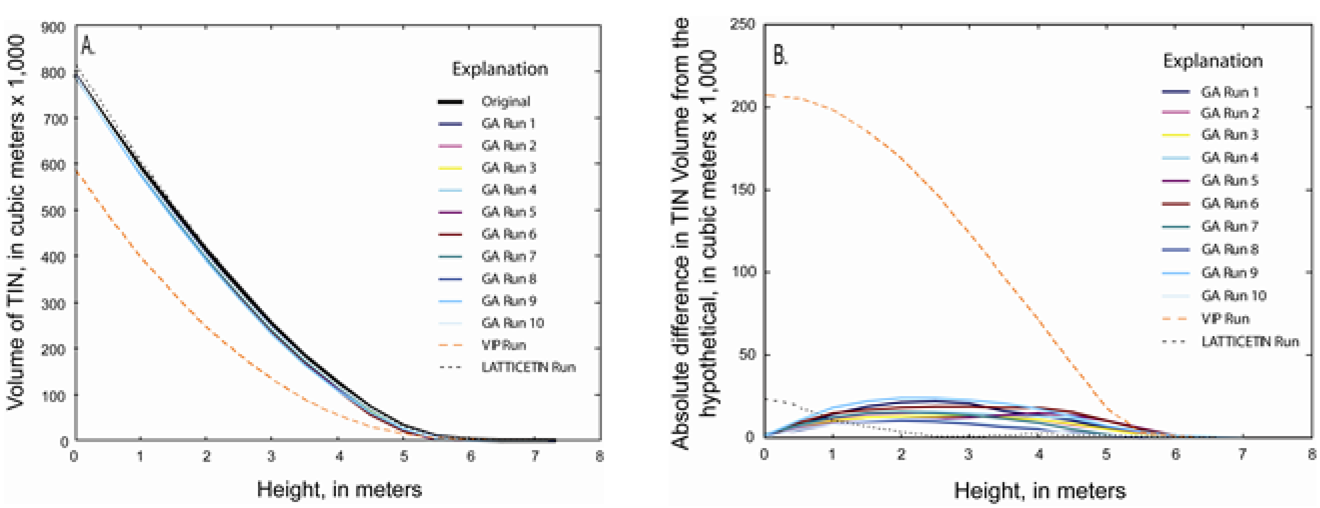


Figure 4.4. (A) TIN volumes for the original, VIP, LATTICETIN, and GA runs and (B) volumetric differences from the original for the hypothetical example.

4.5.2 Coeur d'Alene River Application

For the real world application, a section was extracted from LiDAR and bathymetric datasets from the Coeur d'Alene River and Floodplain near river mile 156 (Berenbrock and Tranmer, 2008). The subsection, 1073 m (0.67 mi) wide and 952 m (0.60 mi) long, contains 10,080 points. The GA initially was run with the same parameter values as in the hypothetical example, except plimit was arbitrarily set to 10 percent (1,008 points).

The GA was run 10 times for the Coeur d'Alene River dataset. The best fitness and number of points for each run are shown in Table 4.2. The run with the lowest fitness value is the superior or "best" run because this GA seeks to minimize the fitness function (Equation 4.1). The range of best fitness for the GA runs varied from 165 m^3 to 1,390 m^3. The fourth GA run had the superior fitness of all the runs. The best, average, and RMSE fitness values for Run 4 are shown in Figure 4.5. Again, as expected, the best fitness decreased as the number of generations increased indicating that the program functions correctly for large datasets. The best fitness decreased from 12,200 m^3 to 165 m^3. The large range in average and RMSE fitness values indicates that the population is highly diverse, and the fluctuations from one generation to the next show the changing diversity in the population. Similar results were observed in the other nine runs.

Figure 4.6 shows a colorized relief terrain representation of the TINs produced by the original dataset, VIP run and the GA run 4. From a qualitative viewpoint, the major topographic features for the VIP run are generally preserved (Figure 4.6B). However, several large discontinuities in the terrain occurred in the study area. The discontinuity in the river could have significant impacts on river flows if this dataset were used in a multi-dimensional model, and the discontinuity of the hill could have impacts on floodplain flows. Neither the Fourth of July Creek, the road/levee north of the river, nor the road south of the river is distinguishable (Figure 4.6B). If the VIP had fewer points located along its boundary (262), there would be more points available for the interior that would give more definition to the interior especially to areas in the river. A LATTICETIN run was not conducted because the LATTICETIN requires regularly spaced data throughout the domain. The LiDAR and bathymetric datasets used in the subsection are regularly spaced but the datasets do not line up to one another to create a regularly spaced dataset. Together the dataset is considered irregularly spaced, and thus, a LATTICETIN run could not be performed. This is a distinct advantage of a TIN and a disadvantage in LATTICETIN that requires regularly spaced data. The major topographic features for the fourth GA run are generally preserved even along the boundary (41 points located on the boundary) (Figure 4.6C) unlike what happen in the hypothetical example. A plausible explanation is that there are enough

Table 4.2. Best fitness value, number of data points, and number of points on the boundary for the Coeur d'Alene River application.

Run	Number of data points	Best fitness value (cubic meters)	Number of points on boundary	Figure no.
Original	10,080	--	396	4.6A
VIP Run	1,010	--	262	4.6B
[1]GA Run 1	1,003	473	52	--
[1]GA Run 2	993	1,310	48	--
[1]GA Run 3	990	1,390	41	4.6C
[1]GA Run 4	1,003	165	41	--
[1]GA Run 5	1,008	903	41	--
[1]GA Run 6	1,008	856	45	--
[1]GA Run 7	962	1,360	50	--
[1]GA Run 8	1,005	242	46	--
[1]GA Run 9	1,008	812	47	--
[1]GA Run 10	1,004	940	45	--

[1]The crossover rate (P_c) was set to 30 percent, the mutation rate (P_m) was set to 1/n, plimit was set to 10 percent or 1,008 points, and n is the number of data points in the original data (10,080).

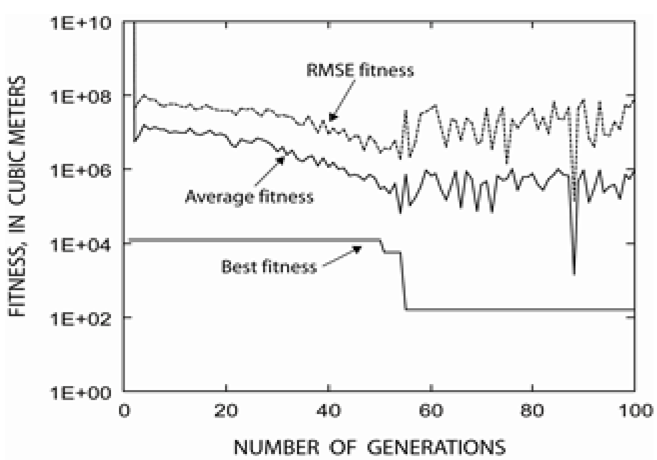


Figure 4.5. Best fitness, average fitness, and root mean squared error (RMSE) fitness for each generation of GA Run 4 for the Coeur d'Alene River.

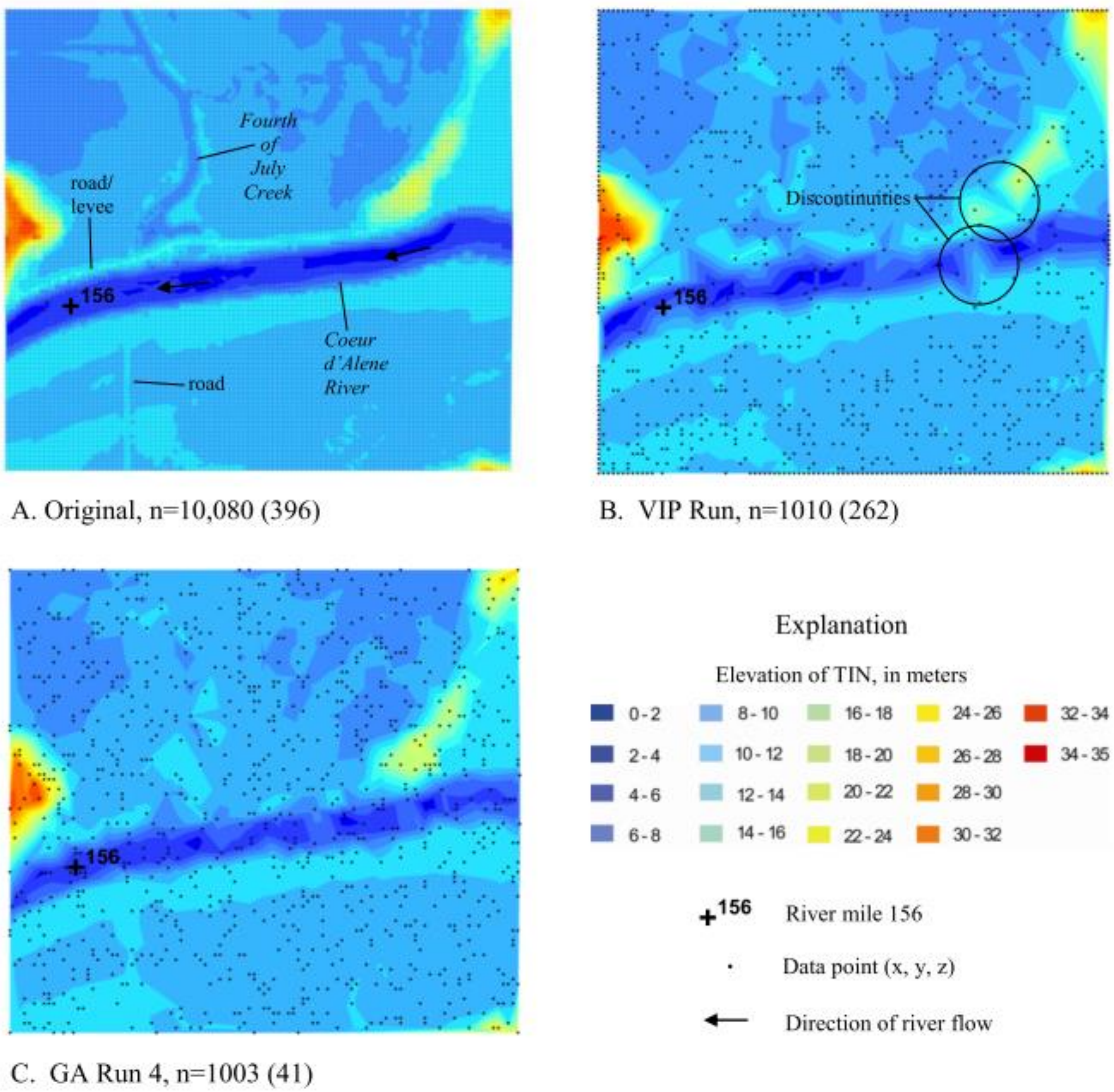


Figure 4.6. Terrain from the original, VIP run and GA run 4 for the Coeur d'Alene River application. (n is the number of LiDAR points, and number inside parenthesis is the number of points located on the boundary).

points near the boundary to obtain a good representation. Neither the Fourth of July Creek nor the road/levee north of the river is distinguishable in the fourth GA run, but the road south of the river is somewhat distinguishable (Figure 4.6C).

Volume was calculated for these runs. These volumetric results [hypsometric curves of volume] also closely tracked the original. Again, results from all GA runs fell within a narrow band similar to results from the hypothetical example, which supports that near-optimal results could be obtained in a single run. Volumetric differences were also calculated for these runs, but only the VIP run and the fourth GA run, the superior GA run, are shown in Figure 4.7. Differences from the original in the VIP run were greater at all heights than in the fourth GA run. The largest differences between the two runs occurred when the height ranged from 10 m to 20 m.

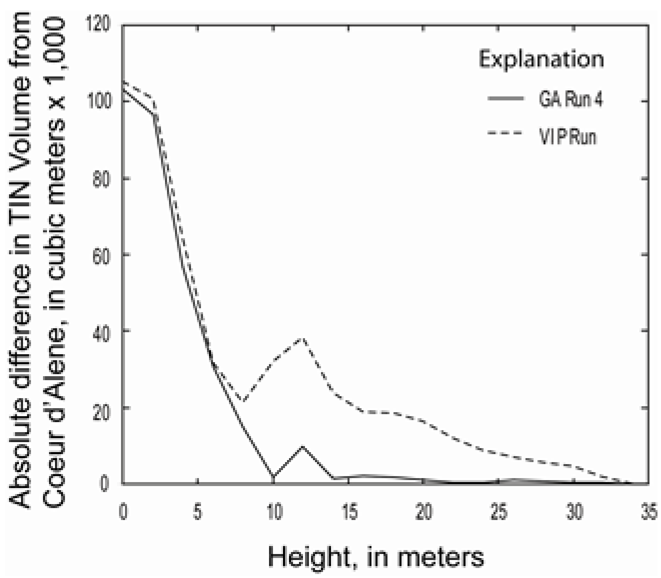


Figure 4.7. Volumetric differences in TIN volumes from the original to the VIP Run and GA Run 4 for the Coeur d'Alene River application.

4.6 Summary and Discussion

This paper demonstrates that a Genetic Algorithm (GA) is a viable approach for solving the LiDAR/bathymetric decimation problem. GAs cannot assure exact solutions, but yield reasonable solutions to optimization and search problems. For the hypothetical example, TINs from the GA runs are fairly representative of the original, but gave poor representation near the boundary. The VIP compared poorly to the original and to the GAs. The LATTICETIN compared favorably to the GA runs. The GA and LATTICETIN runs compared favorably to the original data with the LATTICETIN, overall, closer to the original. To better fit the original data, the fitness function used in the GA needs to integrate the entire volumetric curve and rather than just the total volume of the TINs. Also results from these GA runs demonstrated that near optimal results could be obtained in a single GA run.

For the case study of Coeur d'Alene River and Floodplain, TINs from the GA runs are fairly representative of the subsection even along the boundary. The VIP run also showed fair representation. Some of the finer features such as creeks, levees and roads were poorly defined in the GA and VIP runs. TIN volumes from the superior GA run (no. 4) and VIP compared favorably, with the GA run having a smaller difference from the original. A LATTICETIN run could not be performed because the combined bathymetric and LiDAR data do not line up to produce a regularly spaced dataset.

Although the genetic algorithm was successful in decimating the datasets, it still needs to be tested with datasets having more data points. The current fitness function calculates the volume of an individual at the zero height. A fitness function that integrates volume along its height (hypsometric curve) might cause the GA to select better fit individuals in the population that have smaller differences from the original at all elevations. Also the current fitness function always selects the plimit as the number of inclusions. A fitness function that gradually penalizes more points and gradually gives credit to fewer points might also enable the GA to select a good minimum number of points. These improvements are left for future investigations.

4.7 References

Barton, G.J., Moran, E.H., Berenbrock, C. (2004). Stream Channel Cross Sections for the Kootenai River Between Libby Dam, Montana, and Kootenay Lake, British Columbia, Canada. U.S. Geological Survey Open-File Report 2004-1045, p 35.

Berenbrock, C. (2006). "A genetic algorithm to reduce stream channel cross section data," Journal of the American Water Resources Association, 42(2), pp 387-394.

Berenbrock, C., and Tranmer, A.W. (2008). Simulation of flow, sediment transport, and sediment mobility of the Lower Coeur d'Alene River, Idaho. U.S. Geological Survey Scientific Investigations Report 2008-5093, p 164.

Chen, L. (2003). "Real Coded Genetic Algorithm Optimization of Long Term Reservoir Operation," Journal of the American Water Resources Association (JAWRA) 39(5), pp 1157-1165.

Chen, Z.T., and Guevara, J.A. (1987). “Systematic Selectin of Very Important Points (VIP) from Digital Terrain Model for Constructing Triangular Irregular Networks,” Proceedings of the Eighth International Symposium on Computer-Assisted Cartography, N.R. Chrisman (Editor), pp 57-67.

Cieniawski, W.E., Eheart, J.W., and Ranjithan, S. (1995). “Using Genetic Algorithms to Solve a Multiobjective Groundwater Monitoring Problem,” Water Resources Research 21(2), pp 399-409.

Davis, L., (1991). “Hybridization and Numerical Representation,” In: Handbook of Genetic Algorithms, L. Davis (Editor). Van Nostrand Reinhold, United Kingdom, pp 62-72.

FEMA (Federal Emergency Management Agency). (1995). Guidelines and Specifications for Study Contractors. Federal Emergency Management Agency, Publication 37, U.S. Government Printing Office, Washington, D.C., p 174.

Goldberg, D.E., (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Pub. Co., p 372.

Grefenstette, J.J. (1990). “Genetic Algorithms and Their Application,” In: Encyclopedia of Computer Science and Technology, A. Kent and J.G. Williams (Editors). Marcel Dekker, New York, New York, 21(6), pp 139-152.

Holland, J.H. (1975). Adaptation in Natural and Artifical Systems. University of Michigan Press, Ann Arbor, Michigan.

Knaapen, M.A.F., and Hulscher, S.J.M.H. (2003). “Use of a Genetic Algorithm to Improve Predictions of Alternate Bar Dynamics,” Journal of Water Resources Research 39(9), 1231, doi:10.1029/2002WR001793, 2003.

McKinney, D.C., and Lin, M.D. (1992). “Design Methodology for Efficient Aquifer Remediation Using Pump and Treat Systems,” In: Mathematical Modeling in Water Resources, T. Russel *et al*. (Editors). Elsevier Science, New York, New York, pp 695-702.

Mitchell, M. (2002). An Introduction to Genetic Algorithms. The MIT Press, Cambridge, Massachusetts, (8th printing), p 209.

Moran, E.H. and Berenbrock, C. (2003). “GPS—Time Saver and Functional,” U.S. Geological Survey Western Water Watch, 1(1), pp 6-7.

Reed, P., Minsker, B. and Goldberg, D.E. (2000). “Designing a Competent Simple Genetic Algorithm for Search and Optimization,” Journal of Water Resources Research, 36(12), pp 3757-3761.

Reed, P.B., Minsker, B., and Goldberg, D.E. (2003). “Simplifying Multiobjective Optimization–An Automated Design methodology for the Nondominated Sorted Genetic Algorithm-II,” Journal of Water Resources Research, 39(7), pp 1196-1206.

Vink, K. and Schot, P. (2002). “Multiple-Objective Optimization of Drinking Water Production Strategies Using a Genetic Algorithm,” Journal of Water Resources Research 38(9), pp 1157-1165.

CHAPTER 5. SPECTRAL ANALYSIS OF CROSS-SECTION DATA

5.1 Introduction

Spectral analysis is the process of decomposing a complex signal into simpler parts (Brigham, 2002). Usually, it is used in the analysis of electrical signals or data that has periodic components of time. Specifically it has been used in optics, speech, sonar, radar, medicine, seismology, chemistry, radio astronomy, oceanography, etc. For cross-section data, spectral analysis will be used to determine how energy is distributed over space (spectral content). Also it is hoped that the spectral content of the cross-section data will be unique for different channel types and for different scales of resolution. Cross-section data has no components of time, but the space and time domain will be exchanged. The exchange was set to 1 foot (ft) to 1 second (s) or 1 ft equaling 1 s.

Cross sections from the Kootenai River, Idaho (Barton *et al.*, 2004) will be evaluated for its spectral content. One cross section from each geomorphic or channel type—meander, straight, braided, and canyon (Czuba and Barton, 2011)—will be used. These cross sections are presented in Chapters 2 and 3 of this dissertation.

The MATLAB software was used to conduct spectral analysis on the cross-section data. MATLAB is a proprietary programming language of MathWorks (https://www.mathworks.com). The MATLAB program shown in Appendix O was used to determine the spectral content, specifically the power spectral density function (PSD), of cross-section data. A fast Fourier transform (FFT) algorithm (https://www.mathworks.com/help/matlab/ref/fft.html and Brigham, 2004) was used to determine the PSD.

5.2 Spectral Content of Cross Sections

Four cross section from the Kootenai River, Idaho, will be analyzed for its spectral content. Cross sections 107.658, 152.019, 154.972, and 163.027 that represented the meander, straight, braided, and canyon reach types, respectively, were used in this analysis. Spectral analysis was performed on the entire cross-section data, but the cross-section data had to be modified because spectral analysis requires regularly spaced data. Whereas, the data from these cross sections are irregularly space. To develop regularly spaced data from irregularly spaced data, linearizing between each data point in the cross section was done, and then the data were selected at the desired interval spacing. Data from the linearized cross section was reselected 7 different times to develop modified cross sections having an interval spacing of 0.5 ft, 1 ft, 5 ft, 10 ft, 50 ft, 100 ft, and 200 ft. These modified cross sections represent the different scales of resolution of a cross section.

Cross section 154.972 located in the braided reach was analyzed first because previous analyses considered it to be more complexly shaped than the other cross sections (see Section 3.4.1). The modified cross section 154.972 with an interval spacing of 0.5 ft is shown in Figure 5.1A. The shape of this cross section is very similar to that of the original, irregular spaced data, cross section (Figure 3.2). Note that the horizontal axis in Figure 5.1A is labeled as “Time, in seconds” because the space and time domain were exchanged (1 ft = 1 s) so that the spatial content (energy and frequency) can be determined.

The power spectral density function (PSD) calculates the strength of energy as a function of frequency. It generally indicates which frequencies are strong or weak. Figure 5.1B shows the PSD of the modified cross section 154.972 with an interval spacing of 0.5 ft. The horizontal axis is frequency in the time domain. Note that frequency is inversely proportional to the period. In climate time series data, for example, the PSD is determined to find out periodicity an event. Usually distinctive or sharp peak(s) would appear in a PSD plot to indicate where very strong energies occur(s), but for the modified cross-section 154.972, no distinctive peaks occurred at any frequency (Figure 5.1B). This indicates that there is no periodicity in the landform of this cross section. The PSD plot (Figure 5.1B) also showed energy decreasing exponentially toward higher frequencies. This is especially seen in the smoothed red line in Figure 5.1B. Smoothing the data removes the random variation and can help show data trends. Smoothing was done using a moving average.

Figure 5.2 shows smoothed curves of PSD for cross section 154.972 at 7 different interval spacing ranging from 0.5 ft to 200 ft. No distinctive peaks occur in any of the curves. The curves generally follow the trend of the 0.5 ft curve. A hill of higher energy at low frequencies (near 0.001) for the 100 ft and 200 ft curves is probably an artifact of the smaller amount of data available for analysis, 24 and 13 data points, respectively.

The spectral content, specifically determining PSD, was perform on cross sections 107.658, 152.019, and 163.027. Before PSD could be performed, the cross sections were also modified for each interval spacing. The PSD for these cross sections showed no distinct power increases at any of the frequency ranges and for the different interval spacing. The PSD for these cross sections were similar to that of the PSD for cross section 154.972 as shown in Figure 5.2. Thus, plots of PSD for these cross

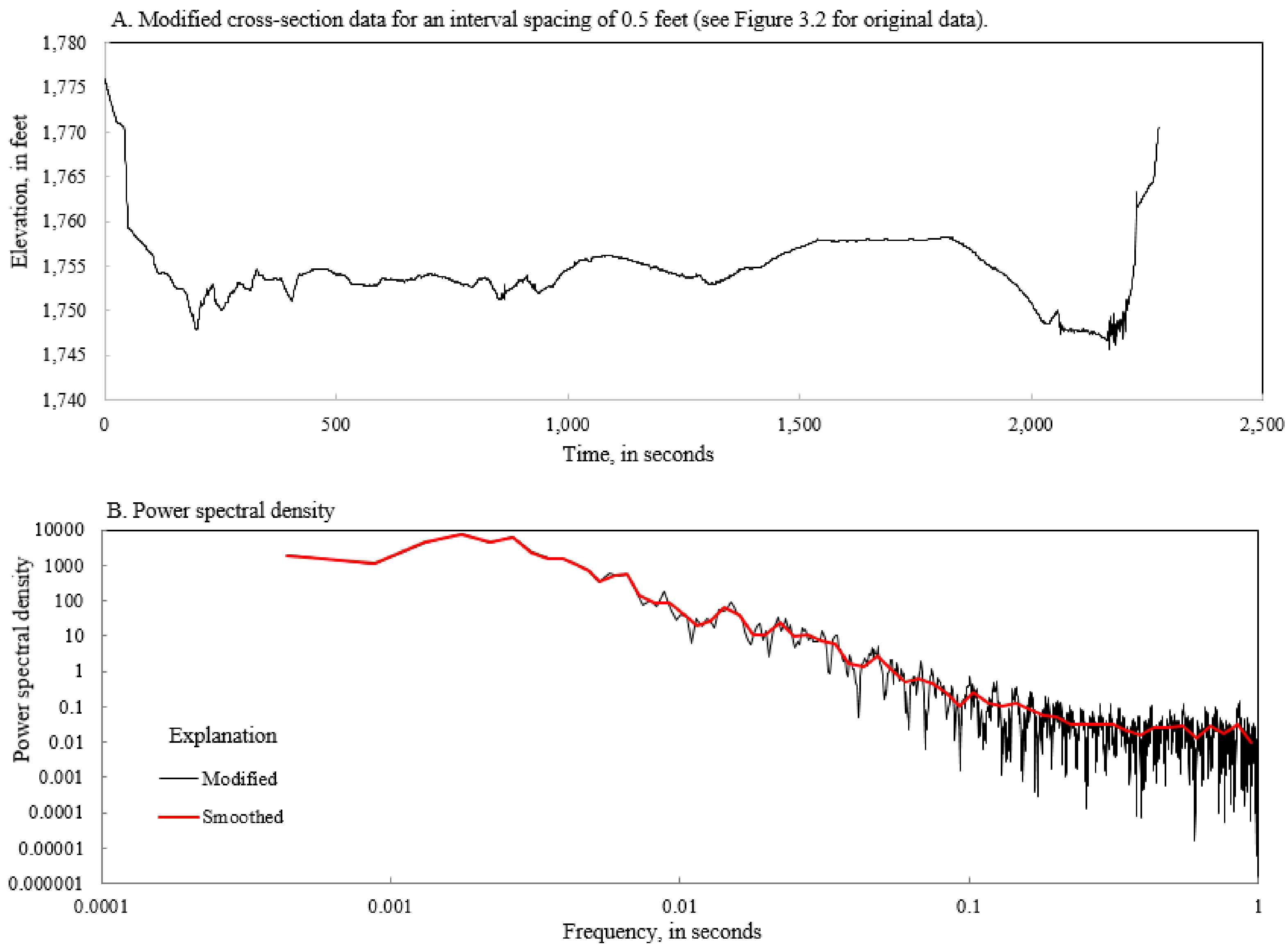


Figure 5.1. Modified cross-section data for an interval spacing of 0.5 feet and its power spectral density for cross section 154.972, braided reach, Kootenai, Idaho.

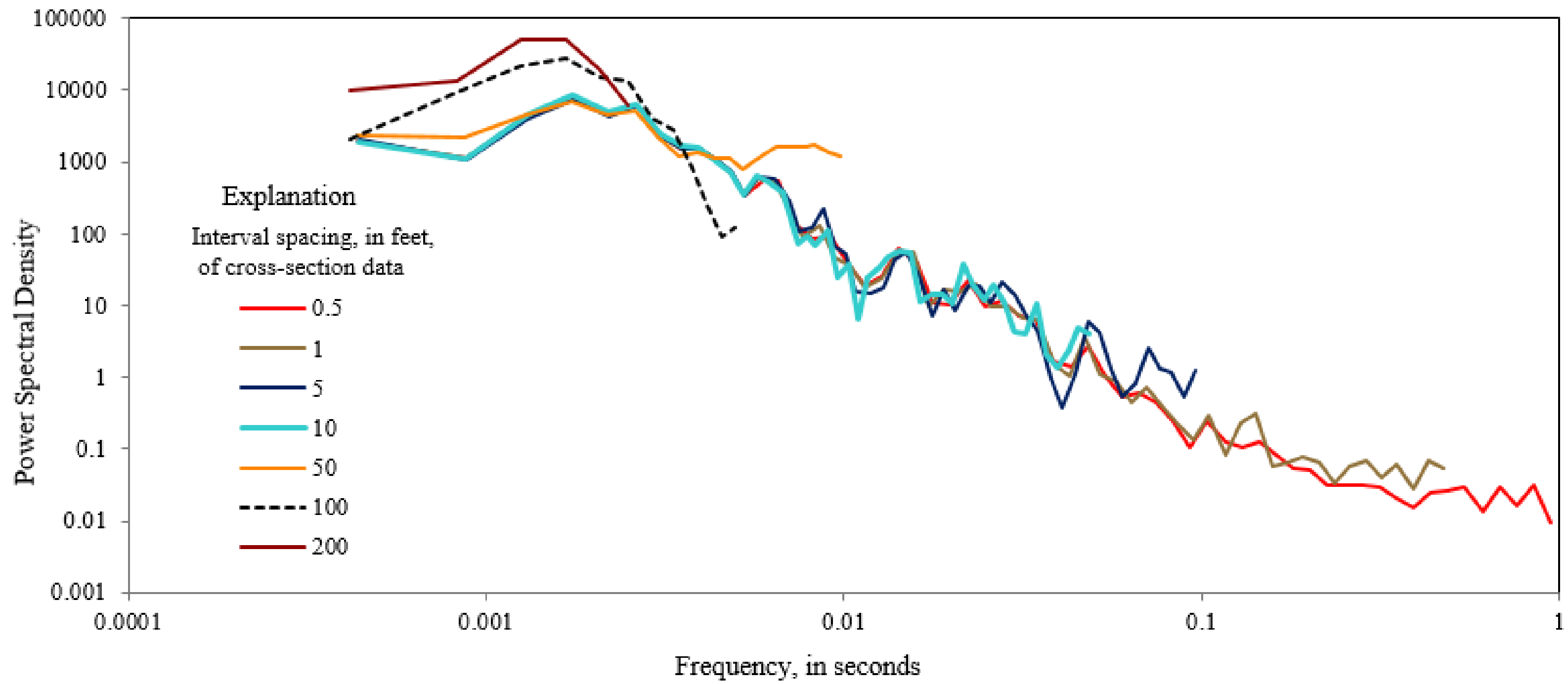


Figure 5.2. Smoothed curves of power spectral density for cross-section 154.972, braided reach, at selected interval spacing of 0.5, 1, 5, 10, 50, 100, and 200 feet of cross-section data.

sections were not shown. Also for several modified cross sections, the PSD could not be determine because the cross section was not long enough to have sufficient amounts of data for determining the PSD. This was especially true for an interval spacing of 100 ft and 200 ft. The PSD indicates that there is no periodicity in the landform of these cross sections suggesting that the different channel types cannot be discerned using spectral analysis.

5.3 Summary

Spectral analysis was perform on data from four cross section from the Kootenai River, Idaho. Each cross section was representative of a different geomorphic or channel type—meander, straight, braided, and canyon. Before spectral analysis could begin, the cross-section data were modified because the original data was irregularly space and spectral analysis requires regularly space data. Thus, the data were modified by linearizing between each data point in a cross section. Data from the linearized cross section was reselected 7 different times to develop modified cross sections having an interval spacing of 0.5 ft, 1 ft, 5 ft, 10 ft, 50 ft, 100 ft, and 200 ft.

The power spectral density function (PSD) for all cross sections at each interval spacing showed no distinct power increases at any frequency range. The spectral energy decreased exponentially as frequency increased. Because there were no distinct power increases, different channel types could not be discerned using spectral analysis. Unfortunately, cross-section data did not lend itself to spectral analysis like wave and (or) seabed landforms.

5.4 References

Barton, G.J., E.H. Moran, and C. Berenbrock. (2004). Stream Channel Cross Sections for the Kootenai River Between Libby Dam, Montana, and Kootenay Lake, British Columbia, Canada. U.S. Geological Survey Open-File Report 2004-1045, p 35.

Brigham, E. Oran, (2004). The Fast Fourier Transform. New York, New York. Prentice-Hall. p. 304.

Czuba, C.R., and G.J. Barton, (2011). Updated one-dimensional hydraulic model of the Kootenai River, Idaho—A supplement to Scientific Investigations Report 2005–5110. U.S. Geological Survey Scientific Investigations Report 2011–5128, 36 p.

MATLAB, Statistics Toolbox Release (2011b), and Signal Processing Toolbox Release (2011b). The MathWorks, Inc., Natick, Massachusetts, United States.

CHAPTER 6. EXECUTIVE SUMMARY

Two new tools have been developed. These tools successfully reduced or decimated the number of points in river geometry data to be handled and consequently used in surface-water models and in any other subsequent processes. The first tool is a genetic algorithm (GA) to reduce stream channel cross section data. A hypothetical cross section consisting of 41 data pairs (distance and elevation) was first tested. Even though the GA relies on randomness in sampling and in creating the initial population, the range in best fitness for all runs was small compared with the ranges in average fitness and root mean squared error (RMSE). Validation included using real-world cross sections from the Kootenai River in Idaho and comparing the best fitness values between GA-reduced and standard reduction method (Barton *et al.*, 2004) cross sections. The best fitness values for the GA-reduced cross sections were all lower than standard reduced cross sections. Several GA-reduced cross sections had fitness values about 50 percent lower (better) than those from standard method. The GA-reduced cross sections also closely represented the original cross section and that near-optimal results could be obtained in a single GA run, even for large problems. Computer run times for the Kootenai River cross sections were much faster for the GA-reduced cross sections than the standard reduction. Estimates indicate that it would take one week to complete the 245 Kootenai River cross sections using the GA, a time savings of 75 percent over the standard procedure.

Additional research reveal that the previous genetic algorithm (original) did not account for irregularly spaced data. Thus, it was modified. To validate the modifications, the 10 cross sections that were used to validate the original GA were used. Results showed that best fitness from the modified GA were consistently lower (demonstrating better performance) than for the standard-method and original GA. On average, the modified GA fitness was 39.2 and 57.0 percent lower than the standard reduction method and original GA, respectively. The modified GA-reduced cross sections approximated the shape of the original cross sections better than the other two reduction methods. Therefore, the modification that was made to the original GA is the more appropriate genetic algorithm-reduction method and was used throughout the study.

Further analyses were conducted to evaluate the effects of the reduced cross sections on cross-section geometry and steady flow profiles for the two methods. Visual analysis (graphs) demonstrated that GA-reduced cross sections approximated the shape of the original cross section better than the standard-reduced cross sections. This was also true for the cross-sectional area. Also an reduction error (RE) was developed to quantify the performance of the cross-section reduction. RE values were lower (better) for the GA-reduced cross sections than for the standard method. RE curves were also developed for the 10 Kootenai River cross sections only using the GA-reduction method. These curves showed that cross sections in canyon and meander reaches need fewer points than cross sections in the braided and straight reaches. It also confirms that a greater amount of data are needed to define more complexly shaped cross sections.

This study also investigated the practical consequences of errors due to cross-section reduction on steady-flow profiles. Thirty-five cross sections from the original steady-flow surface-water model of the Kootenai River were used. Cross-sectional data in these cross sections were reduced to 10, 20, and 30 data points for both reduction methods. Differences in water-surface elevation were less for cross sections developed by the GA-reduction method than by the standard-reduction method. Sometimes the methods did not select enough points in the secondary channels because fitness was not bettered (lower value) by doing so. To rectify this problem the GA needs to be modified so that thalweg points in secondary channels, not just the main, are selected

The second tool is also a GA but for decimating bathymetric/LiDAR datasets. Multi-dimensional datasets such as bathymetry and (or) LiDAR are usually very large and often surpasses the capacity of programs especially those of two- and three-dimensional surface-water models. A hypothetical example consisting of 961 regularly spaced points (LiDAR) was tested. The points were decimated to 15 percent or 144 points. GA results were mostly superior to standard reduction methods–VIP (Very Important Points) and LATTICETIN. Hypsometric curves of volume between the GA runs and original dataset were quite similar while the curves from VIP and LATTICETIN were quite different than the original. Validation of this GA included using bathymetric and LiDAR datasets from the Coeur d’Alene River and Floodplain in Idaho and comparing the reduction methods to the original. A LATTICETIN reduction was not performed because the LATTICETIN requires regularly spaced data and the Coeur d’Alene data are irregularly spaced. The major topographic features were preserved fairly well in the GA runs and VIP. The VIP showed several discontinuities (one in the river and one on a hill) while the best GA (fourth run) did not. Both discontinuity could have impacts on river and floodplain flows. Features such as the Fourth of July Creek and the road/levee north of the river were not distinguishable in either reduction. However, the road south of the river is somewhat distinguishable in the GA run. Volumetric differences from the original were smaller in the GA run than in the VIP, but the largest differences in both methods occurred in the first 5 meters of height. A fitness function that integrates volume along its height (hypsometric curve) might cause the GA to select better fit individuals in the population that have smaller differences from the original at all heights.

CHAPTER 7. FUTURE WORK

Implementation of the hypothetical examples were intended as a proof of concept, but to provide more rigorous validation of the algorithms, real world river geometry data–Kootenai and Coeur d'Alene Rivers in Idaho–were used. This methodology showed that the cross-section genetic algorithm (GA) and the LiDAR-bathymetry GA can greatly reduce the amount of information without a significant loss in precision. However, to extend the usefulness and to address limitations, future enhancements to these GAs could include:

- Automate the process in the program to determine the appropriate operator (crossover and mutation) values instead of the current trial-and-error approach.
- Ability to run multiple instances of the GA program at once.
- Modify the fitness function such that it gradually penalizes more points and gradually gives credit to fewer points might also enable the GA to select a good minimum number of points or plimit.
- Demonstrate how the cross-section GA can be applied to other water resource, ecological, and biological data and to other X-Y datasets and time series datasets.
- Test the cross-section GA using cross-section data of high density from other rivers from around the world and with various channel types (low gradient-meander (single thread and sinuous), braided, anastomosing, nearly straight, etc.; and high gradient-riffle-pool sequence, rapids, step-pools, cascade, etc.).
- Modify the cross-section GA to select more points in cross sections having multiple channels such as braided channels.
- Rewrite the GAs to have a one-dimensional array where all data is stored, similar to the code in MODFLOW (Langevin *et al.* 2017, https://water.usgs.gov/ogw/modflow/). Hopefully, this action allows the GA to run faster, more efficiently, with larger datasets and without the use of supercomputers or high-end PCs.
- Rewrite the GAs with no operators, crossover and mutation (called "Naive Evolution") and test them. Spears and Anand (1991) indicated that some practical problems are better solved using this methodology. Also biologist consider mutation, not crossover, as the main source of evolution (Senaratna, 2005).
- Develop general RE curves for the various channel types to estimate the minimum number of points needed for that type of cross sections.

- Run the LiDAR-Bathymetric GA with datasets having at least several million points or greater.
- Modify the fitness function of the LiDAR-Bathymetric GA to calculate the volume along its height (hypsometric curve) for each reduced dataset curve and compare to the original dataset hypsometric curve. The difference between the original and reduced hypsometric curves would be minimized by the fitness function similar to what is done in Equation 3.1. Hopefully this method causes the GA to select better fit individuals in the population that have smaller differences from the original at all heights.

7.1. References

Spears W.M., Anand V. (1991) A study of crossover operators in genetic programming. In: Ras Z.W., Zemankova M. (eds) Methodologies for Intelligent Systems. ISMIS 1991. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 542. Springer, Berlin, Heidelberg, p. 409-?

Nuwan I. Senaratna,2005, Genetic Algorithms: The Crossover-Mutation Debate, in partial fulfilment of the requirements for the Degree of Bachelor of Computer Science(Special) of the University of Colombo, 22 p.

Langevin, C.D., Hughes, J.D., Banta, E.R., Niswonger, R.G., Panday, Sorab, and Provost, A.M., 2017, Documentation for the MODFLOW 6 Groundwater Flow Model: U.S. Geological Survey Techniques and Methods, book 6, chap. A55, 197 p., https://doi.org/10.3133/tm6A55.

CHAPTER 8. CONCLUSIONS

The size of digital datasets can be quite large, and as technology advances, the size in digital data usually increases too. Large datasets cause numerous problems in storing, handling, transmitting, and with software. Computer models usually have a finite limit on the amount of data it can use, and data reduction or decimating is commonly applied to large datasets to skirt these limitations. Whenever data are reduced, some informational content is lost. Choosing a suitable set of data points for the representation of the natural geometry of a river and floodplain is quite important for the accuracy of surface-water models. The goal was to minimize that loss while maximizing the amount of data reduction because it could affect channel and floodplain geometry determinations and water-surface calculations, which in turn has major effects on the computations of velocity, shear stress, and sediment transport. By decimating intelligently, large datasets such as LiDAR can be reduced to a manageable size for surface-water models and other computer applications while maintaining the original geometry.

Two genetic algorithms (GAs) were developed for decimating river geometry data: one for cross-section data and the other for bathymetry and (or) LiDAR data. These GAs were shown to successfully reduce or decimate the data and were found to be more effective than standard reduction methods–removing or keeping every tenth point, for example, regardless of its significance, which is unacceptable.

The cross-section GA program successfully reduced cross-sectional data by having smaller differences in cross-sectional area and water-surface elevations between the GA-produced and original cross sections than those using standard methods. Fitness (a measure of a solution) values were consistently lower (demonstrating better performance) for GA-produced cross sections. Reduction error (RE) values were also lower (better) for GA-reduced cross sections. RE curves also demonstrated that about 20 data points are needed to approximate the shape of the original cross section in the canyon and meander reaches, and more than 40 points are need in the straight reach and more than 70 data points in the braided reach. However, this needs to be tested against datasets from other rivers.

The bathymetric and (or) LiDAR GA program was also successful in decimating the data by having smaller differences in the terrain between the GA-produced and original than by other methods. Also the GA datasets selected fewer data points on the boundary while the terrain by other methods is under emphasized in the interior and over emphasized at the boundary. Volumetric analysis [hypsometric curves of volume] also showed that the GA terrain more closely tracks the original.

Therefore, this dissertation demonstrates that the genetic algorithm is a viable approach for solving the cross-section reduction and bathymetric/LiDAR decimation problems.

Appendix A. Copyright from Journal of the American Water Resources Association (Chapter 2)

The Journal of the American Water Resources Association (JAWRA) is a John Wiley & Sons publication. For the material in Chapter 2, permission was granted and is shown as follows:

Publisher Tax ID	EU826007151
Billing Type	Invoice
Billing Address	
Total	0.00 USD

Terms and Conditions

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a"Wiley Company") or handled on behalf of a society with which a Wiley Company has exclusive publishing rights in relation to a particular work (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

- The materials you have requested permission to reproduce or reuse (the "Wiley Materials") are protected by copyright.

- You are hereby granted a personal, non-exclusive, non-sub licensable (on a standalone basis), non-transferable, worldwide, limited license to reproduce the Wiley Materials for the purpose specified in the licensing process. This license, and any CONTENT (PDF or image file) purchased as part of your order, is for a one-time use only and limited to any maximum distribution number specified in the license. The first instance of republication or reuse granted by this license must be completed within two years of the date of the grant of this license (although copies prepared before the end date may be distributed thereafter). The Wiley Materials shall not be used in any other manner or for any other purpose, beyond what is granted in the license. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Wiley Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Wiley Material. Any third party content is expressly excluded from this permission.

- With respect to the Wiley Materials, all rights are reserved. Except as expressly granted by the terms of the license, no part of the Wiley Materials may be copied, modified, adapted (except for minor reformatting required by the new Publication), translated, reproduced, transferred or distributed, in any form or by any means, and no derivative works may be made based on the Wiley Materials without the prior permission of the respective copyright owner.For STM Signatory Publishers clearing permission under the terms of the STM Permissions Guidelines only, the terms of the license are extended to include subsequent editions and for editions in other languages, provided such editions are for the work as a whole in situ and does not involve the separate exploitation of the permitted figures or extracts, You may not alter, remove or suppress in any manner any copyright, trademark or other notices displayed by the Wiley Materials. You may not license, rent, sell, loan, lease, pledge, offer as security, transfer or assign the Wiley Materials on a stand-alone basis, or any of the rights granted to you hereunder to any other person.

- The Wiley Materials and all of the intellectual property rights therein shall at all times remain the exclusive property of John Wiley & Sons Inc, the Wiley Companies, or their respective licensors, and your interest therein is only that of having possession of and the right to reproduce the Wiley Materials pursuant to Section 2 herein during the continuance of this Agreement. You agree that you own no right, title or interest in or to the Wiley Materials or any of the intellectual property rights therein. You shall have no rights hereunder other than the license as provided for above in Section 2. No right, license or interest to any trademark, trade name, service mark or other branding ("Marks") of WILEY or its licensors is granted hereunder, and you agree that you shall not assert any such right, license or interest with respect thereto

- NEITHER WILEY NOR ITS LICENSORS MAKES ANY WARRANTY OR REPRESENTATION OF ANY KIND TO YOU OR ANY THIRD PARTY, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO THE MATERIALS OR THE ACCURACY OF ANY INFORMATION CONTAINED IN THE MATERIALS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, USABILITY, INTEGRATION OR NON-INFRINGEMENT AND ALL SUCH WARRANTIES ARE HEREBY EXCLUDED BY WILEY AND ITS LICENSORS AND WAIVED BY YOU.

- WILEY shall have the right to terminate this Agreement immediately upon breach of this Agreement by you.

- You shall indemnify, defend and hold harmless WILEY, its Licensors and their respective directors, officers, agents and employees, from and against any actual or threatened claims, demands, causes of action or proceedings arising from any breach of this Agreement by you.

- IN NO EVENT SHALL WILEY OR ITS LICENSORS BE LIABLE TO YOU OR

ANY OTHER PARTY OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, PROVISIONING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

- Should any provision of this Agreement be held by a court of competent jurisdiction to be illegal, invalid, or unenforceable, that provision shall be deemed amended to achieve as nearly as possible the same economic effect as the original provision, and the legality, validity and enforceability of the remaining provisions of this Agreement shall not be affected or impaired thereby.

- The failure of either party to enforce any term or condition of this Agreement shall not constitute a waiver of either party's right to enforce each and every term and condition of this Agreement. No breach under this agreement shall be deemed waived or excused by either party unless such waiver or consent is in writing signed by the party granting such waiver or consent. The waiver by or consent of a party to a breach of any provision of this Agreement shall not operate or be construed as a waiver of or consent to any other or subsequent breach by such other party.

- This Agreement may not be assigned (including by operation of law or otherwise) by you without WILEY's prior written consent.

- Any fee required for this permission shall be non-refundable after thirty (30) days from receipt by the CCC.

- These terms and conditions together with CCC's Billing and Payment terms and conditions (which are incorporated herein) form the entire agreement between you and WILEY concerning this licensing transaction and (in the absence of fraud) supersedes all prior agreements and representations of the parties, oral or written. This Agreement may not be amended except in writing signed by both parties. This Agreement shall be binding upon and inure to the benefit of the parties' successors, legal representatives, and authorized assigns.

- In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall prevail.

- WILEY expressly reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

- This Agreement will be void if the Type of Use, Format, Circulation, or Requestor Type was misrepresented during the licensing process.

- This Agreement shall be governed by and construed in accordance with the laws of the State of New York, USA, without regards to such state's conflict of law rules. Any legal action, suit or proceeding arising out of or relating to these Terms and Conditions or the breach thereof shall be instituted in a court of competent jurisdiction in New York County in the State of New York in the United States of America and each party hereby consents and submits to the personal jurisdiction of such court, waives any objection to venue in such court and consents to service of process by registered or certified mail, return receipt requested, at the last known address of such party.

WILEY OPEN ACCESS TERMS AND CONDITIONS

Wiley Publishes Open Access Articles in fully Open Access Journals and in Subscription journals offering Online Open. Although most of the fully Open Access journals publish open access articles under the terms of the Creative Commons Attribution (CC BY) License only, the subscription journals and a few of the Open Access Journals offer a choice of Creative Commons Licenses. The license type is clearly identified on the article.

The Creative Commons Attribution License

The Creative Commons Attribution License (CC-BY) allows users to copy, distribute and transmit an article, adapt the article and make commercial use of the article. The CC-BY license permits commercial and non-

Creative Commons Attribution Non-Commercial License

The Creative Commons Attribution Non-Commercial (CC-BY-NC)License permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.(see below)

Creative Commons Attribution-Non-Commercial-NoDerivs License

The Creative Commons Attribution Non-Commercial-NoDerivs License (CC-BY-NC-ND) permits use, distribution and reproduction in any medium, provided the original work is properly cited, is not used for commercial purposes and no modifications or adaptations are made. (see below)

Use by commercial "for-profit" organizations

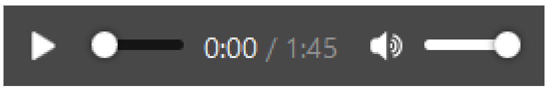
Use of Wiley Open Access articles for commercial, promotional, or marketing purposes requires further explicit permission from Wiley and will be subject to a fee.

Further details can be found on Wiley Online Library http://olabout.wiley.com/WileyCDA /Section/id-410895.html Other Terms and Conditions: v1.10 Last updated September 2015

Appendix B. Copyright from U.S. Geological Survey

From: https://www.usgs.gov/media/audio/are-usgs-reports-copyrighted

Are USGS reports copyrighted?



(Public domain.)

Detailed Description

Listen to hear the answer.

Details

Date Taken: April 17, 2008

Location Taken: US

Transcript

[music fades in]

Hello and welcome to CoreFacts, where we're always short on time and big on science. I'm Steve Sobieszczyk. Today we're looking publishing.

Are USGS reports copyrighted?

USGS-authored or produced data and information are in the public domain. While the content of most USGS web pages and reports are in the public domain, not all information, illustrations, or photographs are. Some are used by USGS with permission. In those cases, you may need to obtain permission from the copyright holder under the copyright law. You are welcome to make a link to any of the websites USGS has published online. There is no need to request permission. However, the use of the USGS visual identifier, or logo, is restricted to official publications. When using information from USGS products, publications, or websites, we ask that proper credit be given. Credit can be provided by including a citation such as:

Credit: U. S. Geological Survey/photo by Steven Sobieszczyk (or photographer's name) or if the originating office is known but the artist or scientist is not, it can be sited as: USGS/Portland, OR (for example).

Additional information is available from links in our show notes: USGS Privacy Policy and Disclaimers and Acknowledging or Crediting USGS as Information Source.

If you have questions concerning the use of USGS information, contact us at 1-888-ASK-USGS or 1-888-275-8747.

And now you know. Join us every weekday for a new CoreFact. If you have a question you think we should answer on the air, email it to us at corefacts@usgs.gov or leave us a voicemail at 703-648-5600; long distance fees do apply.

The USGS CoreFacts is a product of the U.S. Geological Survey, Department of the Interior.

[music fades out]

Appendix C. Copyright from Federal Interagency Hydrologic Modeling Conference Proceedings

The Hydrologic Modeling Workgroup (HMWG) is the organization that plans, organizes, and promotes the Federal Interagency Hydrologic Modeling Conference about every 4 years. The purpose of HMWG is to promote the sharing of information on modeling tools and modeling systems in hydrology and water resources between federal, state and local government agencies, universities, the private sector, water users' organizations and other stakeholder groups. HMWG is a working group of the Subcommittee on Hydrology (SOH) of the Advisory Committee on Water Information (ACWI). The Office of Management and Budget (OMB) Memorandum No. 92-01 designates the Department of the Interior, through the U.S. Geological Survey (USGS), as the lead agency of ACWI (https://acwi.gov/m9201.html). Other Federal organizations that fund, collect, or use water resources information work together with the USGS to implement program recommendations. Therefore, the content of the ACWI web pages, data, and information including conference proceedings are in the public domain. The home web page for ACWI is https://acwi.gov/.

Appendix D. General Description of the Genetic Algorithm Program for Reducing Cross-Section and (or) X-Y Data

The genetic algorithm program that was used to reduce cross-section and (or) X-Y data in Chapters 2 and 3 is generalized in this appendix. However, only the irregularly spaced data module (ftness03_area.f) discussed in chapter 3.3.2 (Genetic Algorithm Reduction) is kept and used. The program was written in the FORTRAN (77 and (or) 90) computer language and is composed of 14 FORTRAN files. The program is composed of one main program and 20 subroutines. A file may include more than one subroutine. A description of the main program and subroutines including the FORTRAN file, and calling subroutines is given in Table D.1. The program follows the flow chart in Figure D.1.

The program uses the GNU FORTRAN (gfortran, http://gcc.gnu.org/wiki/GFortran) to compile and load the program. The following command in the command processor (cmd.exe) window can be used.

```
gfortran dxsxy02.f average02.f best202.f ftnss03_area.f gip.f
rdm01.f read1.f read202.f rmse02.f select02.f tourn02.f xover.f
zlast02.f zmutat.f
```

To run program the program in a command processor window, type "`c:\dxsxy02`". Another way to run the program is from the 'Run' command line in the 'start' menu. Browse to the folder where 'dxsxy02.exe' is located and double click on it.

The genetic algorithm program reads a parameter file named 'param.dat'. This file is in text file. The first line in the file contains the file name of the cross-section or time-series data. The second line is a number representing the size of the population (n); the third line represents the number of points to reduce to (plimit), the fourth line represents the crossover rate (P_c); and the fifth line represents the mutation factor (m_f). The mutation rate is inversely proportional to the population size multiplied by the mutation rate or $P_m = \frac{1}{m_f \times n}$. The parameter file for the hypothetical cross-section example (Chapter 2.5.1, Hypothetical Example) is given in Figure D.2. The cross-section data for the hypothetical example is in a file named "hypo41.dat" (see Appendix F for data). The size of the population was set to 400 individuals, the program was run for 1000 generations, the number of points

Table D.1. Files that compose the genetic algorithm cross-section reduction program.

Main program or subroutine	File	Calling Subroutines	Description
dsxsy	dxsxy02.f	cpu_time, date_and_time, read1, read2, gip, select	Main program
average	average02.f	--	Calculates the average fitness of the population
best2	best202.f	--	Determine the best (superior) individual of the population
ftnss	ftnss03_area.f	--	Calculates the fitness of each individual of the population for irregular spaced data
gip	gip.f		Creates the initial population
rdm	rdm01.f	random_seed, system_clock	Initates the random number generator which is dependend on the date and time
read1	read1.f	--	Reads the ‘param.dat’ file on unit 36
read2	read202.f	--	Reads the cross-section or time-series data on unit 33
rmse	rmse02.f	--	Calculates the root-mean-squared error (RMSE) fitness of the population
select	select02.f	ftnss, average, rmse, zlast, best2, tourn, xover, zmutat	Uses a generational selection method
tourn	tourn02.f	random_number	Conducts tournament selection for the reproduction method with a size of 3
xover	xover.f	random_number	Conducts crossover
zlast	zlast02.f	--	Determine the worst or least superior individual of the population
zmutat	zmutat.f	random_number	Conducts mutation on each chromosome

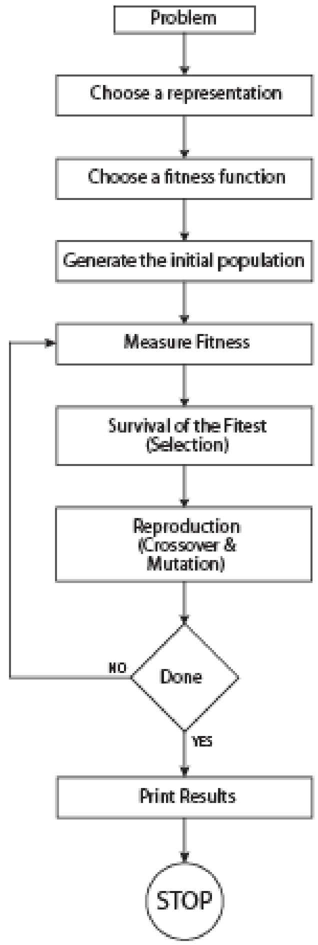


Figure D.1. Flow chart of evolutionary computation (modified from Terrance Soule, 2003, written communication).

```
hypo41.dat
400         /* population size (n)
1000        /* number of generations (ngen)
15          /* number of points to reduce to (plimit)
0.70        /* crossover rate (Pc)
1           /* mutation factor (mf)
```

Figure D.2. The parameter file ("param.dat") for the hypothetical cross-section example.

to reduce too or less than was set to 15 points, the crossover rate (P_c) was set to 70 percent, and the mutation factor (m_f) was set to 1.

The “params.dat” file (Figure D.2) is read in the program from unit 36 from subroutine “read1”. The cross-section data (see Appendix F) or X-Y data are read from unit 33 from subroutine “read2”. The program outputs three text files: r-table_dxsxy.csv, indivi_dxsxy.txt, and r-xsxy_dxsxy.txt (see Appendix G). File “r-table_dxsxy.csv” prints the best fitness, worst fitness, average fitness, rmse fitness per generation. This file is a comma delimited text file that can easily be imported into a spreadsheet; hence, the csv suffix. These results are printed using unit 31. The ‘rmse’ is a statistical abbreviation for root-mean-squared error. The file “indivi_dxsxy.txt” prints the chromosomes from the best fit individual for each generation. The file is a space delimited text file and printed using unit 32. The file “r-xsxy_dxsxy.txt” prints the best GA cross-section data or X-Y data u. If the X-Y data are time-series data, the date and time must be converted to a single numeric value such as Julian date. The file is also a space delimited text file and printed using unit 34. Only one cross-section pair (distance and elevation) or X-Y pair is printed per line.

Appendix E. Listing of Computer Code for Reducing Cross-Sectional Data and (or) X-Y Data Using a Genetic Algorithm

E.1. Main Program (file dxsxy02.f)

```
!
!  ----------------------------------------------------------------------------
!  DECIMATING CROSS SECTION DATA WITH INTEGRITY USING A BINARY GENETIC ALGORITHM
!  ----------------------------------------------------------------------------
!
!
!  by: Charles Berenbrock
!  creation date: December 2006
!  language: FORTRAN
!
!  modified date: 01-27-09, fitness calculates the area between points-curve,
!                             not on distances between points
!
!
!  ------------------------------------------------------------------------------
!  To compile, type:
!  gfortran dxsxy02.f average02.f best202.f ftnss02_area.f gip.f rdm01.f read1.f
!       read202.f rmse02.f select02.f tourn02.f xover.f zlast02.f zmutat.f forig.f
!  ------------------------------------------------------------------------------
!
!
      double precision dist(5000), elev(5000)
!
      dimension ipop(2000,5000)
!
      integer dt(8)
!
      character*40 infile
      character*8  date
      character*10 time
      character*5  zone
!
      call cpu_time ( start )
!
!........Opening files output file
      open (31, file='r-table_dxsxy.csv', status='replace')
      open (32, file='indivi_dxsxy.txt', status='replace')
      open (34, file='r-xsxy_dxsxy.txt', status='replace')
!
!........Writing head notes to screen
      print *,''
      print *,''
      print *,' --------------------------------------------------'
      print *,'   Decimating  Cross  Section (XS)  or  X-Y  Data'
      print *,'         with  Integrity  using  a  Binary '
      print *,'              Genetic  Algorithm'
      print *,' --------------------------------------------------'
      print *,''
      print *,''
      call date_and_time (date,time,zone,dt)
      print *,'DATE: ',date,'  TIME: ',time
      print *,''
!
```

```
!.........Writing head notes to output files
      write(31,*) ''
      write(31,*) '---------------------------------------------'
      write(31,*) 'OUTPUT from the XS or X-Y DECIMATING PROGRAM  '
      write(31,*) '---------------------------------------------'
      write(31,*) ''
      write(31,*) 'RUN DATE:,',date,',  RUN TIME:,',time
      write(31,*) ''
      write(32,*) 'RUN DATE:',date,'   RUN TIME:',time
      write(32,*) ''
!
!........reading needed parameters
      open (36,file='params.dat',status='old')
      call read1 (infile, itp, istop, npts, pc, fpm)
!
!........Reading bathymetry data
      open (33, file=infile, status='old')
      n = 0
      call read2 (n, dist, elev, pm, fpm)
!
      ngen=0
!........Generate initial population
      call gip (n, itp, ipop, npts)
!
!........writing more head notes for output files
      write(31,*) 'Generation,Best,Best,Average,Worst,Worst'
      write(31,*) 'No.,Weight,Fitness,Fitness,Weight,Fitness,RMSE'
      write(32,*) 'BEST'
      write(32,*) 'Generation number, weight, fitness, and individual'
      write(32,*) ''
!........Selection of "survival of the fittest"
      call select(n,itp,ipop,dist,elev,ngen,istop,npts,pc,pm)
!
!........Estimating CPU time
      call cpu_time ( finish )
      cputime = finish - start
      print *,''
      print *,'Seconds to run program (CPU):', cputime
!
!
!........Ending the program
      stop
      end
!
```

E.2. Subroutine read1 (file read1.f)

```
!
!..........Reading population and stopage from screen
      subroutine read1(infile,itp,istop,npts,pc,fpm)
!
!
      character*40 infile
!
!............read input file from infile
!
!..............read file name of bathymetry data
      print *,''
      read (36,*) infile
!
!..............read population size
      read (36,*) itp
      if (itp.lt.50.or.itp.gt.2000) then
        print *,''
        print *,'Population size should be between 50 and 2000.'
        print *,'Please edit the parameter file'
        stop
        endif
!
      print *,'population size=',itp
      write(31,*) 'N =,',itp
!
!............read maximum number of generations
      print *,''
      print *,''
      read (36,*) istop
      write(31,*) 'Gen =,',istop
      if (istop.gt.5000) then
        print *,''
        print *,'Too many generations.'
        print *,'Edit the parameter file so that <=5000'
        stop
        endif
!
      print *,'maximum number of generations=',istop
!
      print *,''
      read (36,*) npts
      print *,'plimit =',npts
      write(31,*) 'plimit =,',npts
!
      print *,''
      read (36,*) pc
      print *,'crossover rate =',pc
      write(31,*) 'Pc =,',pc
!
      print *,''
      read (36,*) fpm
!
!
      return
      end
!
```

E.3. Subroutine read2 (file read202.f)

```
!
!.........Read dist and elev from infile
      subroutine read2(len,dist,elev,pm,fpm)
!
      double precision dist(5000),elev(5000)
!
!
      print *,''
      print *,'Reading cross-section data'
!
!...........read distance and elevation of each point
      i=1
   10 read ( 33, *, end=888 ) dist(i),elev(i)
!      print *,i,':', dist(i), elev(i)
      i=i+1
      goto 10
!
  888 len=i-1
!
      pm=(1./(fpm*(float(len))))
      print *,''
      print *,'mutation rate =',pm
      write ( 31, * ) 'Pm =,',pm
      write ( 31, * ) ''
!
      print *,''
      print *,''
      print *,'length of individuals=',len
      print *,''
!
      return
      end
!
```

E.4. Subroutine gip (file gip.f)

```
!
!.........Generate initial population
      subroutine gip(n,itp,ipop,npts)
!
      dimension ipop(itp,n)
      real rval
!
      call init_random_seed()
!
!
      prob=(float(npts)/float(n))
!
!...........initialize population to zeros and turn bits to 1 based on the
probability
      do 10 i=1,itp
        ipop(i,1)=1
        ipop(i,n)=1
        do 20 j=2,n-1
          ipop(i,j)=0
          call random_number(rval)
          if(rval.le.prob) ipop(i,j)=1
   20     continue
   10   continue
!
!      print *,'look at each individual (population=',itp,')'
!      do 30 i=1,itp
!       print *,i,':',(ipop(i,j),j=1,n)
!   30   continue
!      print *,''
!
!
      return
      end
!
```

E.5. Subroutine rdm (file rdm01.f)

```
SUBROUTINE init_random_seed()
  INTEGER :: i, n, clock
  INTEGER, DIMENSION(:), ALLOCATABLE :: seed

  CALL RANDOM_SEED(size = n)
  ALLOCATE(seed(n))

  CALL SYSTEM_CLOCK(COUNT=clock)

  seed = clock + 37 * (/ (i - 1, i = 1, n) /)
  CALL RANDOM_SEED(PUT = seed)

  DEALLOCATE(seed)
END SUBROUTINE
```

E.6. Subroutine select (file select02.f)

```
!
!.........Selection solution ***Generational GA***
      subroutine select(n,itp,ipop,dist,elev,ngen,istop,npts,pc,pm)
!
!
      double precision dist(n),elev(n)
      double precision f(itp)
      double precision fov,ave,r,ymin
!
      integer ipop(itp,n)
      integer temp(itp,n)
      integer j1(n),j2(n)
!
      dimension iw(itp)
!
!
!...........calculate the area of the original dataset (calc only once)
!      call forig(n,fov,ymin,dist,elev)
!
      print *,''
      print *,'Generations'
      print *,'-----------'
      print *,''
!
    5 ngen=ngen+1
      print *,ngen
!     print *,''
!...........measure the fitness of each individual
!      call ftnss(n,itp,ipop,dist,elev,f,fov,ymin,iw,npts)
      call ftnss(n,itp,ipop,dist,elev,f,iw,npts)
      call average(itp,f,ave)
      call rmse(itp,f,ave,r)
      call zlast(itp,f,m3)
!
!...........elitism (best fitnesses in population)
!     print *,'individuals 1 & 2'
!     print *,'-----------------'
      call best2(f,itp,m1,m2)
!...........print best fitness, average, worst, and rmse to unit 31
      write(31,*) ngen,',',iw(m1),',',f(m1),',',ave,',',iw(m3),',',
     1            f(m3),',',r
!...........print best individual characteristics to unit 32
      write(32,*) ngen,':',iw(m1),':',f(m1),':',(ipop(m1,j),j=1,n)
!
!...........print parents & children
!      print *,m1,':(parents)',(ipop(m1,j),j=1,n)
!      print *,m2,':(parents)',(ipop(m2,j),j=1,n)
!...........put 2 best individuals from old into the NEW population
      do 8 j=1,n
        temp(1,j)=ipop(m1,j)
        temp(2,j)=ipop(m2,j)
    8   continue
!        print *,'1:(children)',(temp(1,j),j=1,n)
!        print *,'2:(children)',(temp(2,j),j=1,n)
!        print *,''
!
!...........determine children by selecting two parents
      do 10 j=3,itp,2
!        print *,'individuals',j,'&',j+1
!        print *,'-----------------------'
        call tourn(f,itp,m1,m2)
```

```
!...........two parents (j1,j2) to create children
        do 20 ii=1,n
          j1(ii)=ipop(m1,ii)
          j2(ii)=ipop(m2,ii)
   20     continue
!...........crossover of parents to create children
        call xover(j1,j2,n,m1,m2,pc)
!...........mutation (probability based) of the two children
!        print *,'children after mutation'
        call zmutat(n,j1,m1,pm)
        call zmutat(n,j2,m2,pm)
!        print *,''
!...........put children into new population
        do 30 i=1,n
          temp(j,i)=j1(i)
          temp(j+1,i)=j2(i)
   30     continue
   10   continue
!
!..........replace NEW population with old population
      do 40 i=1,itp
        do 50 j=1,n
          ipop(i,j)=temp(i,j)
   50     continue
   40   continue
!
!      print *,''
!      print *,'NEW population'
!      print *,'--------------'
!      do 70 i=1,itp
!        print *,i,':',(ipop(i,j),j=1,n)
!   70   continue
!
!      print *,''
!
      if(ngen.lt.istop) goto 5
!
!........Output best reduced cross section
      do 80 j=1,n
   80   if(ipop(1,j).eq.1) write(34,1000) dist(j),elev(j)
 1000 format(1x,2f8.1)
!
!
      return
      end
!
```

E.7. Subroutine ftnss (file ftnss03.f)

```
!
!.........Calculate fitness for each individual based on the
!..........area between the curves: original and GA curve
      subroutine ftnss(n,itp,ipop,dist,elev,fit,iw,npts)

!      use inf_nan_detection


      double precision dist(n),elev(n)
      double precision fit(itp),d2(n)
      double precision cx(5),cy(5)
      double precision slope,b,sum,area1,area2
      double precision area(n)

      dimension ipop(itp,n)
      dimension iw(itp)


!      print *,'fitness for each individual (',itp,')'
!
!..........total the weight for each individual
!      print *,'weight for each individual'
      do 10 i=1,itp
        iw(i)=0
        k=0
        do 20 j=1,n
   20     if(ipop(i,j).eq.1) k=k+1
        iw(i)=k
!        print *,i,':',iw(i),':',(ipop(i,j),j=1,n)
   10 continue
!      print *,''


!..........determine vertical difference (d2) at each point in an individual
      do 30 i=1,itp
!............clearing d2 array
        do 40 j=1,n
   40     d2(j)=0.0d00
!............assigning a value of zero at 1's
        do 50 j=1,n
   50     if(ipop(i,j).eq.1) d2(j)=0.0d00
!
!............determine value of d2 at every point
        j1=1
        j2=1
   52   j2=j2+1
        if(ipop(i,j2).eq.0) goto 52
        slope=(elev(j2)-elev(j1))/(dist(j2)-dist(j1))
        b=elev(j1)-slope*dist(j1)
        do 53 k=j1+1,j2-1
          vy=slope*dist(k)+b
          d2(k)=vy-elev(k)
   53     continue
        j1=j2
        if(j1.lt.n) goto 52


!...........print d2 array
!        print *, 'dist   d2'
!        do 55 j=1,n
!          print *, dist(j), d2(j)
```

```
!   55   continue
!        print *, ''
!        print *, ''


!...........determine area of depth curve (d2) for each point along cross section
        do 60 j=1,n-1

!.............case (1)
          if(ipop(i,j).eq.1.and.ipop(i,j+1).eq.1) then
!            print *, 'Case (1)'
            area(j)=0.d00
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif

!.............case (2)
          if(ipop(i,j).eq.1) then
!            print *, 'Case (2)'
            cx(1)=dist(j)
            cy(1)=0.0d00
            cx(2)=dist(j+1)
            cy(2)=d2(j+1)
            cx(3)=dist(j+1)
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 62 k=1,3
   62         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area(j)=dabs(sum)/2.0d00
!            print *, 'j=', j,'  j+1=',j+1, '  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif

!.............case (3)
          if(ipop(i,j+1).eq.1) then
!            print *, 'Case (3)'
            cx(1)=dist(j)
            cy(1)=0.0d00
            cx(2)=dist(j)
            cy(2)=d2(j)
            cx(3)=dist(j+1)
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 63 k=1,3
   63         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area(j)=dabs(sum)/2.0d00
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif
```

```
!..............case (4)
          if(d2(j).gt.0.0d00.and.d2(j+1).gt.0.0d00.or.d2(j).lt.0.0d00
     1       .and.d2(j+1).lt.0.0d00.or.d2(j).eq.0.0d00.or.d2(j+1).eq.
     2       0.0d00) then
!            print *, 'Case (4)'
            cx(1)=dist(j)
            cy(1)=0.0d00
            cx(2)=dist(j)
            cy(2)=d2(j)
            cx(3)=dist(j+1)
            cy(3)=d2(j+1)
            cx(4)=dist(j+1)
            cy(4)=0.0d00
            cx(5)=cx(1)
            cy(5)=cy(1)
            sum=0.0d00
            do 64 k=1,4
   64         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area(j)=dabs(sum)/2.0d00
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif

!...............case (5)
          if(d2(j).lt.0.0d00.and.d2(j+1).gt.0.0d00) then
!            print *, 'Case (5)'
            cx(1)=dist(j)
            cy(1)=0.0d00
            cx(2)=dist(j)
            cy(2)=d2(j)
            slope=(d2(j+1)-d2(j))/(dist(j+1)-dist(j))
            b=d2(j)-slope*dist(j)
            cx(3)=-b/slope
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 65 k=1,3
   65         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area1=dabs(sum)/2.0d00
!            print *, 'area1=', area1
            cx(1)=cx(3)
            cy(1)=0.0d00
            cx(2)=dist(j+1)
            cy(2)=d2(j+1)
            cx(3)=dist(j+1)
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 865 k=1,3
  865         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area2=dabs(sum)/2.0d00
!            print *, 'area2=', area2
            area(j)=area1+area2
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
```

```
          endif

!.............case (6)
          if(d2(j).gt.0.0d00.and.d2(j+1).lt.0.0d00) then
!            print *, 'Case (6)'
            cx(1)=dist(j)
            cy(1)=0.0d00
            cx(2)=dist(j)
            cy(2)=d2(j)
            slope=(d2(j+1)-d2(j))/(dist(j+1)-dist(j))
            b=d2(j)-slope*dist(j)
            cx(3)=-b/slope
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 66 k=1,3
   66         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area1=dabs(sum)/2.0d00
!            print *, 'area1=', area1
            cx(1)=cx(3)
            cy(1)=0.0d00
            cx(2)=dist(j+1)
            cy(2)=d2(j+1)
            cx(3)=dist(j+1)
            cy(3)=0.0d00
            cx(4)=cx(1)
            cy(4)=cy(1)
            sum=0.0d00
            do 866 k=1,3
  866         sum=sum+cx(k)*cy(k+1)-cx(k+1)*cy(k)
            area2=dabs(sum)/2.0d00
!            print *, 'area2=', area2
            area(j)=area1+area2
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif

!.............case (7)
          if(d2(j).eq.0.0d00.and.d2(j+1).eq.0.0d00) then
!            print *, 'Case (7)'
            area(j)=0.0d00
!            print *, 'j=', j,'  j+1=',j+1,'  area=', area(j)
!            print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!            print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!            print *, ''
            goto 69
          endif

!.............case (8)
!          print *, 'Case (8)'
!          print *, 'if you reach this, sometime went wrong'
!          print *, 'j=', j,'  j+1=',j+1
!          print *,'ipop(i,j)=',ipop(i,j),' ipop(i,j+1)=',ipop(i,j+1)
!          print *, 'd2(j)=', d2(j), '   d2(j+1)=', d2(j+1)
!          print *, ''

   69     continue
   60   continue
!.............summing up areas for total area
```

```
        sum=0.0d00
        do 70 j=1,n-1
          sum=sum+area(j)
   70   continue
        fit(i)=sum
!        print *,'individual=', i, '    total area (fitness)=', fit(i)
   30 continue


      nan=0
      do 80 i=1,itp
        if(iw(i).gt.npts) fit(i)=(fit(i)+1.)*10.**(float(iw(i)-npts))


!...........checking for "not a number" value (NaN)
         if(isnan(fit(i))) then
            nan=nan+1
            fit(i)=10.**30.
            endif
   80   continue


!     print *, 'number of NAN:', nan


      return
      end
!
```

E.8. Subroutine average (file average02.f)

```
!
!.........calculates average fitness

      subroutine average (itp, f, ave)


      double precision ave,sum,outfb
      double precision f(itp)


      m = 0
      outfb = 10.d00**50
      sum = 0.d00
      ave = 0.d00


      do 10 i = 1, itp
        if (f(i). lt. outfb) then
          sum = sum + f(i)
          m = m + 1
          endif
   10   continue


      ave = sum / dfloat(m)


      return
      end
!
```

E.9. Subroutine best2 (file best202.f)

```
!
!............elitism--find the 2 best fitnesses in the entire population
!..............equal or less than the weight limit
      subroutine best2(f,itp,m1,m2)


      double precision f(itp)


!      print *,'Elitism: find 2 best indivials in population (',itp,')'


!......find best value from fitness
      m1=1
      do 20 i=2,itp
        if(f(i).lt.f(m1)) m1=i
!       print *,'** f(',m1,')=',f(m1),'  f(',i,')=',f(i)
   20   continue


!      print *,'best=',m1,'  (fitness=',f(m1),')'


!......now find second best value
      m2=1
      if(m1.eq.1) m2=2
      do 30 i=2,itp
        if(i.eq.m1) goto 30
        if(f(i).lt.f(m2)) m2=i
!       print *,'** f(',m2,')=',f(m2),'  f(',i,')=',f(i)
   30   continue


!      print *,'second best=',m2,'  (fitness=',f(m2),')'


      return
      end
!
```

E.10. Subroutine rmse (file rmse02.f)

```
!
!...........calculates the root-mean-squared error of fitness
      subroutine rmse (itp, f, ave, r)


      double precision f(itp)
      double precision ave,r,outfb


      m = 0
      outfb = 10.d00**50
      sum = 0.d00
      r = 0.d00


      do 10 i = 1, itp
        if(f(i) .lt. outfb) then
          sum = sum + (f(i)-ave)**2
          m = m + 1
          endif
   10   continue


      r = dsqrt(sum/dfloat(m))


      return
      end
!
```

E.11. Subroutine zlast (file zlast02.f)

```
!
!...........find the worst fitness
      subroutine zlast (itp, f, m3)


      double precision outfb
      double precision f(itp)


      m3 = 1
      outfb = 10.d00**50


      do 10 i = 2, itp
        if (f(i) .ge. outfb) goto 10
        if (f(i) .gt. f(m3)) m3 = i
   10   continue


      return
      end
!
```

E.12. Subroutine tourn (file tourn02.f)

```
!
!............tournament selection
      subroutine tourn(f,itp,m1,m2)


      double precision f(itp)

      integer ir(7)


!.........select 3 individuals in the population by random choice
      ii=3
!      print *,'select best of',ii,'individuals (#, fitness)'
      do 10 i=1,ii
    5   call random_number(rval)
        ir(i)=int(rval*itp+0.5)
        if(ir(i).eq.0) goto 5
!        print *,ir(i),':',f(ir(i))
   10   continue


!.........sorting by fitness
      do 30 j=1,ii
        do 20 i=1,ii-1
          if(f(ir(i+1)).lt.f(ir(i))) then
            ia=ir(i)
            ir(i)=ir(i+1)
            ir(i+1)=ia
            endif
   20     continue
   30   continue


!...........the 2 best
      m1=ir(1)
      m2=ir(2)


!     print *,'best=',m1
!     print *,'second best=',m2

      return
      end
!
```

E.13. Subroutine xover (file xover.f)

```
!
!.........performs random two-point crossover between 2 individuals
!...........and only middle portion gets switched
      subroutine xover(j1,j2,n,m1,m2,pc)
!
      integer j1(n),j2(n)
!
!      print *,'children before crossover'
!      print *,m1,':',(j1(i),i=1,n)
!      print *,m2,':',(j2(i),i=1,n)
!
!...........determining if crossover will occur with probability of 70%
!.............NO CROSSOVER if > 70%
      call random_number(rval)
      if(rval.gt.(pc/100.)) then
!        print *,'NO CROSSOVER--probability exceeds Pc',pc,'%'
!        print *,'children are identical copies of parents'
        return
        endif
!
!...........determining random two points
    5 call random_number(rval)
      i1=int(rval*n+0.5)
      if(i1.eq.0) goto 5
!      print *,'random number i1=',i1
   20 call random_number(rval)
      i2=int(rval*n+0.5)
      if(i2.eq.0) goto 20
!      print *,'random number i2=',i2
!
      if(i1-i2)10,20,30
   30 k=i1
      i1=i2
      i2=k
!
   10 continue
!      print *,'2-point random no: i1=',i1,'  i2=',i2
!...........performing crossover between i1 and i2
      do 40 i=i1,i2
        k=j1(i)
        j1(i)=j2(i)
        j2(i)=k
   40   continue
!
!      print *,'children after crossover and before mutation'
!      print *,m1,':',(j1(i),i=1,n)
!      print *,m2,':',(j2(i),i=1,n)
!
      return
      end
!
```

E.14. Subroutine zmutat (file zmutat.f)

```
!
!.........Causes possible mutation of children based on a small probability
!...........Mutation
      subroutine zmutat(n,k,m,pm)
!
      integer k(n)
!
!
!...........determining mutation at each bit in the string with a
!............probability of Pm=1/N
      prob=1.-pm
!
      j=0
      do 10 i=2,n-1
        call random_number(rval)
        if(rval.lt.prob) goto 10
        j=j+1
        if(k(i).eq.0) then
          k(i)=1
          else
          k(i)=0
          endif
   10   continue
!
      if(j.eq.0) then
!        print *,'NO MUTATION of',m
        return
        endif
!
!      print *,m,':',(k(i),i=1,n)
!
      return
      end
!
```

Appendix F. Listing of Input File for the Hypothetical Cross Section Example

The hypothetical cross section (Figure F.1) is described in Chapter 2 and in Berenbrock (2006) and is used here to demonstrate the use of the cross-section genetic algorithm program. The input file must be a space delimited text file composed of x, y pairs representing cross-section data or X-Y data. If the X-Y data are time-series data, the date and time must be converted to a single numeric value such as Julian date. The first value represents x and the second value represents y. The value of x must increase and cannot be negative. Also the value of y cannot be negative.

For the hypothetical cross section (Figure F.1), the x value represents distance from left bank and the y value represents elevation above a datum (Table F.1).

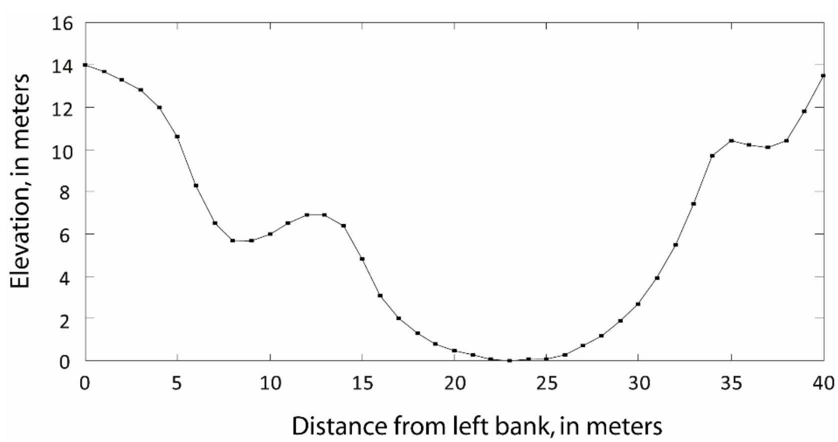


Figure F.1. Graph of the hypothetical cross section example.

Table F.1. Listing of x-y data pairs for the hypothetical cross section example.

0	14.0
1	13.7
2	13.3
3	12.8
4	12.0
5	10.6
6	8.3
7	6.5
8	5.7
9	5.7
10	6.0
11	6.5
12	6.9
13	6.9
14	6.4
15	4.8
16	3.1
17	2.0
18	1.3
19	0.8
20	0.5
21	0.3
22	0.1
23	0.0
24	0.1
25	0.1
26	0.3
27	0.7
28	1.2
29	1.9
30	2.7
31	3.9
32	5.5

33	7.4
34	9.7
35	10.4
36	10.2
37	10.1
38	10.4
39	11.8
40	13.5

Appendix G. Listing of Outputs Files for the Hypothetical Cross Section

The cross-section genetic algorithm program outputs three files: r-table_dxsxy.csv, r-xsxy_dxsxy.txt, and indivi_dxsxy.txt. These files will be replace when the program is reran, so it is suggested that they be renamed if these files are needed. The file r-table_dsxsy.csv is a comma delimited text file containing fitness values (best, average, worst, and RMSE) for every generation. This file is a comma delimited text file that can easily be imported into a spreadsheet; hence, the csv suffix. An example of output file r-table_dxsxy.csv for the hypothetical cross section for the first 100 generations is given in Table G.1. The file r-xsxy_dxsxy.txt is also a text file that contains the final genetic algorithm (GA)-produced cross section or X-Y data. A listing of output file r-xsxy_dxsxy.txt is given in Table G.2. The first column in this file represents distance or the x-value while the second column represents the elevation or the y-value. File indivi_dxsxy.txt is a text file that contains the weight (the number of included pairs), fitness (calculated from Equation 3.1), and chromosomes for the best individual in each generation. The chromosomes in this program are either 0 or 1 because the GA is binary. A 0 bit in the chromosome represents exclusion of that particular data pair on the cross section or X-Y data, and a 1 represents inclusion. The output file indivi_dxsxy.txt is a text file and is given in Table G.3.

Table G.1. Listing of output file r-table_dxsxy.csv.

```
----------------------------------------------
 OUTPUT from the XS or X-Y DECIMATING PROGRAM
----------------------------------------------

RUNDATE:, 20100330, RUNTIME:, 143536.237

N=, 400
Gen=, 100
plimit=, 15
Pc=, 0.69999999
Pm=, 2.43902430E-02

Generation, Best, Best, Average, Worst, Worst
No., Weight, Fitness, Fitness, Weight, Fitness, RMSE
1, 14, 10.322726 181667903.9853, 25, 54357141461.909225, 2774413746.2586784
2, 14, 10.322726 163162130.5744, 25, 58500013276.934601, 2934715274.9809661
```

3, 15, 8.5227265 9538.291652732, 20, 1180789.4858113229, 80828.670715285174
4, 15, 8.5227265 6766.403249407, 20, 1647200.0937302245, 83482.753993863909
5, 15, 8.3184426 1350.510677429, 19, 153500.00587105754, 11624.948178809229
6, 15, 8.3184426 321.3273822682, 18, 19911.475292159575, 1773.0504335748603
7, 15, 8.2092486 413.6691861093, 19, 101857.13591867725, 5131.4225084278532
8, 15, 8.2092486 548.8316125608, 19, 123516.13496195931, 6331.5042636011867
9, 15, 7.5186865 504.5494623409, 19, 103220.00336785450, 5301.8032253187221
10, 15, 7.518686, 594.174162623, 19, 97499.997779726968, 5139.7887583051506
11, 15, 7.166433, 352.4791983070, 18, 15050.000546872610, 1560.5724078042647
12, 15, 7.166433, 457.0920974123, 19, 61500.002473592751, 3431.8069875795754
13, 15, 7.166433, 203.2871105444, 18, 19149.801882014428, 1202.5518866144612
14, 15, 7.166433, 2526.726114204, 20, 890000.08553266560, 44453.067644876886
15, 15, 7.166433, 449.8304849961, 19, 87596.144947869077, 4530.8128365669663
16, 15, 7.166433, 772.8006138946, 19, 148114.75489599962, 7602.9003307948215
17, 15, 6.927778, 1081.579588373, 19, 192000.00485777855, 10611.930976028822
18, 15, 6.927778, 638.8608642598, 19, 84672.430873546939, 4631.2008982552243
19, 15, 6.927778, 3994.026427575, 20, 1070000.0655651093, 54337.792573493454
20, 15, 6.927778, 486.1888147425, 19, 75277.774769215146, 4027.0685715542513
21, 15, 6.927778, 531.7515538686, 19, 96999.990731477723, 5108.8831812833614
22, 15, 6.927778, 1002.981610954, 19, 94909.079995283420, 7834.9668742120411
23, 15, 6.927778, 783.3729713036, 19, 157701.60708096615, 9129.9720919617266
24, 15, 6.477778, 678.6163724741, 19, 127176.47784242159, 6611.0296202633972
25, 15, 6.477777, 771.4565754956, 19, 112717.40144440545, 7979.6923048448425
26, 15, 6.477777, 222.7476480084, 18, 10468.182224571003, 1053.7662359366047
27, 15, 6.477777, 624.5108940430, 19, 91000.006705522537, 5628.1418603301036
28, 15, 6.4777773, 522.15668804781114, 19, 122277.77940402887, 6193.5351068674827
29, 15, 6.4777773, 901.05693122242462, 19, 107000.00263750559, 8004.6955819693731
30, 15, 6.4777773, 552.43941859597442, 19, 115310.36209071535, 5877.9022108231775
31, 15, 6.4777773, 462.74692085069751, 19, 119637.93320647340, 6026.0611613225437
32, 15, 6.4777773, 567.68183156370355, 19, 96172.426619265461, 5037.5226927528574
33, 15, 6.4777773, 504.88265368296123, 19, 79999.993979930849, 4289.2908504786665
34, 15, 6.4777773, 997.37023698515350, 19, 117540.24198798572, 8363.6375626876616
35, 15, 6.4777773, 480.82297199560611, 19, 102999.99618530272, 5260.7075417666019
36, 15, 6.1277774, 2339.5261683249391, 20, 709545.42781942163, 35816.819932540078
37, 15, 6.1277774, 623.30356877758891, 19, 157500.02282857898, 8031.3631620043179
38, 15, 6.1277774, 547.75934870453591, 19, 77499.991416931152, 5224.8403611976510
39, 15, 6.1277774, 2576.3786153332326, 20, 815000.12695789314, 40914.100282420972
40, 15, 6.1277774, 470.05591481456338, 19, 72863.628856899348, 3871.9563633904759
41, 15, 6.1277774, 359.70422796547655, 18, 12317.271783249098, 1507.9532088231385
42, 15, 6.1277774, 606.71706268964783, 19, 67578.943574342164, 4724.8029800193790
43, 15, 6.1277774, 981.91631361441512, 19, 88272.721411152655, 7487.6406751392660
44, 15, 6.1277774, 637.54970416581216, 19, 140000.00119209287, 7125.5744470182890

```
45, 15, 6.1277774, 358.16081938372065, 18, 13889.736926398957, 1649.0873112118716
46, 15, 6.1277774, 2191.4837517003643, 20, 806379.36663542839, 40272.079205325368
47, 15, 6.1277774, 5693.3777470662972, 20, 1250000.0842660672, 74821.977109402826
48, 15, 6.1277774, 1207.2171528647916, 19, 112000.00344961882, 9710.5775564587293
49, 15, 6.1277774, 2890.2234064213744, 20, 951571.40015901404, 47779.106449576888
50, 15, 6.1277774, 579.90108520719798, 19, 91064.514873241293, 4819.6860312680119
51, 15, 6.1277774, 1129.4965080264221, 19, 196578.95368477996, 10943.262665220094
52, 15, 6.1277774, 4799.3902729803249, 20, 1611379.4381610018, 80791.085385455735
53, 15, 6.1277774, 572.57855017564020, 19, 76592.466570881705, 4140.0121352479146
54, 15, 6.1277774, 1292.4960029605529, 19, 168418.04534107150, 11127.016146299060
55, 15, 6.1277774, 2050.8992903553390, 20, 644250.96352770494, 32483.143099152214
56, 15, 6.1277774, 2890.1963395900079, 20, 940355.78781527991, 47298.251637877693
57, 15, 6.1277774, 2377.2530851434353, 20, 638181.86590743659, 32415.414328371618
58, 15, 6.1277774, 526.44850241782956, 19, 77999.993912875638, 4207.1319256709794
59, 15, 6.1277774, 557.80982193248474, 19, 66999.988287687287, 3686.1420048609089
60, 15, 6.1277774, 3274.1397169099182, 20, 917500.12710691418, 46508.498361482278
61, 15, 6.1277774, 436.48830885731809, 19, 67851.675949611978, 3644.5028851682914
62, 15, 6.1277774, 228.50228538720413, 18, 15500.000169873239, 1290.9642907532339
63, 15, 6.1277774, 506.51922621310473, 19, 95846.161592465156, 4933.5202847459741
64, 15, 6.1277774, 29409.072219379119, 21, 11600001.894682650, 579262.12857503467
65, 15, 6.1277774, 448.50035008881883, 19, 105954.54920434263, 5352.7609623445733
66, 15, 6.1277774, 239.35459493005300, 18, 14750.000530481340, 1242.2164062674426
67, 15, 6.1277774, 265.67905422478839, 18, 10353.350066884965, 1186.8719897276201
68, 15, 6.1277774, 705.79577433377574, 19, 84500.002950429916, 5940.3132577331307
69, 15, 6.1277774, 708.84337239835861, 19, 97500.011026859283, 6393.0272015689097
70, 15, 6.1277774, 454.88144353059886, 19, 89129.868043764771, 4644.1360875839973
71, 15, 6.1277774, 265.82154857649186, 18, 15630.645464163792, 1376.1689431170869
72, 15, 6.1277774, 1216.3772975245772, 19, 218500.00059604636, 12482.019939096395
73, 15, 6.1277774, 1909.1907708108376, 20, 636643.41015750298, 31808.883368015293
74, 15, 6.1277774, 418.13574795750594, 19, 79749.997437001279, 4094.5453667043425
75, 15, 6.1277774, 412.22651690772540, 19, 86910.669474083799, 4483.3473209199392
76, 15, 6.1277774, 343.41787440701609, 18, 14549.999544024466, 1596.8935844319747
77, 15, 6.1277774, 281.35462196415540, 18, 13649.999877810478, 1371.7278447272258
78, 15, 6.1277774, 692.34266744127979, 19, 88857.141461909152, 6075.3280997819365
79, 15, 6.1277774, 175.21038776193802, 18, 10135.986527390496, 829.19401830934601
80, 15, 6.1277774, 875.63929229221219, 19, 96500.006526708545, 7841.5848194099135
81, 15, 6.1277774, 678.90069694279146, 19, 79068.191865755682, 5779.5020962017134
82, 15, 6.1277774, 23899.195139191019, 21, 9301614.4185778089, 464501.20893500373
83, 15, 6.1277774, 4476.6057849239160, 20, 1350000.1299381263, 68297.108146099417
84, 15, 6.1277774, 385.03467414603983, 19, 68134.918541356892, 3514.9702246249540
85, 15, 6.1277774, 400.25391679212873, 19, 78910.675377921725, 4055.0277335673059
86, 15, 6.1277774, 3756.9352278095816, 20, 1439999.9582767480, 71906.403591335315
```

```
87, 15, 6.1277774, 401.72922058628797, 19, 60346.166688662292, 3213.8432569121974
88, 15, 6.1277774, 1394.0702105980306, 19, 117500.01835823053, 10392.655622120845
89, 15, 6.1277774, 129.74138543304184, 18, 8527.7773801192252, 646.03646955880129
90, 15, 6.1277774, 514.37436707057577, 19, 74277.771476611102, 4025.2497462890424
91, 15, 6.1277774, 426.86923808296524, 19, 76484.091028944851, 3994.0698441564591
92, 15, 6.1277774, 1094.9772408407564, 19, 114000.00816583633, 8756.7545403534059
93, 15, 6.1277774, 374.49284781282074, 18, 18549.999898672097, 1844.5111655937462
94, 15, 6.1277774, 179.06695505707853, 18, 11858.209266480560, 888.38316057881241
95, 15, 6.1277774, 544.80260929715087, 19, 102172.43061277660, 5260.8420922890282
96, 15, 6.1277774, 471.51634278943311, 19, 73499.995186924920, 3896.1245154640528
97, 15, 6.1277774, 429.87258789686581, 19, 90318.178961349098, 4611.2148507741431
98, 15, 6.1277774, 693.77210981942198, 19, 97500.008970498995, 6751.0111124186424
99, 15, 6.1277774, 240.56878828633796, 18, 12977.778428415913, 1150.2581623270491
100, 15, 6.127777, 652.82244611355611, 19, 149818.19332606849, 7577.8523606626168
```

Table G.2. Listing of output file r-xsxy_dxsxy.txt which contains the final genetic algorithm-produced cross section or X-Y data. Note that the first column represents distance or x-value, and second column represents elevation or y-value.

```
  0.0    14.0
  3.0    12.8
  5.0    10.6
  7.0     6.5
 10.0     6.0
 12.0     6.9
 14.0     6.4
 17.0     2.0
 20.0     0.5
 23.0     0.0
 27.0     0.7
 30.0     2.7
 35.0    10.4
 38.0    10.4
 40.0    13.5
```

Table G.3. Listing of output file indivi_dxsxy.txt.

[weight is the number of included pairs, fitness is calculated using Equation 3.1.]

```
RUN DATE:20100330 RUN TIME: 143536.237

BEST
Generation number, weight, fitness, and individual

 1: 14: 10.322726473828311: 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1
 2: 14: 10.322726473828311: 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1
 3: 15: 8.5227265423736505: 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1
 4: 15: 8.5227265423736505: 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1
 5: 15: 8.3184426559105553: 1 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1
 6: 15: 8.3184426559105553: 1 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1
 7: 15: 8.2092486538437690: 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1
 8: 15: 8.2092486538437690: 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1
 9: 15: 7.5186865061753521: 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 1
10: 15: 7.5186865061753521: 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 1
11: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
12: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
13: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
14: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
15: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
16: 15: 7.1664339927965548: 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1
17: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
18: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
19: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
20: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
```

```
21: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
22: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
23: 15: 6.9277780715331048: 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
24: 15: 6.4777781669005368: 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
25: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
26: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
27: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
28: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
29: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
30: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
31: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
32: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
33: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
34: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
35: 15: 6.4777773085936516: 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
36: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
37: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
38: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
39: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
40: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
41: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
42: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
43: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
44: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
45: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
46: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
47: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
48: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
49: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
```

```
50: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
51: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
52: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
53: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
54: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
55: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
56: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
57: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
58: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
59: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
60: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
61: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
62: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
63: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
64: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
65: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
66: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
67: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
68: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
69: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
70: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
71: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
72: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
73: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
74: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
75: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
76: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
77: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
78: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
```

```
 79: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 80: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 81: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 82: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 83: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 84: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 85: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 86: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 87: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 88: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 89: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 90: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 91: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 92: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 93: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 94: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 95: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 96: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 97: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 98: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
 99: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
100: 15: 6.1277774099215465: 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1
```

Appendix H. General Description of the Genetic Algorithm for Decimating Bathymetry and (or) LiDAR Data

The genetic algorithm (GA) program for decimating bathymetry and (or) LiDAR data was written in the FORTRAN computer language (version 77 and (or) 90) and is composed of 17 FORTRAN files. The GA program is composed of one main program and 18 subroutines. A file may include more than one subroutine. A generalize description of the main program and subroutines including the FORTRAN file and calling subroutines is given in Table H.1. Several subroutines also came from other sources, and permission by the authors/owners to use them are given in Appendices L, M, and N. The program follows the flow chart in Figure H.1. The computer code for decimating bathymetry and (or) LiDAR data is given in Appendix I.

The program uses the GNU FORTRAN (gfortran, http://gcc.gnu.org/wiki/GFortran) to compile and load the program. The following command in the command processor (cmd.exe) window is used.

```
      gfortran d3dga08.f select03.f gip02.f ftnss04.f average02.f
median01.f rmse02.f last02.f best202.f tourn02.f xover03.f
mutate04.f calcvol02.f isort01.f zxc201.f locpt02.f test0704.f
```

To run the LiDAR and bathymetric GA program in a command processor window, type "c:\d2sga08". Another way to run the program is from the 'Run' command line in the 'start' menu. Browse to the folder where file 'd2dga08.exe' is located and double click on it.

The LiDAR and bathymetric genetic algorithm program reads all data into the program from unit 21. Six data files are needed: params.dat, seq_out.txt, seq_info.txt, bndpts.txt, vol_out.txt, and hull_out2.txt. The file "params.dat" defines the GA parameters for simulation and is a text file (Figure H.2). The first line is a number representing the size of the population (n); the second line represents the number of generations to simulate, the third line represents the number of points to reduce to (plimit), the fourth line represents the crossover rate (P_c); and the fifth line represents the mutation factor (m_f). The mutation rate (P_m) is inversely proportional to the product of the population size and mutation rate. Another words, $P_m = \frac{1}{m_f \times n}$. The parameter file for the hypothetical LiDAR example is shown in Figure H.2. LiDAR data for the hypothetical example (see Appendix J) is a

Table H.1. Files that compose the decimating bathymetry and (or) LiDAR program.

Main program or subroutine	File	Calling Subroutines	Description
d3dga	d3dga08.f	cpu_time, date_and_time, gip, select	Main program
average	average02.f		Calculates the average fitness of the population
median	median01.f	isort	Calculates the median fitness of the population
best2	best202.f		Determine the best (superior) individual in the population
ftnss	ftnss04.f	test0704, zxc2, calcvol	Calculates the fitness of each individual in the population
gip	gip02.f	Init_random_seed, random_number	Creates the initial population
init_random_seed	gip02.f	random_seed, system_clock	Initiates the random number generator which is dependent on date and time
test0704	test0704.f		GEOMPACK—for developing Delaunay triangulation(modified from http://orion.math.iastate.edu/burkardt/f_src/geompack/geompack_prb.f90)
median	median01.f		Calculates the median fitness value for the population
zxc2	file zxc201.f	locpt	Deletes TINs outside the boundary and modifies the hull output file
calcvol	calcvol02.f		Determines the volume of each individual by calculating the volume in each TIN
isort	isort01.f		Sorts the data (modified from http://www.netlib.no/netlib/slatec/src/isort.f)
rmse	rmse02.f		Calculates the root-mean-squared error (RMSE) fitness of the population

locpt	locpt02.f		Determines if a point is inside or outside a polygon (modified from http://jblevins.org/mirror/amiller/locpt.f90)
select	select03.f	ftnss, average, median, rmse, last, best2, tourn, xover, mutat	Uses a generational selection method
tourn	tourn02.f	random_number	Conducts tournament selection for the reproduction method with a size of 3
xover	xover03.f	random_number	Conducts crossover
last	last02.f	--	Determine the worst or least superior individual of the population
mutat	Mutat04.f	Random_number	Conducts mutation

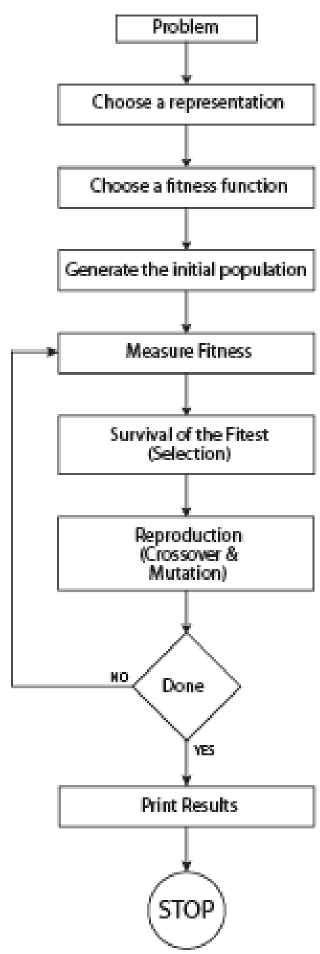


Figure H.1. Flow chart of evolutionary computation (modified from Terrance Soule, 2003, written communication).

square grid with dimensions of 31×31 columns and rows. The data points are spaced 16 m apart. The size of the population in the GA was set to 200 individuals, and was run for 1,000 generations. The number of points to reduce was set to 15 percent, the crossover rate (P_c) was set to 70 percent, and the mutation factor (m_f) was set to 1.

```
200     /* population size (n)
1000    /* number of generations to stop (ngen)
15      /* number of points to reduce to in percent (plimit)
0.70    /* crossover rate (Pc)
1       /* mutation factor (mf)
```

Figure H.2. The parameter file (contained in file "param.dat") for the hypothetical LiDAR example.

The file "seq_out.txt" contains the LiDAR data points (x, y, and z) in centimeters. The file must contain one data point on each line with its x, y, and z values given in that order and separated by at least by one space. A listing of the LiDAR data for the hypothetical LiDAR example is given in Appendix J.

The file "seq_info.txt" defines additional information that is needed by the program (Figure H.3). This file is in text format and values are separated by at least one space. The multiplication and subtraction values in lines 2 through 4 are used to convert the LiDAR data from meters to centimeters. For example, the original LiDAR data for the hypothetical example are in units of meters. To convert from meters to centimeters, one multiplies by 100. Hence, 100 is placed for the first value in lines two through four. The second number in these lines represents the minimum value that will be used to subtract other respective values. The number on the fifth line represents the number of data points that are within the boundary points. In the hypothetical example, all points are within the boundary; its value will be the same as on the first line. The number on the sixth line represents the number of boundary data points, and the first and last point must be identical.

```
961               /* total number of LiDAR data points
100    314400     /* x-value multiplication and subtraction values
100   1228000     /* y-value multiplication and subtraction values
100      1791     /* z-value multiplication and subtraction values
961               /* number of LiDAR data points within the boundary
5                 /* number of boundary data points
```

Figure H.3. The "seq_info.txt" file for the hypothetical LiDAR example.

The file "bndpts.txt" contains the boundary data points (x, y, and z) in meters. Figure H.4 shows the boundary data points for the hypothetical example. The first and last boundary data points must be identical.

```
3144.0 12280.0
3144.0 12760.0
3624.0 12760.0
3624.0 12280.0
3144.0 12280.0
```

Figure H.4. Boundary points (contained in file "bndpts.txt") for the hypothetical LiDAR example.

The file "vol_out.txt" contains the volume of the LiDAR data above a zero vertical datum in cubic meters (m^3). The volume for the hypothetical example is shown in Figure H.5. This value probability can be estimated by mapping, computer-aided design (CAD), and (or) geographic information system (GIS) software.

```
793626.02666666685
```

Figure H.5. Volume (contained in file "vol_out.txt") for the hypothetical LiDAR example.

The last input file is the hull file (“hull_out2.txt”). This file contains a listing of points representing the convex hull of the LiDAR data. The listing corresponds to the line number of the coordinate (x, y, and z) in the LiDAR data (“seq_out.txt” file). The hull file for the hypothetical example is shown in Figure H.6. The first number in Figure H.6, for example, corresponds to the coordinate in line number 1 of the seq_out.txt file, which has the coordinate of x = 0, y = 0, and z = 532. The second number corresponds to the coordinate in line number 31, x = 0, y = 48000, and z = 598. Coordinates for the remaining hull points can be found in a similar method.

```
  1
 31
961
931
```

Figure H.6. Hull points (contained in file “hull_out2.txt”) for the hypothetical LiDAR example.

The LiDAR and bathymetric genetic algorithm program outputs two files: r-table_dxsxy.csv and r-xsxy_dxsxy.txt. The file “r-table_dxsxy.csv” prints the best fitness, worst fitness, average fitness, median fitness, and rmse fitness per generation. The file is a comma delimited text file that can easily be read into a spreadsheet; hence, the csv suffix. The file “r-xsxy_dxsxy.txt” prints the best GA cross-section data or X-Y data. The file is a space delimited text file with one data point (x-y-z) per line. A listing of the output files for this hypothetical example are given in Appendix K.

Appendix I. Listing of Computer Code for Decimating Bathymetry and (or) LiDAR Data Using a Genetic Algorithm

I.1. Main program (file d3dga08.f)

```
!
!  ----------------------------------------------------
!  Intelligent Decimation of LiDAR and Bathymetry Data
!      for Use in 2D and 3D surface-water models
!          Using a Binary Genetic Algorithm
!  ----------------------------------------------------


!  by: Charles Berenbrock
!  creation date: November 2009
!  language: FORTRAN77&90



!  ---------------------------------------------------------------------------
!
!  To compile:
!
!  gfortran d3dga08.f select03.f gip02.f ftnss04.f average02.f median01.f rmse02.f
last02.f best202.f tourn02.f xover03.f mutate04.f calcvol02.f isort01.f zxc201.f
locpt02.f test0704.f90
!
!  ---------------------------------------------------------------------------



!.........declare variables
      dimension ix(11000), iy(11000), iz(11000)
      dimension ihv(1000)
      dimension ipop(300,11000)
      integer dt(8)
      double precision avol
      double precision bx(1000), by(1000)
      character*8  date
      character*10 time
      character*5  zone


      common /lidar/ ix, iy, iz
      common /conv/ multx, minx, multy, miny, multz, minz
      common /populace/ ipop
      common /bound/ bx, by
      common /hulls/ nh, ihv



      call cpu_time ( start )


!.........writing head notes to screen
      print *, ''
```

```
      print *, ''
      print *, ' ------------------------------------------------'
      print *, '  Intelligent Decimation of LiDAR and Bathymetry'
      print *, '   Data for Use in 2D & 3D Surface-Water Models'
      print *, '        Using a Binary Genetic Algorithm'
      print *, ' ---------------------------------------------'
      print *, ''
      print *, ''
      call date_and_time (date,time,zone,dt)
      print *, 'DATE: ', date,'  TIME: ', time
      print *, ''
      print *, ''


!.........reading parameters
      open (21, file='params.dat',    status='old')
      print *, 'reading the parameter file'
      read (21, *) nsize
      read (21, *) nstop
      read (21, *) prob
      read (21, *) cor
      read (21, *) zmrf
      close (21)
      print *, 'population size=', nsize
      print *, 'number of generations to stop=', nstop
      print *, 'data reduction, in percent=', prob
      print *, 'crossover rate=', cor
      print *, 'mutation rate factor=', zmrf
      print *, ''


!.........reading the number of points (vertices) and conversion units from info
file
      open (21, file='seq_info.txt',  status='old')
      print *, 'reading conversion file'
      read (21, *) i
      read (21, *) multx, minx
      read (21, *) multy, miny
      read (21, *) multz, minz
      read (21, *) nv
      close (21)
      print *, 'number of LiDAR points or vertices=', nv


!.........calculating the probability of mutation
      pm = (1./(zmrf*(float(nv))))


!.........calculating the number of bits to turn on
      npr = int(float(nv) * prob / 100.0 + 0.5)
      print *, 'number of points to reduce to=', npr
!      len = int( 1.2 * float(npr) + 0.5 )
      len = int( 1.5 * float(npr) + 0.5 )
      print *, 'length of individuals=', len


!.........reading hull points
      open (21, file='hull_out2.txt', status='old')
      print *, 'reading hull data'
      i = 0
      do
        i = i + 1
        read (21, *, end = 888) ihv(i)
```

```
      end do
  888 close (21)
      nh = i - 1
      print *, 'number of points defining the hull=', nh


!.........reading boundary data
      open (21, file='bndpts.txt', status='old')
      print *, 'reading boundary data'
      i = 0
      do
        i = i + 1
        read (21, *, end = 889) bx(i), by(i)
        bx(i) = dfloat(int(bx(i)*dfloat(multx)+0.5d00) - minx)
        by(i) = dfloat(int(by(i)*dfloat(multy)+0.5d00) - miny)
      end do
  889 close (21)
      nb = i - 1
      print *, 'number of points defining the boundary=', nb


!.........generate initial population
      print *, ''
      print *, 'generating the inital population'
      call gip ( nv, prob, npr, nsize, len )


!.........reading LiDAR & Bathymetry data
      open (21, file='seq_out.txt',   status='old')
      print *, 'reading LiDAR & Bathymetry data'
      do i = 1, nv
        read (21, *) ix(i), iy(i), iz(i)
      end do
      close (21)


!.........reading original volume
      open (21, file='vol_out.txt',   status='old')
      read (21, *) avol
      close (21)
      print *, 'reading original volume= (', avol, ')'


!.........writing output to UNIT 22
      open (22, file='stats_out.csv', status='REPLACE')
      write(22,1000)
 1000 format('Generation',2(',Best'),',Average',2(',Median'),
     12(',Worst'),',RMSE')
      write(22,2000)
 2000 format('No.,Weight,Fitness',2(',Fitness,Weight'),
     12(',Fitness'))


!.........Selection of "survival of the fittest"
      call select (nv, nb, nsize, nstop, len, npr, cor, pm, avol)


      close (22)


!.........Estimating CPU time
      call cpu_time ( finish )
      cput = (finish - start)
```

```
      print *, ''
      print *, ''
      print *, 'Seconds to run program (CPU):', cput
      print *, ''
      print *, ''

!.........Ending the program

      stop
      end
```

I.2. Subroutine gip (file gip02f)

```
      subroutine gip (nv, prob, npr, nsize, len )


!...........this subroutine generates the initial population


!...........declare variables
      dimension ihv(11000)
      dimension ipop(300,11000)

      common /populace/ ipop
      common /hulls/ nh, ihv

      call init_random_seed()


!...........probability of occurance
      prob = (float(npr) - float(nh)) / float(nv)
      print *, 'probability of occurance=', prob


!...........setting hull indices to 1
      print *, 'loading individuals with hull vertices'
      do i = 1, nsize
        do j = 1, nh
          ipop(i,ihv(j)) = 1
        end do
      end do


!...........generating the population
      do i = 1, nsize

        do j = 1, nv
          call random_number (rn)
          if (rn .le. prob) ipop(i,j) = 1
        end do

      end do


      return
      end
```

I.3. Subroutine init_random_seed() (included in file gip02.f)

```
SUBROUTINE init_random_seed()

INTEGER :: i, n, clock

INTEGER, DIMENSION(:), ALLOCATABLE :: seed

CALL RANDOM_SEED(size = n)

ALLOCATE(seed(n))

CALL SYSTEM_CLOCK(COUNT=clock)

seed = clock + 37 * (/ (i - 1, i = 1, n) /)

CALL RANDOM_SEED(PUT = seed)

DEALLOCATE(seed)

END SUBROUTINE
```

I.4. Subroutine isort (file isort01.f) (from http://www.netlib.no/netlib/slatec/src/isort.f; see Appendix L for copyright information)

```
*DECK ISORT
      SUBROUTINE ISORT (IX, IY, N, KFLAG)
C***BEGIN PROLOGUE  ISORT
C***PURPOSE  Sort an array and optionally make the same interchanges in
C            an auxiliary array.  The array may be sorted in increasing
C            or decreasing order.  A slightly modified QUICKSORT
C            algorithm is used.
C***LIBRARY   SLATEC
C***CATEGORY  N6A2A
C***TYPE      INTEGER (SSORT-S, DSORT-D, ISORT-I)
C***KEYWORDS  SINGLETON QUICKSORT, SORT, SORTING
C***AUTHOR  Jones, R. E., (SNLA)
C           Kahaner, D. K., (NBS)
C           Wisniewski, J. A., (SNLA)
C###############################################################################
C                http://www.netlib.org/slatec/src/isort.f
C###############################################################################
C***DESCRIPTION
C
C   ISORT sorts array IX and optionally makes the same interchanges in
C   array IY.  The array IX may be sorted in increasing order or
C   decreasing order.  A slightly modified quicksort algorithm is used.
C
C   Description of Parameters
C      IX - integer array of values to be sorted
C      IY - integer array to be (optionally) carried along
C      N  - number of values in integer array IX to be sorted
C      KFLAG - control parameter
C            =  2  means sort IX in increasing order and carry IY along.
C            =  1  means sort IX in increasing order (ignoring IY)
C            = -1  means sort IX in decreasing order (ignoring IY)
C            = -2  means sort IX in decreasing order and carry IY along.
C
C***REFERENCES  R. C. Singleton, Algorithm 347, An efficient algorithm
C                 for sorting with minimal storage, Communications of
C                 the ACM, 12, 3 (1969), pp. 185-187.
C***ROUTINES CALLED  XERMSG
C***REVISION HISTORY  (YYMMDD)
C   761118  DATE WRITTEN
C   810801  Modified by David K. Kahaner.
C   890531  Changed all specific intrinsics to generic.  (WRB)
C   890831  Modified array declarations.  (WRB)
C   891009  Removed unreferenced statement labels.  (WRB)
C   891009  REVISION DATE from Version 3.2
C   891214  Prologue converted to Version 4.0 format.  (BAB)
C   900315  CALLs to XERROR changed to CALLs to XERMSG.  (THJ)
C   901012  Declared all variables; changed X,Y to IX,IY. (M. McClain)
C   920501  Reformatted the REFERENCES section.  (DWL, WRB)
C   920519  Clarified error messages.  (DWL)
C   920801  Declarations section rebuilt and code restructured to use
C           IF-THEN-ELSE-ENDIF.  (RWC, WRB)
C***END PROLOGUE  ISORT
C     .. Scalar Arguments ..
      INTEGER KFLAG, N
C     .. Array Arguments ..
C      INTEGER IX(*), IY(*)
      integer ix(11000), iy(11000)
C     .. Local Scalars ..
```

```
      REAL R
      INTEGER I, IJ, J, K, KK, L, M, NN, T, TT, TTY, TY
C     .. Local Arrays ..
      INTEGER IL(21), IU(21)
C     .. External Subroutines ..
c      EXTERNAL XERMSG
C     .. Intrinsic Functions ..
      INTRINSIC ABS, INT
C***FIRST EXECUTABLE STATEMENT  ISORT
      NN = N
c      IF (NN .LT. 1) THEN
c         CALL XERMSG ('SLATEC', 'ISORT',
c     +      'The number of values to be sorted is not positive.', 1, 1)
c         RETURN
c      ENDIF
C
      KK = ABS(KFLAG)
c      IF (KK.NE.1 .AND. KK.NE.2) THEN
c         CALL XERMSG ('SLATEC', 'ISORT',
c     +      'The sort control parameter, K, is not 2, 1, -1, or -2.', 2,
c     +      1)
c         RETURN
c      ENDIF
C
C     Alter array IX to get decreasing order if needed
C
      IF (KFLAG .LE. -1) THEN
         DO 10 I=1,NN
            IX(I) = -IX(I)
   10    CONTINUE
      ENDIF
C
      IF (KK .EQ. 2) GO TO 100
C
C     Sort IX only
C
      M = 1
      I = 1
      J = NN
      R = 0.375E0
C
   20 IF (I .EQ. J) GO TO 60
      IF (R .LE. 0.5898437E0) THEN
         R = R+3.90625E-2
      ELSE
         R = R-0.21875E0
      ENDIF
C
   30 K = I
C
C     Select a central element of the array and save it in location T
C
      IJ = I + INT((J-I)*R)
      T = IX(IJ)
C
C     If first element of array is greater than T, interchange with T
C
      IF (IX(I) .GT. T) THEN
         IX(IJ) = IX(I)
         IX(I) = T
         T = IX(IJ)
      ENDIF
      L = J
```

```
C
C     If last element of array is less than than T, interchange with T
C
      IF (IX(J) .LT. T) THEN
         IX(IJ) = IX(J)
         IX(J) = T
         T = IX(IJ)
C
C        If first element of array is greater than T, interchange with T
C
         IF (IX(I) .GT. T) THEN
            IX(IJ) = IX(I)
            IX(I) = T
            T = IX(IJ)
         ENDIF
      ENDIF
C
C     Find an element in the second half of the array which is smaller
C     than T
C
   40 L = L-1
      IF (IX(L) .GT. T) GO TO 40
C
C     Find an element in the first half of the array which is greater
C     than T
C
   50 K = K+1
      IF (IX(K) .LT. T) GO TO 50
C
C     Interchange these elements
C
      IF (K .LE. L) THEN
         TT = IX(L)
         IX(L) = IX(K)
         IX(K) = TT
         GO TO 40
      ENDIF
C
C     Save upper and lower subscripts of the array yet to be sorted
C
      IF (L-I .GT. J-K) THEN
         IL(M) = I
         IU(M) = L
         I = K
         M = M+1
      ELSE
         IL(M) = K
         IU(M) = J
         J = L
         M = M+1
      ENDIF
      GO TO 70
C
C     Begin again on another portion of the unsorted array
C
   60 M = M-1
      IF (M .EQ. 0) GO TO 190
      I = IL(M)
      J = IU(M)
C
   70 IF (J-I .GE. 1) GO TO 30
      IF (I .EQ. 1) GO TO 20
      I = I-1
```

```
C
   80 I = I+1
      IF (I .EQ. J) GO TO 60
      T = IX(I+1)
      IF (IX(I) .LE. T) GO TO 80
      K = I
C
   90 IX(K+1) = IX(K)
      K = K-1
      IF (T .LT. IX(K)) GO TO 90
      IX(K+1) = T
      GO TO 80
C
C     Sort IX and carry IY along
C
  100 M = 1
      I = 1
      J = NN
      R = 0.375E0
C
  110 IF (I .EQ. J) GO TO 150
      IF (R .LE. 0.5898437E0) THEN
         R = R+3.90625E-2
      ELSE
         R = R-0.21875E0
      ENDIF
C
  120 K = I
C
C     Select a central element of the array and save it in location T
C
      IJ = I + INT((J-I)*R)
      T = IX(IJ)
      TY = IY(IJ)
C
C     If first element of array is greater than T, interchange with T
C
      IF (IX(I) .GT. T) THEN
         IX(IJ) = IX(I)
         IX(I) = T
         T = IX(IJ)
         IY(IJ) = IY(I)
         IY(I) = TY
         TY = IY(IJ)
      ENDIF
      L = J
C
C     If last element of array is less than T, interchange with T
C
      IF (IX(J) .LT. T) THEN
         IX(IJ) = IX(J)
         IX(J) = T
         T = IX(IJ)
         IY(IJ) = IY(J)
         IY(J) = TY
         TY = IY(IJ)
C
C        If first element of array is greater than T, interchange with T
C
         IF (IX(I) .GT. T) THEN
            IX(IJ) = IX(I)
            IX(I) = T
            T = IX(IJ)
```

```
            IY(IJ) = IY(I)
            IY(I) = TY
            TY = IY(IJ)
         ENDIF
      ENDIF
C
C     Find an element in the second half of the array which is smaller
C     than T
C
  130 L = L-1
      IF (IX(L) .GT. T) GO TO 130
C
C     Find an element in the first half of the array which is greater
C     than T
C
  140 K = K+1
      IF (IX(K) .LT. T) GO TO 140
C
C     Interchange these elements
C
      IF (K .LE. L) THEN
         TT = IX(L)
         IX(L) = IX(K)
         IX(K) = TT
         TTY = IY(L)
         IY(L) = IY(K)
         IY(K) = TTY
         GO TO 130
      ENDIF
C
C     Save upper and lower subscripts of the array yet to be sorted
C
      IF (L-I .GT. J-K) THEN
         IL(M) = I
         IU(M) = L
         I = K
         M = M+1
      ELSE
         IL(M) = K
         IU(M) = J
         J = L
         M = M+1
      ENDIF
      GO TO 160
C
C     Begin again on another portion of the unsorted array
C
  150 M = M-1
      IF (M .EQ. 0) GO TO 190
      I = IL(M)
      J = IU(M)
C
  160 IF (J-I .GE. 1) GO TO 120
      IF (I .EQ. 1) GO TO 110
      I = I-1
C
  170 I = I+1
      IF (I .EQ. J) GO TO 150
      T = IX(I+1)
      TY = IY(I+1)
      IF (IX(I) .LE. T) GO TO 170
      K = I
C
```

```
  180 IX(K+1) = IX(K)
      IY(K+1) = IY(K)
      K = K-1
      IF (T .LT. IX(K)) GO TO 180
      IX(K+1) = T
      IY(K+1) = TY
      GO TO 170
C
C     Clean up
C
  190 IF (KFLAG .LE. -1) THEN
         DO 200 I=1,NN
            IX(I) = -IX(I)
  200    CONTINUE
      ENDIF
C
      RETURN
      END
```

I.5. Subroutine select (file select03.f)

```
      subroutine select(nv, nb, nsize, nstop, len, npr, cor, pm, avol)


!.........Selection solution * * * Generational GA * * *


!..........declare variables
      dimension j1(11000), j2(11000)
      dimension ix(11000), iy(11000), iz(11000)
      dimension ihv(1000)
      dimension ipop(300,11000)
      dimension itemp(300,11000)
      double precision a1, a2, a3
      double precision avol, ave, r
      double precision bx(1000), by(1000)
      double precision fit(300)

      common /lidar/ ix, iy, iz
      common /conv/ multx, minx, multy, miny, multz, minz
      common /populace/ ipop
      common /bound/ bx, by
      common /hulls/ nh, ihv
      common /health/ fit


      igen = 0

      print *, 'Generation'


    5 igen = igen + 1
      print *, igen

!..........zero out the fitness array
      do i = 1, 300
        fit(i) = 0.d00
      end do

!..........determine the fitness of each individual in the populace
!      print *, 'calling ftnss'
      call ftnss (nv, nb, nsize, avol, len, npr)


!..........determine the median fitness of the populace
!      print *, 'calling median'
      call median ( nsize, m4 )

!..........determine the average fitness of the populace
!      print *, 'calling average'
      call average ( nsize, ave )

!..........determine the root-mean-squared-error (rmse) of the populace
!      print *, 'calling rmse'
      call rmse ( nsize, ave, r)

!..........determine the worst fit individual in the populace
!      print *, 'calling zlast'
      call last ( nsize, m3 )
```

```
!..........elitism (finding the best 2 individuals in populace based on fitness)
!      print *,'individuals 1 & 2'
!      print *,'-----------------'

!      print *, 'calling best2'
      call best2 ( nsize, m1, m2 )

!      print *, 'm1=', m1, '  m2=', m2, '  m3=', m3, '   m4=', m4


!..........determining who in the population has the best, median,and worst
fitness
      k1 = 0
!      do i = 1, len
      do i = 1, nv
        if (ipop(m1,i) .eq. 1) k1 = k1 + 1
      end do
!      print *, 'Best(m1):', m1, (ipop(m1,i), i = 1, nv)

      k4 = 0
!      do i = 1, len
      do i = 1, nv
        if (ipop(m4,i) .eq. 1) k4 = k4 + 1
      end do
!      print *, 'Median(m4):', m4, (ipop(m4,i), i = 1, nv)

      k3 = 0
!      do i = 1, len
      do i = 1, nv
        if (ipop(m3,i) .eq. 1) k3 = k3 + 1
      end do
!      print *, 'Worst(m3):', m3, (ipop(m3,i), i = 1, nv)


!..........printing to UNIT 22
      write(22,1000)igen,k1,fit(m1),ave,k4,fit(m4),k3,fit(m3),r
 1000 format(2(i6,','),2(e15.3,','),2(i6,',',e12.3,','),e15.5)


!..........placing the best two individuals from OLD population (ipop) into the
!............TEMPORARY population (itemp)
!      print *, 'writing 1st & 2nd individuals to temp array'
!      do i = 1, len
      do i = 1, nv
        itemp (1, i) = ipop (m1, i)
        itemp (2, i) = ipop (m2, i)
      end do

!
!..........selecting 2 parents
      do j = 3, nsize, 2
!        print *, 'j=', j
!        print *,'individuals', j,'&', j+1
!        print *,'------------------------------------'

!        print *, 'calling tourn'
        call tourn ( nsize, m1, m2 )

!        print *, 'm1=', m1, 'm2=', m2

!..........creating 2 single parents arrays: j1 and j2
!        print *, 'creating 2 parents'
        do i = 1, len
```

```
          j1 (i) = ipop (m1, i)
          j2 (i) = ipop (m2, i)
        end do

!...........crossover of parents to create 2 children
!        print *, 'calling cross over'
        call xover ( nv, j1, j2, len, m1, m2, cor)

!...........mutating (probability based) the children (two)
!        print  *, 'calling mutation'
        call mutate ( nv, len, j1, m1, pm )
!        print *, 'calling mutation'
        call mutate ( nv, len, j2, m2, pm )

!...........placing the 2 children into the TEMPORARY population
!        print *, 'placing j1 & j2 into temp array'
!        do i = 1, len
        do i = 1, nv
          itemp (j, i) = j1 (i)
          itemp (j+1, i) = j2 (i)
        end do
      end do


!..........reintializing the TEMPORARY population to zero
!      print *, 'putting the temp array back into the ipop array'
      do i = 1, nsize
!        do j = 1, len
        do j = 1, nv
          ipop (i,j) = itemp (i,j)
          itemp (i,j) = 0
        end do
      end do

      if (igen .lt. nstop) goto 5


!..........print best individual after simulating to UNIT 23
      open (23, file='bi_out.csv', status='REPLACE')
      write (23, '(a)') 'BEST INDIVIDUAL'
      write (23, '(a)') 'X, Y, Z'
      write (23, '(a)') ' '
      write (23,*) (ipop(1,i), i=1,nv)
      write (23, '(a)') ' '

      j = 0
!      do i = 1, len
      do i = 1, nv
        if (ipop(1,i) .eq. 1) then
          a1 = dfloat(ix(i)+minx) / dfloat(multx)
          a2 = dfloat(iy(i)+miny) / dfloat(multy)
          a3 = dfloat(iz(i)+minz) / dfloat(multz)
          j = j + 1
          write(23, '(4(i10))') i, ix(i),iy(i),iz(i)
!          write(23,'(2(e15.5,a),e15.5)') a1, ',', a2, ',', a3
        endif
      end do
      print *, ' '
      print *, 'Best individual has', j, '  points'
      close (23)

      return
      end
```

I.6. Subroutine ftnss (file ftnss04.f)

```
      subroutine ftnss (nv, nb, nsize, avol, len, npr)


!.........declaring variables
      dimension ix(11000), iy(11000), iz(11000)
      dimension ixcor(11000), iycor(11000), izcor(11000)
      dimension icon(6,25000)
      integer til(3,25000)
      integer tnbr(3,25000)
      dimension ipop(300,1000)
      double precision vol, avol
      double precision fit(300)
      double precision xcor(11000), ycor(11000), zcor(11000)
      double precision bx(1000), by(1000)


      common /lidar/ ix, iy, iz
      common /populace/ ipop
      common /bound/ bx, by
      common /health/ fit
      common /triang/ icon


!      print *, 'nsize=', nsize, '     nv=', nv
!........creating individual coordinate arrays for bits > 0
      do j = 1, nsize
        ii = 0
!        do i = 1, len
        do i = 1, nv
          if (ipop(j,i) .ne. 0) then
            ii = ii + 1
            ixcor(ii) = ix(i)
            iycor(ii) = iy(i)
            izcor(ii) = iz(i)
            xcor(ii)  = dfloat(ixcor(ii))
            ycor(ii)  =   dfloat(iycor(ii))
            zcor(ii)  =   dfloat(izcor(ii))
!            print *,'i=',i,'ii=',ii,ix(i),iy(i),iz(i)
!            print *,'i=',i,'ii=',ii,xcor(ii),ycor(ii),zcor(ii)
          end if
        end do


!........generating TINs
        call test0704 (ii, xcor(1:ii), ycor(1:ii), ntins, til, tnbr)
!        print *, 'ii=', ii, '    ntins=', ntins


        do k = 1, ntins
          icon(1,k) = tnbr(1,k)
          icon(2,k) = tnbr(2,k)
          icon(3,k) = tnbr(3,k)
          icon(4,k) = til(1,k)
          icon(5,k) = til(2,k)
          icon(6,k) = til(3,k)
        end do
        do k = ntins+1, 25000
          icon(1,k) = 0
          icon(2,k) = 0
          icon(3,k) = 0
```

```
          icon(4,k) = 0
          icon(5,k) = 0
          icon(6,k) = 0
        end do


!........deletes TINs outside of the boundary & update TIN matrix
        call zxc2 (ii, ntins, nb, xcor(1:ii), ycor(1:ii))


!........determine the volume from updated TIN matrix
        vol = 0.d00
        call calcvol(ii,ntins,ixcor(1:ii),iycor(1:ii),izcor(1:ii),vol)

        fit(j) = dabs( vol - avol )

!........penalizing fitness if ii > nprp
        if (ii .gt. npr) then
          fit(j) = fit(j) * (10.d0**dfloat(ii-npr))
        else
          fit(j) = fit(j) * (dfloat(1+npr-ii))
        end if


!...........checking for "not a number" value (NaN)
        if(isnan(fit(j))) fit(i) = 10.d0**50.d0


!        write(*,*) ' ind',j, ':  size=',ii, '  vol =',vol, 'fit=',
!     1dabs(vol-avol), '  adjusted fit =', fit(j)

!        write (*, '(1x,3i6,2e20.7)') j, ii, (ii-npr), vol, fit(j)


      end do


      return
      end
```

I.7. Subroutine average (file average02.f)

```
!.........calculates average fitness

      subroutine average ( itp, ave )

      double precision sum, outfb, ave
      double precision f(300)

      common /health/ f

      m = 0
      outfb = 10.d0**50.d0
      sum = 0.d0
      ave = 0.d0

      do i = 1, itp
!        if (f(i) .lt. outfb) then
          sum = sum + f(i)
          m = m + 1
!        end if
      end do

      ave = sum / dfloat(m)

      return
      end
```

I.8. Subroutine median (file median01.f)

```
!.........calculates average fitness


      subroutine median ( nsize, m4 )


      integer ifit(300), ib(300)
      double precision f(300)


      common /health/ f


!      print *, '   i    fitness     integer_fitness    who in populace'
      do i = 1, nsize
        if(f(i) .gt. 100000000) then
          ifit(i) = 100000000
          ib(i) = i
        else
          ifit(i) = abs(int(f(i)))
          ib(i) = i
        end if
!        print *, i, f(i), ifit(i), ib(i)
      end do

      k = nsize


!..........sorting both ifit and ib arrays
      call isort(ifit, ib, k, 2)


!      print *, '   i          fitness          who in populace'
!      do i = 1, nsize
!        print *, i, ifit(i), ib(i)
!      end do

      k = nsize/2

!.........median point
      m4 = ib(k)

!      print *, 'm4=', m4


      return
      end
```

I.9. Subroutine best2 (file best202.f)

```
!
!............elitism--find the 2 best fitnesses in the entire population
!...............equal or less than the weight limit


      subroutine best2 ( itp, m1, m2 )


      double precision f(300)


      common /health/ f


!      print *,'Elitism: find 2 best indivials in population (',itp,')'


!......find best value from fitness
      m1 = 1
      do i = 2, itp
        if (f(i) .lt. f(m1)) m1 = i
!        print *, '** f(', m1,')=', f(m1), '  f(',i,')=', f(i)
      end do


!      print *, 'best=', m1, '  (fitness=', f(m1), ')'


!......now find second best value
      m2 = 1
      if (m1 .eq. 1) m2 = 2
      do i=2,itp
        if (i .eq. m1) cycle
        if (f(i) .lt. f(m2)) m2 = i
!        print *, '** f(', m2,')=', f(m2), '  f(',i,')=', f(i)
      end do


!      print *, 'second best=', m2, '  (fitness=', f(m2), ')'


      return
      end
```

I.10. Subroutine rmse (file rmse02.f)

```
!
!..........calculates the root-mean-squared error of fitness


      subroutine rmse ( itp, ave, r )


      double precision ave, outfb, sum, r
      double precision f(300)


      common /health/ f


      m = 0
      outfb = 10.d0**50.d0
      sum = 0.d0
      r = 0.d0


      do i = 1, itp
        if(f(i) .lt. outfb) then
          sum = sum + (f(i) - ave)**2.d0
          m = m + 1
        end if
      end do


      r = dsqrt(sum / dfloat(m))


      return
      end
```

I.11. Subroutine last (file last02.f)

```
!
!..........find the worst fitness


      subroutine last ( itp, m3 )


      double precision outfb
      double precision f(300)


      common /health/ f


      m3 = 1
      outfb = 10.**50.


      do  i = 2, itp
!         if (f(i) .ge. outfb) cycle
        if (f(i) .gt. f(m3)) m3 = i
      end do


      return
      end
```

I.12. Subroutine tourn (file tourn02.f)

```
!
      subroutine tourn ( itp, m1, m2 )

      integer ir(7)
      double precision f(300)

      common /health/ f

!.........select 3 individuals in the population by random choice
      ii=3

!       print *,'select best of', ii, 'individuals (#, fitness)'
      do i = 1, ii
    5   call random_number ( rval )
        ir(i) = int(rval * float(itp) + 0.5)
        if (ir(i) .eq. 0) goto 5
!         print *, ir(i), ':', f(ir(i))
      end do


!.........sorting by fitness
      do j = 1, ii
        do i = 1, ii-1
          if (f(ir(i+1)) .lt. f(ir(i))) then
            ia = ir(i)
            ir(i) = ir(i+1)
            ir(i+1) = ia
          end if
        end do
      end do


!...........the 2 best
      m1 = ir(1)
      m2 = ir(2)


!       print *, 'best=', m1
!       print *, 'second best=', m2
!       print *, ''


      return
      end
```

I.13. Subroutine xover (file xover03.f)

```
!
!.........performs random two-point crossover between 2 individuals
!...........and only middle portion gets switched


      subroutine xover ( nv, j1, j2, len, m1, m2, pc )


      integer j1(11000), j2(11000)
      integer jj1(11000), jj2(11000)


!...........determining if crossover will occur with probability of pc%
!.............NO CROSSOVER if > pc%
      call random_number ( rval )
      if (rval .gt. pc) goto 40


!...........uniform cross over
      do i = 1, nv
    5   call random_number(rval)
        if (rval .le. 0.5) then
          jj1(i) = j1(i)
          jj2(i) = j2(i)
        else
          jj1(i) = j2(i)
          jj2(i) = j1(i)
        end if
      end do

      do i = 1, nv
        j1(i) = jj1(i)
        j2(i) = jj2(i)
      end do

   40 continue

      return
      end
```

I.14. Subroutine mutat (file mutat04.f)

```
!
!.........Causes possible mutation of children based on a small probability
!...........Mutation
      subroutine mutate (nv, len, k, m, pm)
!
      integer k(11000)
      integer ihv(1000)


      common /hulls/ nh,ihv
!
!
!...........determining mutation at each bit in the string with a
!............probability of Pm=1/N
      prob = 1. - pm
!

      do i = 1, nv
        call random_number(rval)
        if(rval.lt.prob) cycle
        if(k(i).eq.0) then
          k(i)=1
          else
          k(i)=0
        end if
      end do

!...........redefining hulls to be included (=1)
      do i = 1, nh
        k(ihv(i))=1
      end do

      return
      end
!
```

I.15. Subroutine calcvol (file calcvol02.f)

```
      subroutine calcvol (ii, ntins, ixcor, iycor, izcor, vol_sum)


      dimension ixcor(11000), iycor(11000), izcor(11000)
      dimension icon(6,25000)
      dimension kx(4), ky(4), kz(4)
      double precision area, height, vol_tin, vol_sum


      common /conv/ multx, minx, multy, miny, multz, minz
      common /triang/ icon


!      print *, 'printing tins from inside the volume subroutine'
!      do i = 1, ntins
!        print *, (icon(j,i), j=1,6)
!      end do

!      print *, 'printing integer ixcor, iycor, izcor arrays'
!      do i = 1, ii
!        print *, i, ixcor(i), iycor(i), izcor(i)
!      end do


      vol_sum = 0.d00
      vol_tin = 0.d00

!      print *, ''

      do i = 1, ntins

!        print *, 'tin=', i
        if (icon(1,ntins) .lt. 0) cycle
!        print *, 'tin=', i, ':', (icon(j,i), j=4,6)


!.........setting up the triangle
        do j = 1, 3
          kx(j) = ixcor(icon(j+3,i))
          ky(j) = iycor(icon(j+3,i))
          kz(j) = izcor(icon(j+3,i))
!          print *, j, '   icon=', icon(j+3,i), '   x=', kx(j), '   y=', ky(j), '
z=', kz(j)
        end do

!...............the 4th point = 1st point
        kx(4) = kx(1)
        ky(4) = ky(1)

!.........calculating area of triangle
        area = 0.d00
        height = 0.d00

        do j = 1, 3
          area = area + dfloat(kx(j)*ky(j+1) - kx(j+1)*ky(j))
        end do
        area = area/2.0d0

!.........converting area to original units
        area = area/(dfloat(multx) * dfloat(multy))
```

```
!.........calculating average height of prism
        height = (dfloat(kz(1)+kz(2)+kz(3)))/3.d00

!.........converting height to original units
        height = height/dfloat(multz)

!.........calculating volume of the prism in orginal units
        vol_tin = area * height

!.........summing the volume
        vol_sum = vol_sum + vol_tin

!        print *, '   area=', area, '   volume=', vol_tin

      end do

!      print *, 'total volume=', vol_sum
!      print *, ''


!..........end of volume program


      return
      end
```

I.16. Subroutine locpt (file locpt02.f) (from http://jblevins.org/mirror/amiller/locpt.f90; See Appendix M for copyright information)

```
!
      SUBROUTINE locpt (x0, y0, n, l, m)
!-----------------------------------------------------------------------
! GIVEN A POLYGONAL LINE CONNECTING THE VERTICES (X(I),Y(I)) (I = 1,...,N)
! TAKEN IN THIS ORDER.  IT IS ASSUMED THAT THE POLYGONAL PATH IS A LOOP,
! WHERE (X(N),Y(N)) = (X(1),Y(1)) OR THERE IS AN ARC FROM (X(N),Y(N)) TO
! (X(1),Y(1)).  N.B. The polygon may cross itself any number of times.

! (X0,Y0) IS AN ARBITRARY POINT AND L AND M ARE VARIABLES.
! On output, L AND M ARE ASSIGNED THE FOLLOWING VALUES ...

!    L = -1   IF (X0,Y0) IS OUTSIDE THE POLYGONAL PATH
!    L =  0   IF (X0,Y0) LIES ON THE POLYGONAL PATH
!    L =  1   IF (X0,Y0) IS INSIDE THE POLYGONAL PATH

! M = 0 IF (X0,Y0) IS ON OR OUTSIDE THE PATH.  IF (X0,Y0) IS INSIDE THE
! PATH THEN M IS THE WINDING NUMBER OF THE PATH AROUND THE POINT (X0,Y0).

! Fortran 66 version by A.H. Morris
! Converted to ELF90 compatibility by Alan Miller, 15 February 1997

!-----------------------

      double precision x0, y0
      double precision x(1000), y(1000)

!     Local variables
      double precision angle, eps, pi, pi2, sum, theta
      double precision theta1, thetai, tol, u, v


      common /bound/ x, y


!     ****** EPS IS A MACHINE DEPENDENT CONSTANT. EPS IS THE
!            SMALLEST NUMBER SUCH THAT 1.0 + EPS > 1.0

      eps = EPSILON(1.0)

!-----------------------------------------------------------------------
      n0 = n
      IF (x(1) == x(n) .AND. y(1) == y(n)) n0 = n - 1
      pi = DATAN2(0.0D00, -1.0D00)
      pi2 = 2.0*pi
      tol = 4.0*eps*pi
      l = -1
      m = 0

      u = x(1) - x0
      v = y(1) - y0
      IF (u == 0.0 .AND. v == 0.0) GO TO 20
      IF (n0 < 2) RETURN
      theta1 = DATAN2(v, u)

      sum = 0.0
      theta = theta1
      DO i = 2, n0
        u = x(i) - x0
```

```
          v = y(i) - y0
          IF (u == 0.0 .AND. v == 0.0) GO TO 20
          thetai = DATAN2(v, u)

          angle = DABS(thetai - theta)
          IF (DABS(angle - pi) < tol) GO TO 20
          IF (angle > pi) angle = angle - pi2
          IF (theta > thetai) angle = -angle
          sum = sum + angle
          theta = thetai
        END DO

        angle = DABS(theta1 - theta)
        IF (DABS(angle - pi) < tol) GO TO 20
        IF (angle > pi) angle = angle - pi2
        IF (theta > theta1) angle = -angle
        sum = sum + angle

!       SUM = 2*PI*M WHERE M IS THE WINDING NUMBER

        m = DABS(sum)/pi2 + 0.2
        IF (m == 0) RETURN
        l = 1
        IF (sum < 0.0) m = -m
        RETURN

!       (X0, Y0) IS ON THE BOUNDARY OF THE PATH

   20   l = 0

        RETURN
! END SUBROUTINE locpt
        END
```

I.17. Subroutine zxc2 (file zxc01.f)

```
      subroutine zxc2 ( nv, nt, nb, x, y)


      integer A(11000), B(11000)
      dimension kx(11000), ky(11000)
      dimension icon(6,25000)
      double precision cx, cy, cxave, cyave
      double precision x(11000), y(11000)

      common /triang/ icon


!.......nv   = number of points/vertices in his individual
!.......nt   = number of TINS in this individual


!      print *, ''
!      print *, 'Entered subroutine zxc'
!      print *, 'nv=', nv, '     nt=', nt
!.........converting x & y to interger arrays (kx, ky)
      do i = 1, nv
        kx(i) = int(x(i))
        ky(i) = int(y(i))
      end do


!.........finding tins inside boundary
!      print *, ''
!      print *, 'finding tins inside boundary'
!      print *, '  10pt sweep per tin . . . .  .  .'
!      print *, ''
!.........1st ==> sweeping using centroids--this finds ~99% of the triangles
      j = 0
      do i = 1, nt
!        print *, 'TIN:', i, '      vertices=', icon(4,i), icon(5,i), icon(6,i)
!        print *, '1:  x=', kx(icon(4,i)),'  y=', ky(icon(4,i))
!        print *, '2:  x=', kx(icon(5,i)),'  y=', ky(icon(5,i))
!        print *, '3:  x=', kx(icon(6,i)),'  y=', ky(icon(6,i))
        cx = dfloat(kx(icon(4,i))+kx(icon(5,i))+kx(icon(6,i)))/3.0d00
        cy = dfloat(ky(icon(4,i))+ky(icon(5,i))+ky(icon(6,i)))/3.0d00
!        print *, 'Centroid:  cx=', cx, '   cy=', cy
        call locpt (cx, cy, nb, l, m)
!        print *, 'l=' ,l
        A(i) = 1
        if ( l .lt. 0 ) then
          j = j + 1
          B(j) = i
        end if

!.........2nd sweep on the triangle edges (3 per side of triangle)
!...........catch ~99.9% of the triangles
        if ( l .ge. 0) then
          do ii = 1, 3
            select case ( ii )
              case ( 1 )
!..............vertice #1 of triangle
                kx1 = kx(icon(4,i))
                kx2 = kx(icon(5,i))
                ky1 = ky(icon(4,i))
                ky2 = ky(icon(5,i))
```

```
              case (  2 )
!...............vertice #2 of triangle
                kx1 = kx(icon(5,i))
                kx2 = kx(icon(6,i))
                ky1 = ky(icon(5,i))
                ky2 = ky(icon(6,i))
              case ( 3 )
!...............vertice #3 of triangle
                kx1 = kx(icon(6,i))
                kx2 = kx(icon(4,i))
                ky1 = ky(icon(6,i))
                ky2 = ky(icon(4,i))
            end select
!...............determine if mid-point on line is outside the boundary
            cxave = dfloat(kx1 + kx2)/2.0d00
            cyave = dfloat(ky1 + ky2)/2.0d00
            call locpt (cxave, cyave, nb, l, m)
            A(i) = l
            if (l .lt. 0) then
              j = j + 1
              B(j) = i
!              print *, 'outside from 2nd sweep'
              exit
            end if
!...............determine if point that is 1/3 away is outside the boundary
            cx = (dfloat(kx1) + cxave)/2.0d00
            cy = (dfloat(ky1) + cyave)/2.0d00
            call locpt (cx, cy, nb, l, m)
            A(i) = l
            if (l .lt. 0) then
              j = j + 1
              B(j) = i
!              print *, 'outside from 2nd sweep'
              exit
            end if
!...............determine if point that is 2/3 away is outside the boundary
            cx = (dfloat(kx2) + cxave)/2.0d00
            cy = (dfloat(ky2) + cyave)/2.0d00
            call locpt (cx, cy, nb, l, m)
            A(i) = l
            if (l .lt. 0) then
              j = j + 1
              B(j) = i
!              print *, 'outside from 2nd sweep'
              exit
            end if
          end do
        end if

      end do

      nost = j
!      print *, 'number of outside tins from sweep=', nost

!      print *, 'writing tin number of outside tins'
!      do k = 1, j
!        print *, B(k)
!      end do



!.........updating the TIN from the sweeps if there are updates
```

```
      if (nost .ne. 0) then
!        print *, 'updating the tin matrix due to the sweeps'
        do i = 1, nt
          do k = 1, nost
            if (icon(1,i) .eq. B(k)) icon(1,i) = 0
            if (icon(2,i) .eq. B(k)) icon(2,i) = 0
            if (icon(3,i) .eq. B(k)) icon(3,i) = 0
            if (B(k) .eq. i) then
              icon(1,i) = -99
              icon(2,i) = -99
              icon(3,i) = -99
            end if
          end do
!          print *, (icon(k,i), k = 1, 6)
        end do
!      else
!        print *, 'there are no tins outside the polygon'
      end if


      return
      end
```

I.18. GEOMPACK code (file test0704.f) (modified from GEOMPACK—used for developing Delaunay triangulation. The original download (1/18/10) was on the web at http://orion.math.iastate.edu/burkardt/f_src/geompack/geompack_prb.f90. The code on 10/29/17 resides at https://people.sc.fsu.edu/~jburkardt/f77_src/geompack/geompack.f; See Appendix N for copyright information)

```
  subroutine test0704 (npt, xvcl, yvcl, ntri, til, tnbr)

!
!*********************************************************************
!
!! TEST07 tests DTRIW2;
!
  implicit none
!
!  double precision, parameter :: large = 1000.0D+00
  integer, parameter :: maxnp = 25000
  integer, parameter :: maxst = 25000
!
  integer a
!  integer alg
  integer b
  double precision binexp
  integer c
  integer d
  integer diaedg
  integer i
  integer ierror
  integer ind(maxnp+3)
  integer j
  integer jp1
  integer jp2
  integer k
  integer msglvl
  integer nlo
  integer npt
  integer ntri
  integer stack(maxst)
!  integer til(3,maxnp*2+1)
  integer til(3,25000)
!  integer tnbr(3,maxnp*2+1)
  integer tnbr(3,25000)
  double precision xvcl(11000)
  double precision yvcl(11000)
!  double precision vcl(2,maxnp+3)
  double precision vcl(2,11000)
!
!  ALG =
!    2: DTRIW2;
!    3: DTRIW2 with bounding triangle;
!    4: DTRIW2 with call to BNSRT2 first.
!  MSGLVL
!     0: print arrays;
!     4: also print edges as they are created and swapped.
!
!  I HAVE NO IDEA HOW TO CHOOSE BINEXP
!
  msglvl = 0
  binexp = 0.5D+00
```

```
   do i = 1, npt
     vcl(1,i) = xvcl(i)
     vcl(2,i) = yvcl(i)
   end do

!  npt = 24

!  write ( *, '(a)' ) ' '
!  write ( *, '(a)' ) 'TEST07'
!  write ( *, '(a,i6)' ) '  MSGLVL = ', msglvl
!  write ( *, '(a,i6)' ) '  NPT =    ', npt
!  write ( *, '(a,g14.6)' ) '  BINEXP = ', binexp

!  if ( npt > maxnp ) then
!    write ( *, '(a)' ) ' '
!    write ( *, '(a)' ) 'TEST07 - Error!'
!    write ( *, '(a)' ) '  NPT > MAXNP.'
!    return
!  end if

!  write ( *, '(a)' ) ' '
!  write ( *, '(a,i6)' ) '  The number of points to triangulate is ', npt
!  write ( *, '(a)' ) ' '
!  write ( *, '(a)' ) '  The coordinates of the points are:'
!  write ( *, '(a)' ) ' '
!  do i = 1, npt
!    write ( *, '(i5,2f15.7)' ) i, vcl(1,i), vcl(2,i)
!  end do

!  do alg = 2, 4

!    npt = 24

!    write ( *, '(a,i6)' ) 'ALG =    ', alg

!    if ( alg /= 3 ) then

      do i = 1, npt
        ind(i) = i
      end do

!    else

!      vcl(1,npt+1) = -large
!      vcl(2,npt+1) = -large
!      vcl(1,npt+2) = large
!      vcl(2,npt+2) = -large
!      vcl(1,npt+3) = 0.0D+00
!      vcl(2,npt+3) = large
!      ind(1) = npt + 1
!      ind(2) = npt + 2
!      ind(3) = npt + 3
!      do i = 1, npt
!        ind(i+3) = i
!      end do

!      npt = npt + 3

!    end if

!    if ( alg == 4 ) then
!      call bnsrt2 ( binexp, npt, vcl, ind, til, tnbr )
!    end if
```

```
    call dtriw2 ( npt, maxst, vcl, ind, ntri, til, tnbr, stack, ierror )

!    if ( ierror /= 0 ) then
!      write ( *, '(a)' ) ' '
!      write ( *, '(a)' ) 'TEST07 - Error!'
!      write ( *, '(a,i6)' ) '  IERROR = ', ierror
!      return
!    end if

    nlo = 0

    do i = 1, ntri

      do j = 1, 3

        k = tnbr(j,i)

        if ( k > i ) then

          jp1 = j + 1
          if ( jp1 > 3) then
            jp1 = 1
          end if

          jp2 = jp1 + 1

          if ( jp2 > 3) then
            jp2 = 1
          end if

          a = til(j,i)
          b = til(jp1,i)
          c = til(jp2,i)

          if ( til(1,k) == b ) then
            d = til(3,k)
          else if ( til(2,k) == b ) then
            d = til(1,k)
          else
            d = til(2,k)
          end if

          if ( diaedg(vcl(1,c),vcl(2,c),vcl(1,a),vcl(2,a),vcl(1,d), &
               vcl(2,d),vcl(1,b),vcl(2,b)) == 1 ) then
            nlo = nlo + 1
          end if

        end if

      end do

    end do

!!    write ( *, '(a)' ) ' '
!    write ( *, '(a,i6)' ) '  NLO =  ', nlo

!    call delaunay_print ( npt, vcl, ntri, til, tnbr )

!!   write ( *, '(a,i6)' ) 'No. of triangles =', ntri
!!   write ( *, '(a)' ) ' '
!!   do i = 1, ntri
!!     write(*,'(7i6)')i,tnbr(1,i),tnbr(2,i),tnbr(3,i),til(1,i),til(2,i),til(3,i)
```

```
!!   end do

!   end do

   return
!   stop
end

!-------------------------------------------------------------------------------

function diaedg ( x0, y0, x1, y1, x2, y2, x3, y3 )
!
!*******************************************************************************
!
!! DIAEDG chooses one of the diagonals of a quadrilateral.
!
!
!  Discussion:
!
!    The routine determines whether 0--2 or 1--3 is the diagonal edge
!    that should be chosen, based on the circumcircle criterion, where
!    (X0,Y0), (X1,Y1), (X2,Y2), (X3,Y3) are the vertices of a simple
!    quadrilateral in counterclockwise order.
!
!  Modified:
!
!    19 February 2001
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, double precision X0, Y0, X1, Y1, X2, Y2, X3, Y3, the
!    coordinates of the vertices of a quadrilateral, given in
!    counter clockwise order.
!
!    Output, integer DIAEDG, chooses a diagonal:
!    +1, if diagonal edge 02 is chosen;
!    -1, if diagonal edge 13 is chosen;
!     0, if the four vertices are cocircular.
!
  double precision ca
  double precision cb
  integer diaedg
  double precision dx10
  double precision dx12
  double precision dx30
  double precision dx32
  double precision dy10
  double precision dy12
  double precision dy30
  double precision dy32
  double precision s
  double precision tol
  double precision tola
  double precision tolb
  double precision x0
  double precision x1
```

```
  double precision x2
  double precision x3
  double precision y0
  double precision y1
  double precision y2
  double precision y3
!
  tol = 100.0D+00 * epsilon ( tol )

  dx10 = x1 - x0
  dy10 = y1 - y0
  dx12 = x1 - x2
  dy12 = y1 - y2
  dx30 = x3 - x0
  dy30 = y3 - y0
  dx32 = x3 - x2
  dy32 = y3 - y2

  tola = tol * max ( abs ( dx10 ), abs ( dy10 ), abs ( dx30 ), abs ( dy30 ) )
  tolb = tol * max ( abs ( dx12 ), abs ( dy12 ), abs ( dx32 ), abs ( dy32 ) )

  ca = dx10 * dx30 + dy10 * dy30
  cb = dx12 * dx32 + dy12 * dy32

  if ( ca > tola .and. cb  >  tolb ) then

    diaedg = -1

  else if ( ca < -tola .and. cb < -tolb ) then

    diaedg = 1

  else

    tola = max ( tola, tolb )
    s = ( dx10 * dy30 - dx30 * dy10 ) * cb + ( dx32 * dy12 - dx12 * dy32 ) * ca

    if ( s > tola ) then
      diaedg = -1
    else if ( s < -tola ) then
      diaedg = 1
    else
      diaedg = 0
    end if

  end if

  return
end

!-------------------------------------------------------------------------------

subroutine dtriw2 ( npt, maxst, vcl, ind, ntri, til, tnbr, stack, ierror )
!
!*******************************************************************************
!
!! DTRIW2 constructs an incremental Delaunay triangulation in 2D.
!
!
!  Purpose:
!
!    Construct Delaunay triangulation of 2-D vertices using
!    incremental approach and diagonal edge swaps. Vertices are
```

```
!    inserted one at a time in order given by IND array. The initial
!    triangles created due to a new vertex are obtained by a walk
!    through the triangulation until location of vertex is known.
!
!  Modified:
!
!    12 July 1999
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, integer NPT, the number of 2-D points (vertices).
!
!    Input, integer MAXST, the maximum size available for STACK array; should
!    be about NPT to be safe, but MAX(10,2*LOG2(NPT)) usually enough.
!
!    Input, double precision VCL(1:2,1:*), the coordinates of 2-D vertices.
!
!    Input, integer IND(1:NPT), indices in VCL of vertices to be triangulated;
!    vertices are inserted in order given by this array.
!
!    Output, integer NTRI, the number of triangles in triangulation; equal to
!    2*NPT - NB - 2 where NB = number of boundary vertices.
!
!    Output, integer TIL(1:3,1:NTRI), the triangle incidence list; elements
!    are indices of VCL; vertices of triangles are in counter clockwise order.
!
!    Output, integer TNBR(1:3,1:NTRI), the triangle neighbor list; positive
!    elements are indices of TIL; negative elements are used for links
!    of counter clockwise linked list of boundary edges; LINK = -(3*I + J-1)
!    where I, J = triangle, edge index; TNBR(J,I) refers to
!    the neighbor along edge from vertex J to J+1 (mod 3).
!
!    Workspace, integer STACK(1:MAXST), used for stack of triangles for which
!    circumcircle test must be made.
!
!    Output, integer IERROR, error flag. For abnormal return,
!    IERROR is set to 8, 224, 225, or 226.
!
  integer maxst
  integer npt
!
  integer bedg
  integer btri
  double precision cmax
  integer e
  integer em1
  integer ep1
  integer ntri
  integer i
  integer i3
  integer ierror
  integer ind(npt)
  integer j
  integer l
  integer ledg
  integer lr
```

```
  integer lrline
  integer ltri
  integer m
  integer m1
  integer m2
  integer m3
  integer, parameter :: msglvl = 0
  integer n
  integer redg
  integer rtri
  integer stack(maxst)
  integer t
  integer til(3,npt*2)
  integer tnbr(3,npt*2)
  integer top
  double precision tol
  double precision vcl(2,*)
!
  ierror = 0
  tol = 100.0D+00 * epsilon ( tol )
!
!  Determine the initial triangle.
!
  m1 = ind(1)
  m2 = ind(2)

  do j = 1, 2
    cmax = max ( abs ( vcl(j,m1) ), abs ( vcl(j,m2) ) )
    if ( abs ( vcl(j,m1) - vcl(j,m2) ) > tol * cmax .and. cmax > tol ) then
      go to 20
    end if
  end do

  ierror = 224
  return

20 continue

  i3 = 3

30 continue

  if ( i3 > npt ) then
    ierror = 225
    return
  end if

  m = ind(i3)
  lr = lrline ( vcl(1,m), vcl(2,m), vcl(1,m1), vcl(2,m1), vcl(1,m2), &
    vcl(2,m2), 0.0D+00 )

  if ( lr == 0 ) then
    i3 = i3 + 1
    go to 30
  end if

  if ( i3 /= 3 ) then
    ind(i3) = ind(3)
    ind(3) = m
  end if

  ntri = 1
```

```
  if ( lr == -1 ) then
    til(1,1) = m1
    til(2,1) = m2
  else
    til(1,1) = m2
    til(2,1) = m1
  end if

  til(3,1) = m
  tnbr(1,1) = -4
  tnbr(2,1) = -5
  tnbr(3,1) = -3

!!  if ( msglvl == 4 ) then
!!    write ( *,600) 1,vcl(1,m1),vcl(2,m1),vcl(1,m2),vcl(2,m2)
!!    write ( *,600) 1,vcl(1,m2),vcl(2,m2),vcl(1,m),vcl(2,m)
!!    write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m1),vcl(2,m1)
!!  end if
!
!  Insert vertices one at a time from anywhere.
!  Walk through the triangulation to determine the location of the new vertex.
!  Apply diagonal edge swaps until Delaunay triangulation of vertices
!  (so far) is obtained.
!
  top = 0

  do i = 4, npt

!!    if ( msglvl == 4 ) then
!!      write ( *,600) i
!!    end if

    m = ind(i)
    rtri = ntri

    call walkt2 ( vcl(1,m), vcl(2,m), ntri, vcl, til, tnbr, rtri, redg, ierror )

    if ( redg == 0 ) then

      m1 = til(1,rtri)
      m2 = til(2,rtri)
      m3 = til(3,rtri)
      til(3,rtri) = m

      if ( tnbr(1,rtri) > 0 ) then
        top = 1
        stack(top) = rtri
      end if

      ntri = ntri + 1
      til(1,ntri) = m2
      til(2,ntri) = m3
      til(3,ntri) = m
      n = tnbr(2,rtri)
      tnbr(1,ntri) = n

      if ( n > 0 ) then
        if ( tnbr(1,n) == rtri ) then
          tnbr(1,n) = ntri
        else if ( tnbr(2,n) == rtri ) then
          tnbr(2,n) = ntri
        else
          tnbr(3,n) = ntri
```

```
  end if
  top = top + 1
  stack(top) = ntri
end if

ntri = ntri + 1
til(1,ntri) = m3
til(2,ntri) = m1
til(3,ntri) = m
n = tnbr(3,rtri)
tnbr(1,ntri) = n

if ( n > 0 ) then
  if ( tnbr(1,n) == rtri ) then
    tnbr(1,n) = ntri
  else if ( tnbr(2,n) == rtri ) then
    tnbr(2,n) = ntri
  else
    tnbr(3,n) = ntri
  end if
  top = top + 1
  stack(top) = ntri
end if

tnbr(2,rtri) = ntri - 1
tnbr(3,rtri) = ntri
tnbr(2,ntri-1) = ntri
tnbr(3,ntri-1) = rtri
tnbr(2,ntri) = rtri
tnbr(3,ntri) = ntri - 1

if ( tnbr(1,ntri-1) <= 0 ) then

  t = rtri
  e = 1

  do

    if ( tnbr(e,t) <= 0 ) then
      exit
    end if

    t = tnbr(e,t)

    if ( til(1,t) == m2 ) then
      e = 3
    else if ( til(2,t) == m2 ) then
      e = 1
    else
      e = 2
    end if

  end do

  tnbr(e,t) = -3 * ntri + 3

end if

if ( tnbr(1,ntri) <= 0 ) then

  t = ntri - 1
  e = 1
```

```
        do

          if ( tnbr(e,t) <= 0 ) then
            exit
          end if

          t = tnbr(e,t)
          if ( til(1,t) == m3 ) then
            e = 3
          else if ( til(2,t) == m3 ) then
            e = 1
          else
            e = 2
          end if

        end do

        tnbr(e,t) = -3 * ntri

      end if

!!      if ( msglvl == 4 ) then
!!        write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m1),vcl(2,m1)
!!        write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m2),vcl(2,m2)
!!        write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m3),vcl(2,m3)
!!      end if

    else if ( redg < 0 ) then

      redg = -redg
      ltri = 0
      call vbedg ( vcl(1,m), vcl(2,m), vcl, til, tnbr, ltri, ledg, rtri, redg )
      n = ntri + 1
      l = -tnbr(ledg,ltri)

60    continue

        t = l / 3
        e = mod ( l, 3 ) + 1
        l = -tnbr(e,t)
        m2 = til(e,t)

        if ( e <= 2 ) then
          m1 = til(e+1,t)
        else
          m1 = til(1,t)
        end if

        ntri = ntri + 1
        tnbr(e,t) = ntri
        til(1,ntri) = m1
        til(2,ntri) = m2
        til(3,ntri) = m
        tnbr(1,ntri) = t
        tnbr(2,ntri) = ntri - 1
        tnbr(3,ntri) = ntri + 1
        top = top + 1

        if ( top > maxst ) then
          ierror = 8
          go to 100
        end if
```

```
        stack(top) = ntri

!!          if ( msglvl == 4 ) then
!!            write (*,600) 1,vcl(1,m),vcl(2,m),vcl(1,m2),vcl(2,m2)
!!          end if

      if ( t /= rtri .or. e /= redg ) then
        go to 60
      end if

!!        if ( msglvl == 4 ) then
!!          write (*,600) 1,vcl(1,m),vcl(2,m),vcl(1,m1),vcl(2,m1)
!!        end if

      tnbr(ledg,ltri) = -3*n - 1
      tnbr(2,n) = -3*ntri - 2
      tnbr(3,ntri) = -l

   else if ( redg <= 3 ) then

      m1 = til(redg,rtri)

      if ( redg == 1 ) then
        e = 2
        ep1 = 3
      else if ( redg == 2 ) then
        e = 3
        ep1 = 1
      else
        e = 1
        ep1 = 2
      end if

      m2 = til(e,rtri)
      til(e,rtri) = m
      m3 = til(ep1,rtri)

      if ( tnbr(ep1,rtri) > 0 ) then
        top = 1
        stack(top) = rtri
      end if

      ntri = ntri + 1
      til(1,ntri) = m
      til(2,ntri) = m2
      til(3,ntri) = m3
      n = tnbr(e,rtri)
      tnbr(2,ntri) = n
      tnbr(3,ntri) = rtri
      tnbr(e,rtri) = ntri

      if ( n > 0 ) then
        if ( tnbr(1,n) == rtri ) then
          tnbr(1,n) = ntri
        else if ( tnbr(2,n) == rtri ) then
          tnbr(2,n) = ntri
        else
          tnbr(3,n) = ntri
        end if
        top = top + 1
        stack(top) = ntri
      end if
```

```
!!      if ( msglvl == 4 ) then
!!        write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m3),vcl(2,m3)
!!      end if

      ltri = tnbr(redg,rtri)

      if ( ltri <= 0 ) then
        tnbr(1,ntri) = ltri
        tnbr(redg,rtri) = -3*ntri
        if ( tnbr(2,ntri) <= 0 ) then
          tnbr(1,ntri) = -3*ntri - 1
        end if

      else

        tnbr(1,ntri) = ntri + 1
        tnbr(redg,rtri) = ltri

        if ( til(1,ltri) == m2 ) then
          ledg = 1
          em1 = 2
          e = 3
        else if ( til(2,ltri) == m2 ) then
          ledg = 2
          em1 = 3
          e = 1
        else
          ledg = 3
          em1 = 1
          e = 2
        end if

        til(ledg,ltri) = m
        m3 = til(e,ltri)
        if ( tnbr(em1,ltri) > 0 ) then
          top = top + 1
          stack(top) = ltri
        end if
        ntri = ntri + 1
        til(1,ntri) = m2
        til(2,ntri) = m
        til(3,ntri) = m3
        tnbr(1,ntri) = ntri - 1
        tnbr(2,ntri) = ltri
        n = tnbr(e,ltri)
        tnbr(3,ntri) = n
        tnbr(e,ltri) = ntri

        if ( n > 0 ) then
          if ( tnbr(1,n) == ltri ) then
            tnbr(1,n) = ntri
          else if ( tnbr(2,n) == ltri ) then
            tnbr(2,n) = ntri
          else
            tnbr(3,n) = ntri
          end if
          top = top + 1
          stack(top) = ntri
        end if

!!        if ( msglvl == 4 ) then
!!          write ( *,600) 1,vcl(1,m),vcl(2,m),vcl(1,m3),vcl(2,m3)
!!        end if
```

```
    if ( tnbr(2,ntri-1) <= 0 ) then

      t = ntri
      e = 3

      do

        if ( tnbr(e,t) <= 0 ) then
          exit
        end if

        t = tnbr(e,t)
        if ( til(1,t) == m2 ) then
          e = 3
        else if ( til(2,t) == m2 ) then
          e = 1
        else
          e = 2
        end if

      end do

      tnbr(e,t) = -3 * ntri + 2

    end if

     if ( tnbr(3,ntri) <= 0 ) then

      t = ltri

      if ( ledg <= 2 ) then
         e = ledg + 1
      else
         e = 1
      end if

      do

        if ( tnbr(e,t) <= 0 ) then
          exit
        end if

        t = tnbr(e,t)
        if ( til(1,t) == m3 ) then
          e = 3
        else if ( til(2,t) == m3 ) then
          e = 1
        else
          e = 2
        end if

      end do

      tnbr(e,t) = -3 * ntri - 2

     end if

  end if

else
  ierror = 224
  go to 100
```

```
    end if

    btri = 0
    bedg = 0

    call swapec ( m, top, maxst, btri, bedg, vcl, til, tnbr, stack, ierror )

    if ( ierror /= 0 ) then
      exit
    end if

  end do

100 continue

  if ( i3 /= 3 ) then
    t = ind(i3)
    ind(i3) = ind(3)
    ind(3) = t
  end if

!!  if ( msglvl == 4 ) then
!!    write ( *,600) npt+1
!!  end if

!! 600 format (1x,i7,4f15.7)

  return
end

!------------------------------------------------------------------------------

function lrline ( xu, yu, xv1, yv1, xv2, yv2, dv )
!
!******************************************************************************
!
!! LRLINE determines if a point is left of, right or, or on a directed line.
!
!
!  Discussion:
!
!    The directed line is paralled to, and at a signed distance DV from
!    a directed base line from (XV1,YV1) to (XV2,YV2).
!
!  Modified:
!
!    14 July 2001
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, double precision XU, YU, the coordinates of the point whose
!    position relative to the directed line is to be determined.
!
!    Input, double precision XV1, YV1, XV2, YV2, the coordinates of two points
!    that determine the directed base line.
!
```

```
!    Input, double precision DV, the signed distance of the directed line
!    from the directed base line through the points (XV1,YV1) and (XV2,YV2).
!    DV is positive for a line to the left of the base line.
!
!    Output, integer LRLINE, the result:
!    +1, the point is to the right of the directed line;
!     0, the point is on the directed line;
!    -1, the point is to the left of the directed line.
!
  double precision dv
  double precision dx
  double precision dxu
  double precision dy
  double precision dyu
  integer lrline
  double precision t
  double precision tol
  double precision tolabs
  double precision xu
  double precision xv1
  double precision xv2
  double precision yu
  double precision yv1
  double precision yv2
!
  tol = 100.0D+00 * epsilon ( tol )

  dx = xv2 - xv1
  dy = yv2 - yv1
  dxu = xu - xv1
  dyu = yu - yv1

  tolabs = tol * max ( abs ( dx ), abs ( dy ), abs ( dxu ), &
    abs ( dyu ), abs ( dv ) )

  t = dy * dxu - dx * dyu + dv * sqrt ( dx * dx + dy * dy )

  if ( tolabs < t ) then
    lrline = 1
  else if ( -tolabs <= t ) then
    lrline = 0
  else
    lrline = -1
  end if

  return
end

!------------------------------------------------------------------------------

subroutine walkt2 ( x, y, ntri, vcl, til, tnbr, itri, iedg, ierror )
!
!******************************************************************************
!
!! WALKT2 searches for a triangle containing a point.
!
!
!  Purpose:
!
!    Walk through neighboring triangles of a 2-D Delaunay
!    triangulation until a triangle is found containing point (X,Y)
!    or (X,Y) is found to be outside the convex hull.  Search is
!    guaranteed to terminate for a Delaunay triangulation, else a
```

```
!    cycle may occur.
!
!  Modified:
!
!    14 July 2001
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, double precision X, Y, the coordinates of a 2-D point.
!
!    Input, integer NTRI, the number of triangles in the triangulation; used
!    to detect cycle.
!
!    Input, double precision VCL(2,1:*), the coordinates of 2-D vertices.
!
!    Input, integer TIL(3,NTRI), the triangle incidence list.
!
!    Input, integer TNBR(3,NTRI), the triangle neighbor list.
!
!    Input/output, integer ITRI.  On input, the index of triangle to begin
!    search at.  On output, the index of triangle that search ends at.
!
!    Output, integer IEDG, indicates the position of the point (X,Y) in
!    triangle ITRI.  A small tolerance is allowed in positions:
!    0, the interior of the triangle;
!    1, interior of edge 1;
!    2, interior of edge 2;
!    3, interior or edge 3;
!    4, vertex 1;
!    5, vertex 2;
!    6, vertex 3;
!    -1, outside convex hull, past edge 1;
!    -2, outside convex hull, past edge 2;
!    -3, outside convex hull, past edge 3.
!
!    Output, integer IERROR, error flag.  On abnormal return,
!    IERROR is set to 226.
!
  integer ntri
!
  integer a
  double precision alpha
  integer b
  double precision beta
  integer c
  integer cnt
  double precision det
  double precision dx
  double precision dxa
  double precision dxb
  double precision dy
  double precision dya
  double precision dyb
  double precision gamma
  integer i
  integer iedg
```

```
  integer ierror
  integer itri
  integer til(3,ntri)
  integer tnbr(3,ntri)
  double precision tol
  double precision vcl(2,*)
  double precision x
  double precision y
!
  ierror = 0
  tol = 100.0D+00 * epsilon ( tol )

  cnt = 0
  iedg = 0
  ierror = 0

  do

    cnt = cnt + 1

!!    if ( cnt > ntri ) then
!!      write ( *, '(a)' ) ' '
!!      write ( *, '(a)' ) 'WALKT2 - Fatal error!'
!!      write ( *, '(a)' ) '  All triangles have been searched.'
!!      ierror = 226
!!      return
!!    end if
!
!  Get the vertices of triangle ITRI.
!
    a = til(1,itri)
    b = til(2,itri)
    c = til(3,itri)
!
!  Using vertex C as a base, compute the distances to vertices A and B,
!  and the point (X,Y).
!
    dxa = vcl(1,a) - vcl(1,c)
    dya = vcl(2,a) - vcl(2,c)

    dxb = vcl(1,b) - vcl(1,c)
    dyb = vcl(2,b) - vcl(2,c)

    dx = x - vcl(1,c)
    dy = y - vcl(2,c)

    det = dxa * dyb - dya * dxb
!
!  Compute the barycentric coordinates of the point (X,Y) with respect
!  to this triangle.
!
    alpha = ( dx * dyb - dy * dxb ) / det
    beta = ( dxa * dy - dya * dx ) / det
    gamma = 1.0D+00 - alpha - beta
!
!  If the barycentric coordinates are all positive, then the point
!  is inside the triangle.
!
    if ( alpha > tol .and. beta > tol .and. gamma > tol ) then
      exit
    end if
!
!  If any barycentric coordinate is (strongly) negative with respect to
```

```
!  a side, and if that side is on the convex hull, the point is outside
!  the triangles, and we are done.
!
    if ( alpha < -tol ) then
      i = tnbr(2,itri)
      if ( i <= 0 ) then
        iedg = -2
        exit
      end if
    else if ( beta < -tol ) then
      i = tnbr(3,itri)
      if ( i <= 0 ) then
        iedg = -3
        exit
      end if
    else if ( gamma < -tol ) then
      i = tnbr(1,itri)
      if ( i <= 0 ) then
        iedg = -1
        exit
      end if
!
!  At least one barycentric coordinate is between -TOL and TOL,
!  and no barycentric coordinate is less than -TOL.  We are going
!  to assign the position to an edge or vertex.
!
    else if ( alpha <= tol ) then
      if ( beta <= tol ) then
        iedg = 6
      else if ( gamma <= tol ) then
        iedg = 5
      else
        iedg = 2
      end if
      exit
    else if ( beta <= tol ) then
      if ( gamma <= tol ) then
        iedg = 4
      else
        iedg = 3
      end if
      exit
    else
      iedg = 1
      exit
    end if
!
!  If we fell through, then at least one barycentric coordinate was negative
!  for a side of the current triangle, and that side has a neighboring
!  triangle I.  Let's go there.
!
    itri = i

  end do

  return
end

!-------------------------------------------------------------------------------

subroutine vbedg ( x, y, vcl, til, tnbr, ltri, ledg, rtri, redg )
!
!*******************************************************************************
```

```
!
!! VBEDG determines visible boundary edges of a 2D triangulation.
!
!
!  Purpose:
!
!    Determine boundary edges of 2-D triangulation which are
!    visible from point (X,Y) outside convex hull.
!
!  Modified:
!
!    14 July 2001
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, double precision X, Y, the coordinates of a 2-D point outside
!    the convex hull.
!
!    Input, double precision VCL(1:2,1:*), the coordinates of 2-D vertices.
!
!    Input, integer TIL(1:3,1:*), the triangle incidence list.
!
!    Input, integer TNBR(1:3,1:*), the triangle neighbor list; negative
!    values are used for links of counter clockwise linked list of boundary
!    edges; LINK = -(3*I + J-1) where I, J = triangle, edge index.
!
!    Input/output, integer LTRI, LEDG.  On input, if LTRI /= 0 then they
!    are assumed to be as defined below and are not changed, else they are
!    updated.  On output, LTRI is the index of the boundary triangle to the
!    left of leftmost boundary triangle visible from (X,Y), and LEDG is the
!    boundary edge of triangle LTRI to left of leftmost
!    boundary edge visible from (X,Y).  1 <= LEDG <= 3.
!
!    Input/output, integer RTRI, on input, the index of boundary triangle
!    to begin search at.  On output, the index of rightmost boundary triangle
!    visible from (X,Y).
!
!    Input/output, integer REDG.  On input, the edge of triangle RTRI that
!    is visible from (X,Y).  On output, REDG has been updated so that this
!    is still true. 1 <= REDG <= 3.
!
  integer a
  integer b
  integer e
  integer i_wrap
  integer l
  logical ldone
  integer ledg
  integer lr
  integer lrline
  integer ltri
  integer redg
  integer rtri
  integer t
  integer til(3,*)
  integer tnbr(3,*)
```

```
  double precision vcl(2,*)
  double precision x
  double precision y
!
!  Find rightmost visible boundary edge using links, then possibly
!  leftmost visible boundary edge using triangle neighbor information.
!
  if ( ltri == 0 ) then
    ldone = .false.
    ltri = rtri
    ledg = redg
  else
    ldone = .true.
  end if

10 continue

  l = -tnbr(redg,rtri)
  t = l / 3
  e = mod ( l, 3 ) + 1
  a = til(e,t)

  if ( e <= 2 ) then
    b = til(e+1,t)
  else
    b = til(1,t)
  end if

  lr = lrline ( x, y, vcl(1,a), vcl(2,a), vcl(1,b), vcl(2,b), 0.0D+00 )

  if ( lr > 0 ) then
    rtri = t
    redg = e
    go to 10
  end if

  if ( ldone ) then
    return
  end if

  t = ltri
  e = ledg

  do

    b = til(e,t)
    e = i_wrap ( e-1, 1, 3 )

    do while ( tnbr(e,t) > 0 )

      t = tnbr(e,t)

      if ( til(1,t) == b ) then
        e = 3
      else if ( til(2,t) == b ) then
        e = 1
      else
        e = 2
      end if

    end do

    a = til(e,t)
```

```
    lr = lrline ( x, y, vcl(1,a), vcl(2,a), vcl(1,b), vcl(2,b), 0.0D+00 )

    if ( lr <= 0 ) then
      exit
    end if

  end do

  ltri = t
  ledg = e

  return
end

!-------------------------------------------------------------------------------

subroutine swapec ( i, top, maxst, btri, bedg, vcl, til, tnbr, stack, ierror )
!
!*******************************************************************************
!
!! SWAPEC swaps diagonal edges until all triangles are Delaunay.
!
!
!  Discussion:
!
!    The routine swaps diagonal edges in a 2-D triangulation, based on
!    the empty circumcircle criterion, until all triangles are Delaunay,
!    given that I is the index of the new vertex added to triangulation.
!
!  Modified:
!
!    19 February 2001
!
!  Author:
!
!    Barry Joe,
!    Department of Computing Science,
!    University of Alberta,
!    Edmonton, Alberta, Canada  T6G 2H1
!
!  Parameters:
!
!    Input, integer I, the index in VCL of the new vertex.
!
!    Input/output, integer TOP, the index of the top of the stack.
!    On output, TOP is zero.
!
!    Input, integer MAXST, the maximum size available for the STACK array.
!
!    Input/output, integer BTRI, BEDG; on input, if positive, are the
!    triangle and edge indices of a boundary edge whose updated indices
!    must be recorded.  On output, these may be updated because of swaps.
!
!    Input, double precision VCL(2,*), the coordinates of the vertices.
!
!    Input/output, integer TIL(3,*), the triangle incidence list.  May be updated
!    on output because of swaps.
!
!    Input/output, integer TNBR(3,*), the triangle neighbor list; negative
!    values are used for links of the counter-clockwise linked list of boundary
!    edges; May be updated on output because of swaps.
!
!      LINK = -(3*I + J-1) where I, J = triangle, edge index.
```

```
!
!    Workspace, integer STACK(1:MAXST); on input, entries 1 through TOP
!    contain the indices of initial triangles (involving vertex I)
!    put in stack; the edges opposite I should be in interior;  entries
!    TOP+1 through MAXST are used as a stack.
!
!    Output, integer IERROR is set to 8 for abnormal return.
!
  integer maxst
!
  integer a
  integer b
  integer bedg
  integer btri
  integer c
  integer diaedg
  integer e
  integer ee
  integer em1
  integer ep1
  integer f
  integer fm1
  integer fp1
  integer i
  integer ierror
  integer l
  integer r
  integer s
  integer stack(maxst)
  integer swap
  integer t
  integer til(3,*)
  integer tnbr(3,*)
  integer top
  integer tt
  integer u
  double precision vcl(2,*)
  double precision x
  double precision y
!
!  Determine whether triangles in stack are Delaunay, and swap
!  diagonal edge of convex quadrilateral if not.
!
  ierror = 0
  x = vcl(1,i)
  y = vcl(2,i)

  do

    if ( top <= 0 ) then
      exit
    end if

    t = stack(top)
    top = top - 1

    if ( til(1,t) == i ) then
      e = 2
      b = til(3,t)
    else if ( til(2,t) == i ) then
      e = 3
      b = til(1,t)
    else
```

```
  e = 1
  b = til(2,t)
end if

a = til(e,t)
u = tnbr(e,t)

if ( tnbr(1,u) == t ) then
  f = 1
  c = til(3,u)
else if ( tnbr(2,u) == t ) then
  f = 2
  c = til(1,u)
else
  f = 3
  c = til(2,u)
end if

swap = diaedg ( x, y, vcl(1,a), vcl(2,a), vcl(1,c), vcl(2,c), &
  vcl(1,b), vcl(2,b) )

if ( swap == 1 ) then

  em1 = i_wrap ( e - 1, 1, 3 )
  ep1 = i_wrap ( e + 1, 1, 3 )
  fm1 = i_wrap ( f - 1, 1, 3 )
  fp1 = i_wrap ( f + 1, 1, 3 )

  til(ep1,t) = c
  til(fp1,u) = i
  r = tnbr(ep1,t)
  s = tnbr(fp1,u)
  tnbr(ep1,t) = u
  tnbr(fp1,u) = t
  tnbr(e,t) = s
  tnbr(f,u) = r

  if ( tnbr(fm1,u) > 0 ) then
    top = top + 1
    stack(top) = u
  end if

  if ( s > 0 ) then

    if ( tnbr(1,s) == u ) then
      tnbr(1,s) = t
    else if ( tnbr(2,s) == u ) then
      tnbr(2,s) = t
    else
      tnbr(3,s) = t
    end if

    top = top + 1

    if ( top > maxst ) then
      ierror = 8
      return
    end if

    stack(top) = t

  else
```

```
    if ( u == btri .and. fp1 == bedg ) then
      btri = t
      bedg = e
    end if

    l = - ( 3 * t + e - 1 )
    tt = t
    ee = em1

    do while ( tnbr(ee,tt) > 0 )

      tt = tnbr(ee,tt)

      if ( til(1,tt) == a ) then
        ee = 3
      else if ( til(2,tt) == a ) then
        ee = 1
      else
        ee = 2
      end if

    end do

    tnbr(ee,tt) = l

  end if

  if ( r > 0 ) then

    if ( tnbr(1,r) == t ) then
      tnbr(1,r) = u
    else if ( tnbr(2,r) == t ) then
      tnbr(2,r) = u
    else
      tnbr(3,r) = u
    end if

  else

    if ( t == btri .and. ep1 == bedg ) then
      btri = u
      bedg = f
    end if

    l = - ( 3 * u + f - 1 )
    tt = u
    ee = fm1

    do while ( tnbr(ee,tt) > 0 )

      tt = tnbr(ee,tt)

      if ( til(1,tt) == b ) then
        ee = 3
      else if ( til(2,tt) == b ) then
        ee = 1
      else
        ee = 2
      end if

    end do

    tnbr(ee,tt) = l
```

```
      end if

    end if

  end do

  return
end

!----------------------------------------------------------------------------------

function i_wrap ( ival, ilo, ihi )
!
!*********************************************************************************
!
!! I_WRAP forces an integer to lie between given limits by wrapping.
!
!
!  Example:
!
!    ILO = 4, IHI = 8
!
!    I  I_WRAP
!
!    -2     8
!    -1     4
!     0     5
!     1     6
!     2     7
!     3     8
!     4     4
!     5     5
!     6     6
!     7     7
!     8     8
!     9     4
!    10     5
!    11     6
!    12     7
!    13     8
!    14     4
!
!  Modified:
!
!    15 July 2000
!
!  Author:
!
!    John Burkardt
!
!  Parameters:
!
!    Input, integer IVAL, an integer value.
!
!    Input, integer ILO, IHI, the desired bounds for the integer value.
!
!    Output, integer I_WRAP, a "wrapped" version of IVAL.
!
  integer i_modp
  integer i_wrap
  integer ihi
  integer ilo
```

```
  integer ival
  integer wide
!
  wide = ihi + 1 - ilo

  if ( wide == 0 ) then
    i_wrap = ilo
  else
    i_wrap = ilo + i_modp ( ival-ilo, wide )
  end if

  return
end

!-------------------------------------------------------------------------------

function i_modp ( i, j )
!
!*******************************************************************************
!
!! I_MODP returns the nonnegative remainder of integer division.
!
!
!  Formula:
!
!    If
!      NREM = I_MODP ( I, J )
!      NMULT = ( I - NREM ) / J
!    then
!      I = J * NMULT + NREM
!    where NREM is always nonnegative.
!
!  Comments:
!
!    The MOD function computes a result with the same sign as the
!    quantity being divided.  Thus, suppose you had an angle A,
!    and you wanted to ensure that it was between 0 and 360.
!    Then mod(A,360) would do, if A was positive, but if A
!    was negative, your result would be between -360 and 0.
!
!    On the other hand, I_MODP(A,360) is between 0 and 360, always.
!
!  Examples:
!
!        I     J     MOD  I_MODP    Factorization
!
!      107    50       7       7    107 =  2 *  50 + 7
!      107   -50       7       7    107 = -2 * -50 + 7
!     -107    50      -7      43   -107 = -3 *  50 + 43
!     -107   -50      -7      43   -107 =  3 * -50 + 43
!
!  Modified:
!
!    02 March 1999
!
!  Author:
!
!    John Burkardt
!
!  Parameters:
!
!    Input, integer I, the number to be divided.
!
```

```
!    Input, integer J, the number that divides I.
!
!    Output, integer I_MODP, the nonnegative remainder when I is
!    divided by J.
!
  integer i
  integer i_modp
  integer j
!
!!  if ( j == 0 ) then
!!    write ( *, '(a)' ) ' '
!!    write ( *, '(a)' ) 'I_MODP - Fatal error!'
!!    write ( *, '(a,i6)' ) '  I_MODP ( I, J ) called with J = ', j
!!    stop
!!  end if

  i_modp = mod ( i, j )

  if ( i_modp < 0 ) then
    i_modp = i_modp + abs ( j )
  end if

  return
end

!-------------------------------------------------------------------------------
!
```

Appendix J. Listing of Input File for the Hypothetical LiDAR Example

The hypothetical LiDAR data (Figure J.1) is described in section 3 and in Berenbrock (2010) and is used here to demonstrate the use of the LiDAR and Bathymetric genetic algorithm (GA) program. The input file must be a space delimited text file with one data point or coordinate per line. The coordinate is composed of three variables: x, y, and z and given in that respective order. The x coordinate represents the easting or longitude of a data point, y represents the northing or latitude of a data point, and z represents the height or elevation of a point. The values of x, y, and z variables are given in centimeters, are integers numbers (no decimals), and cannot be less than zero. Usually, LiDAR data are given in units of meters, but the LiDAR and Bathymetric GA program requires that the user converts the data to units of centimeters because working with integer numbers is easer, faster, and requires less storage space than for computer programs that use floating-point numbers. See Appendix H for setting up the input files for the program.

The original LiDAR dataset for the hypothetical LiDAR example is listed on the left-side of the Table J.1. These units are in meters. Data that is inputted into the LiDAR and Bathymetry GA program is listed on the right-side of Table J.1. The inputted data are illustrated in Figure J.1.

The following shows how to calculate inputted data from the original data. These calculations can easily be performed by using a spreadsheet program. First determine the minimum values for northing, easting, and elevation from the original data. For the hypothetical LiDAR example, the minimums are 3144.0, 12280.0, and 17.912, for northing, easting, and elevation, respectively. Note that northing represents y values, easting represents x values, and elevation represents z values. To calculate x values, subtract each easting value from the minimum easting value (12760.0) and multiply by 100 and round to the nearest whole number (integer). For example, the calculation for determining the first x value in Table J-1 is (12760.0 – 12280.0) × 100 = 48000. To calculate y values, subtract each northing value from the minimum northing value (3144.0) and multiply by 100 and round to the nearest whole number (integer). For example, the calculation for determining the first y value in Table J.1 is (3144.0 – 3144.0) × 100 = 0. To calculate z values, subtract each elevation value from the minimum elevation value (17.912) and multiply by 100 and round to the nearest whole number (integer). For example, the calculation for determining the first z value in Table J.1 is
(23.890 – 17.912) × 100 = 598.

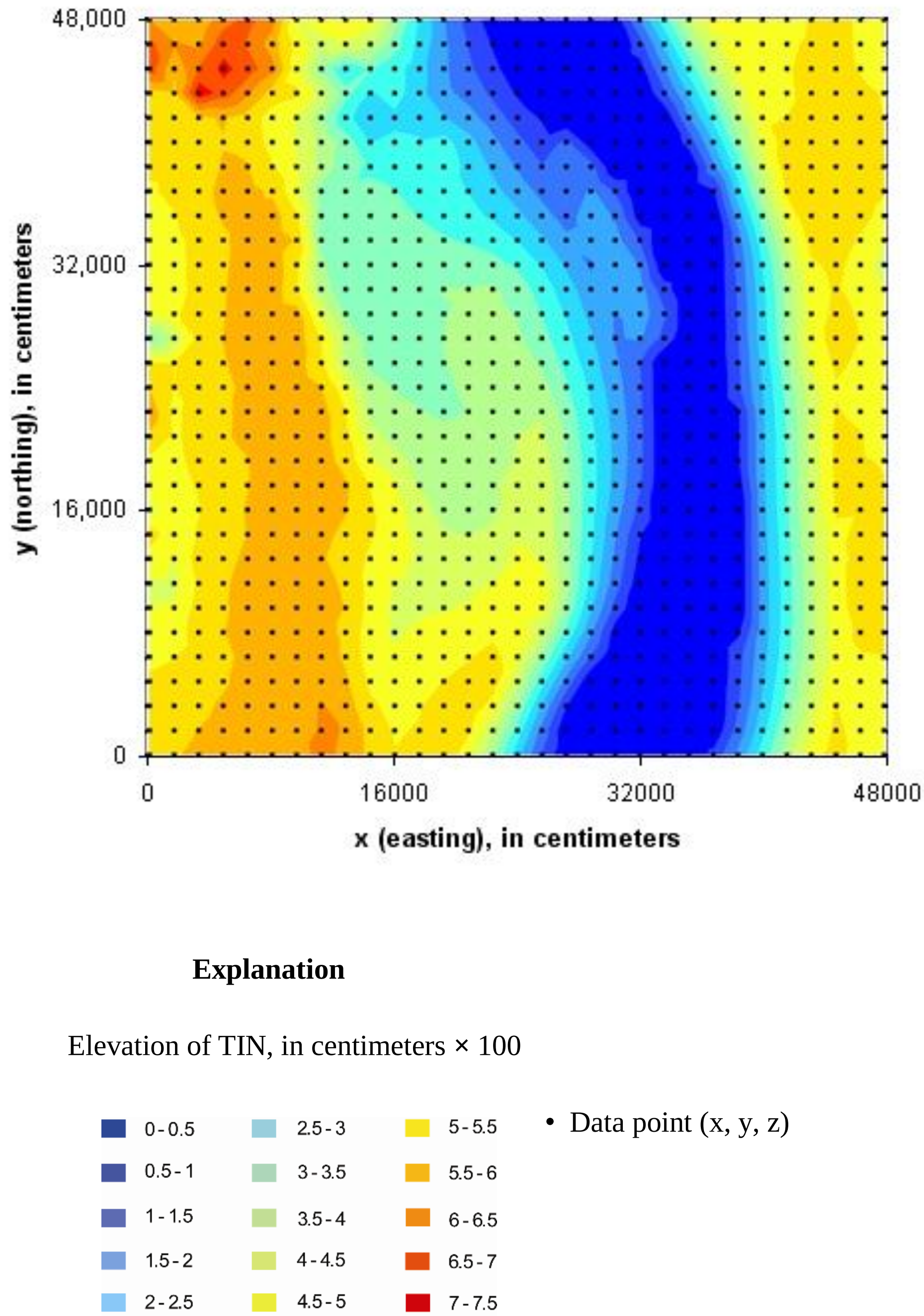


Figure J.1. Inputted LiDAR data for the hypothetical LiDAR data.

Table J.1. Listing of hypothetical and inputted LiDAR data. The original LiDAR data (Northing, Easting, and Elevation) are listed on the left-side (units in meters) and inputted data (x, y, and z) are listed on the right-side (units in centimeters) minus the lowest respective value.

Original LiDAR data			Inputted LiDAR data		
Northing	Easting	Elevation	x	y	z
3144.0	12760.0	23.890	48000	0	598
3160.0	12760.0	23.400	48000	1600	549
3176.0	12760.0	23.510	48000	3200	560
3192.0	12760.0	24.200	48000	4800	629
3208.0	12760.0	24.580	48000	6400	667
3224.0	12760.0	24.135	48000	8000	622
3240.0	12760.0	22.382	48000	9600	447
3256.0	12760.0	22.935	48000	11200	502
3272.0	12760.0	22.798	48000	12800	489
3288.0	12760.0	22.644	48000	14400	473
3304.0	12760.0	21.524	48000	16000	361
3320.0	12760.0	20.376	48000	17600	246
3336.0	12760.0	19.455	48000	19200	154
3352.0	12760.0	18.740	48000	20800	83
3368.0	12760.0	18.122	48000	22400	21
3384.0	12760.0	17.912	48000	24000	0
3400.0	12760.0	17.912	48000	25600	0
3416.0	12760.0	17.913	48000	27200	0
3432.0	12760.0	17.913	48000	28800	0
3448.0	12760.0	18.182	48000	30400	27
3464.0	12760.0	19.429	48000	32000	152
3480.0	12760.0	20.720	48000	33600	281
3496.0	12760.0	22.009	48000	35200	410
3512.0	12760.0	22.678	48000	36800	477
3528.0	12760.0	22.573	48000	38400	466
3544.0	12760.0	22.503	48000	40000	459
3560.0	12760.0	22.675	48000	41600	476
3576.0	12760.0	22.962	48000	43200	505

3592.0	12760.0	23.000	48000	44800	509
3608.0	12760.0	22.868	48000	46400	496
3624.0	12760.0	22.538	48000	48000	463
3144.0	12744.0	24.560	46400	0	665
3160.0	12744.0	23.978	46400	1600	607
3176.0	12744.0	23.802	46400	3200	589
3192.0	12744.0	24.722	46400	4800	681
3208.0	12744.0	24.358	46400	6400	645
3224.0	12744.0	23.685	46400	8000	577
3240.0	12744.0	22.757	46400	9600	485
3256.0	12744.0	22.115	46400	11200	420
3272.0	12744.0	22.337	46400	12800	443
3288.0	12744.0	21.938	46400	14400	403
3304.0	12744.0	21.111	46400	16000	320
3320.0	12744.0	20.254	46400	17600	234
3336.0	12744.0	19.502	46400	19200	159
3352.0	12744.0	18.884	46400	20800	97
3368.0	12744.0	18.368	46400	22400	46
3384.0	12744.0	17.913	46400	24000	0
3400.0	12744.0	17.913	46400	25600	0
3416.0	12744.0	17.914	46400	27200	0
3432.0	12744.0	17.914	46400	28800	0
3448.0	12744.0	17.915	46400	30400	0
3464.0	12744.0	18.702	46400	32000	79
3480.0	12744.0	20.018	46400	33600	211
3496.0	12744.0	21.293	46400	35200	338
3512.0	12744.0	22.295	46400	36800	438
3528.0	12744.0	22.717	46400	38400	481
3544.0	12744.0	22.577	46400	40000	467
3560.0	12744.0	22.825	46400	41600	491
3576.0	12744.0	23.000	46400	43200	509
3592.0	12744.0	23.010	46400	44800	510
3608.0	12744.0	22.870	46400	46400	496
3624.0	12744.0	22.638	46400	48000	473
3144.0	12728.0	24.725	44800	0	681

3160.0	12728.0	23.440	44800	1600	553
3176.0	12728.0	24.460	44800	3200	655
3192.0	12728.0	25.050	44800	4800	714
3208.0	12728.0	24.680	44800	6400	677
3224.0	12728.0	24.077	44800	8000	617
3240.0	12728.0	22.610	44800	9600	470
3256.0	12728.0	21.340	44800	11200	343
3272.0	12728.0	20.452	44800	12800	254
3288.0	12728.0	20.967	44800	14400	306
3304.0	12728.0	20.632	44800	16000	272
3320.0	12728.0	20.091	44800	17600	218
3336.0	12728.0	19.509	44800	19200	160
3352.0	12728.0	19.018	44800	20800	111
3368.0	12728.0	18.547	44800	22400	64
3384.0	12728.0	18.093	44800	24000	18
3400.0	12728.0	17.914	44800	25600	0
3416.0	12728.0	17.915	44800	27200	0
3432.0	12728.0	17.915	44800	28800	0
3448.0	12728.0	17.916	44800	30400	0
3464.0	12728.0	18.227	44800	32000	32
3480.0	12728.0	19.409	44800	33600	150
3496.0	12728.0	20.628	44800	35200	272
3512.0	12728.0	21.766	44800	36800	385
3528.0	12728.0	22.592	44800	38400	468
3544.0	12728.0	22.593	44800	40000	468
3560.0	12728.0	22.870	44800	41600	496
3576.0	12728.0	23.042	44800	43200	513
3592.0	12728.0	23.000	44800	44800	509
3608.0	12728.0	22.897	44800	46400	499
3624.0	12728.0	22.788	44800	48000	488
3144.0	12712.0	23.190	43200	0	528
3160.0	12712.0	23.323	43200	1600	541
3176.0	12712.0	25.260	43200	3200	735
3192.0	12712.0	24.683	43200	4800	677
3208.0	12712.0	23.987	43200	6400	608

3224.0	12712.0	23.200	43200	8000	529
3240.0	12712.0	23.040	43200	9600	513
3256.0	12712.0	22.173	43200	11200	426
3272.0	12712.0	21.093	43200	12800	318
3288.0	12712.0	20.360	43200	14400	245
3304.0	12712.0	20.657	43200	16000	275
3320.0	12712.0	20.047	43200	17600	214
3336.0	12712.0	19.568	43200	19200	166
3352.0	12712.0	19.229	43200	20800	132
3368.0	12712.0	18.804	43200	22400	89
3384.0	12712.0	18.347	43200	24000	44
3400.0	12712.0	17.959	43200	25600	5
3416.0	12712.0	17.915	43200	27200	0
3432.0	12712.0	17.916	43200	28800	0
3448.0	12712.0	17.917	43200	30400	1
3464.0	12712.0	17.917	43200	32000	1
3480.0	12712.0	18.834	43200	33600	92
3496.0	12712.0	19.990	43200	35200	208
3512.0	12712.0	21.175	43200	36800	326
3528.0	12712.0	22.242	43200	38400	433
3544.0	12712.0	22.772	43200	40000	486
3560.0	12712.0	22.850	43200	41600	494
3576.0	12712.0	23.048	43200	43200	514
3592.0	12712.0	23.022	43200	44800	511
3608.0	12712.0	22.975	43200	46400	506
3624.0	12712.0	22.892	43200	48000	498
3144.0	12696.0	23.195	41600	0	528
3160.0	12696.0	23.247	41600	1600	534
3176.0	12696.0	23.340	41600	3200	543
3192.0	12696.0	23.458	41600	4800	555
3208.0	12696.0	23.190	41600	6400	528
3224.0	12696.0	22.670	41600	8000	476
3240.0	12696.0	22.405	41600	9600	449
3256.0	12696.0	21.550	41600	11200	364
3272.0	12696.0	20.487	41600	12800	258

3288.0	12696.0	20.245	41600	14400	233
3304.0	12696.0	20.308	41600	16000	240
3320.0	12696.0	20.233	41600	17600	232
3336.0	12696.0	19.944	41600	19200	203
3352.0	12696.0	19.624	41600	20800	171
3368.0	12696.0	19.162	41600	22400	125
3384.0	12696.0	18.696	41600	24000	78
3400.0	12696.0	18.363	41600	25600	45
3416.0	12696.0	17.945	41600	27200	3
3432.0	12696.0	17.917	41600	28800	1
3448.0	12696.0	17.917	41600	30400	1
3464.0	12696.0	17.918	41600	32000	1
3480.0	12696.0	18.040	41600	33600	13
3496.0	12696.0	19.391	41600	35200	148
3512.0	12696.0	20.574	41600	36800	266
3528.0	12696.0	21.778	41600	38400	387
3544.0	12696.0	22.798	41600	40000	489
3560.0	12696.0	22.923	41600	41600	501
3576.0	12696.0	23.035	41600	43200	512
3592.0	12696.0	23.028	41600	44800	512
3608.0	12696.0	23.005	41600	46400	509
3624.0	12696.0	22.915	41600	48000	500
3144.0	12680.0	23.058	40000	0	515
3160.0	12680.0	23.070	40000	1600	516
3176.0	12680.0	23.195	40000	3200	528
3192.0	12680.0	23.392	40000	4800	548
3208.0	12680.0	23.320	40000	6400	541
3224.0	12680.0	22.720	40000	8000	481
3240.0	12680.0	22.382	40000	9600	447
3256.0	12680.0	21.775	40000	11200	386
3272.0	12680.0	21.202	40000	12800	329
3288.0	12680.0	20.407	40000	14400	250
3304.0	12680.0	20.510	40000	16000	260
3320.0	12680.0	20.462	40000	17600	255
3336.0	12680.0	20.610	40000	19200	270

3352.0	12680.0	20.071	40000	20800	216
3368.0	12680.0	19.531	40000	22400	162
3384.0	12680.0	19.044	40000	24000	113
3400.0	12680.0	18.611	40000	25600	70
3416.0	12680.0	19.023	40000	27200	111
3432.0	12680.0	18.104	40000	28800	19
3448.0	12680.0	17.918	40000	30400	1
3464.0	12680.0	17.919	40000	32000	1
3480.0	12680.0	17.919	40000	33600	1
3496.0	12680.0	18.628	40000	35200	72
3512.0	12680.0	19.947	40000	36800	204
3528.0	12680.0	21.271	40000	38400	336
3544.0	12680.0	22.527	40000	40000	462
3560.0	12680.0	23.160	40000	41600	525
3576.0	12680.0	23.077	40000	43200	517
3592.0	12680.0	23.105	40000	44800	519
3608.0	12680.0	23.073	40000	46400	516
3624.0	12680.0	23.050	40000	48000	514
3144.0	12664.0	22.975	38400	0	506
3160.0	12664.0	23.007	38400	1600	510
3176.0	12664.0	23.205	38400	3200	529
3192.0	12664.0	23.455	38400	4800	554
3208.0	12664.0	23.438	38400	6400	553
3224.0	12664.0	22.952	38400	8000	504
3240.0	12664.0	22.472	38400	9600	456
3256.0	12664.0	21.790	38400	11200	388
3272.0	12664.0	21.075	38400	12800	316
3288.0	12664.0	20.840	38400	14400	293
3304.0	12664.0	20.785	38400	16000	287
3320.0	12664.0	20.615	38400	17600	270
3336.0	12664.0	20.694	38400	19200	278
3352.0	12664.0	20.287	38400	20800	238
3368.0	12664.0	19.822	38400	22400	191
3384.0	12664.0	19.367	38400	24000	146
3400.0	12664.0	18.941	38400	25600	103

3416.0	12664.0	19.409	38400	27200	150
3432.0	12664.0	19.092	38400	28800	118
3448.0	12664.0	17.947	38400	30400	4
3464.0	12664.0	17.919	38400	32000	1
3480.0	12664.0	17.920	38400	33600	1
3496.0	12664.0	17.920	38400	35200	1
3512.0	12664.0	19.304	38400	36800	139
3528.0	12664.0	20.728	38400	38400	282
3544.0	12664.0	22.144	38400	40000	423
3560.0	12664.0	23.217	38400	41600	531
3576.0	12664.0	23.210	38400	43200	530
3592.0	12664.0	23.228	38400	44800	532
3608.0	12664.0	23.243	38400	46400	533
3624.0	12664.0	23.470	38400	48000	556
3144.0	12648.0	22.865	36800	0	495
3160.0	12648.0	23.010	36800	1600	510
3176.0	12648.0	23.250	36800	3200	534
3192.0	12648.0	23.532	36800	4800	562
3208.0	12648.0	23.573	36800	6400	566
3224.0	12648.0	23.330	36800	8000	542
3240.0	12648.0	22.490	36800	9600	458
3256.0	12648.0	21.225	36800	11200	331
3272.0	12648.0	20.995	36800	12800	308
3288.0	12648.0	21.017	36800	14400	311
3304.0	12648.0	20.927	36800	16000	302
3320.0	12648.0	20.745	36800	17600	283
3336.0	12648.0	20.650	36800	19200	274
3352.0	12648.0	20.393	36800	20800	248
3368.0	12648.0	20.067	36800	22400	216
3384.0	12648.0	19.678	36800	24000	177
3400.0	12648.0	19.271	36800	25600	136
3416.0	12648.0	19.031	36800	27200	112
3432.0	12648.0	19.503	36800	28800	159
3448.0	12648.0	18.854	36800	30400	94
3464.0	12648.0	17.920	36800	32000	1

3480.0	12648.0	17.921	36800	33600	1
3496.0	12648.0	17.921	36800	35200	1
3512.0	12648.0	18.375	36800	36800	46
3528.0	12648.0	20.194	36800	38400	228
3544.0	12648.0	21.704	36800	40000	379
3560.0	12648.0	23.047	36800	41600	514
3576.0	12648.0	23.215	36800	43200	530
3592.0	12648.0	23.230	36800	44800	532
3608.0	12648.0	23.167	36800	46400	526
3624.0	12648.0	22.962	36800	48000	505
3144.0	12632.0	22.765	35200	0	485
3160.0	12632.0	22.897	35200	1600	499
3176.0	12632.0	23.210	35200	3200	530
3192.0	12632.0	23.460	35200	4800	555
3208.0	12632.0	23.618	35200	6400	571
3224.0	12632.0	23.653	35200	8000	574
3240.0	12632.0	22.962	35200	9600	505
3256.0	12632.0	21.205	35200	11200	329
3272.0	12632.0	21.015	35200	12800	310
3288.0	12632.0	21.128	35200	14400	322
3304.0	12632.0	21.077	35200	16000	317
3320.0	12632.0	20.882	35200	17600	297
3336.0	12632.0	20.694	35200	19200	278
3352.0	12632.0	20.560	35200	20800	265
3368.0	12632.0	20.345	35200	22400	243
3384.0	12632.0	20.015	35200	24000	210
3400.0	12632.0	19.616	35200	25600	170
3416.0	12632.0	19.198	35200	27200	129
3432.0	12632.0	19.742	35200	28800	183
3448.0	12632.0	19.348	35200	30400	144
3464.0	12632.0	18.149	35200	32000	24
3480.0	12632.0	17.922	35200	33600	1
3496.0	12632.0	17.922	35200	35200	1
3512.0	12632.0	17.922	35200	36800	1
3528.0	12632.0	19.651	35200	38400	174

3544.0	12632.0	21.178	35200	40000	327
3560.0	12632.0	22.565	35200	41600	465
3576.0	12632.0	23.418	35200	43200	551
3592.0	12632.0	23.235	35200	44800	532
3608.0	12632.0	23.073	35200	46400	516
3624.0	12632.0	22.747	35200	48000	484
3144.0	12616.0	22.657	33600	0	475
3160.0	12616.0	22.800	33600	1600	489
3176.0	12616.0	23.045	33600	3200	513
3192.0	12616.0	23.347	33600	4800	544
3208.0	12616.0	23.647	33600	6400	574
3224.0	12616.0	23.692	33600	8000	578
3240.0	12616.0	23.362	33600	9600	545
3256.0	12616.0	21.253	33600	11200	334
3272.0	12616.0	21.050	33600	12800	314
3288.0	12616.0	21.215	33600	14400	330
3304.0	12616.0	21.280	33600	16000	337
3320.0	12616.0	21.142	33600	17600	323
3336.0	12616.0	20.975	33600	19200	306
3352.0	12616.0	20.860	33600	20800	295
3368.0	12616.0	20.699	33600	22400	279
3384.0	12616.0	20.401	33600	24000	249
3400.0	12616.0	19.987	33600	25600	208
3416.0	12616.0	19.526	33600	27200	161
3432.0	12616.0	19.725	33600	28800	181
3448.0	12616.0	19.604	33600	30400	169
3464.0	12616.0	18.849	33600	32000	94
3480.0	12616.0	17.923	33600	33600	1
3496.0	12616.0	17.923	33600	35200	1
3512.0	12616.0	17.923	33600	36800	1
3528.0	12616.0	19.215	33600	38400	130
3544.0	12616.0	20.718	33600	40000	281
3560.0	12616.0	22.094	33600	41600	418
3576.0	12616.0	23.131	33600	43200	522
3592.0	12616.0	23.235	33600	44800	532

3608.0	12616.0	22.847	33600	46400	494
3624.0	12616.0	22.405	33600	48000	449
3144.0	12600.0	22.653	32000	0	474
3160.0	12600.0	22.772	32000	1600	486
3176.0	12600.0	23.050	32000	3200	514
3192.0	12600.0	23.362	32000	4800	545
3208.0	12600.0	23.587	32000	6400	568
3224.0	12600.0	23.710	32000	8000	580
3240.0	12600.0	23.038	32000	9600	513
3256.0	12600.0	21.522	32000	11200	361
3272.0	12600.0	20.925	32000	12800	301
3288.0	12600.0	21.210	32000	14400	330
3304.0	12600.0	21.330	32000	16000	342
3320.0	12600.0	21.308	32000	17600	340
3336.0	12600.0	21.288	32000	19200	338
3352.0	12600.0	21.166	32000	20800	325
3368.0	12600.0	21.106	32000	22400	319
3384.0	12600.0	20.828	32000	24000	292
3400.0	12600.0	20.363	32000	25600	245
3416.0	12600.0	19.851	32000	27200	194
3432.0	12600.0	19.337	32000	28800	143
3448.0	12600.0	19.828	32000	30400	192
3464.0	12600.0	19.362	32000	32000	145
3480.0	12600.0	18.007	32000	33600	10
3496.0	12600.0	17.924	32000	35200	1
3512.0	12600.0	17.925	32000	36800	1
3528.0	12600.0	19.049	32000	38400	114
3544.0	12600.0	20.434	32000	40000	252
3560.0	12600.0	21.753	32000	41600	384
3576.0	12600.0	22.825	32000	43200	491
3592.0	12600.0	23.067	32000	44800	516
3608.0	12600.0	22.642	32000	46400	473
3624.0	12600.0	22.240	32000	48000	433
3144.0	12584.0	22.712	30400	0	480
3160.0	12584.0	22.798	30400	1600	489

3176.0	12584.0	23.015	30400	3200	510
3192.0	12584.0	23.385	30400	4800	547
3208.0	12584.0	23.595	30400	6400	568
3224.0	12584.0	23.718	30400	8000	581
3240.0	12584.0	22.978	30400	9600	507
3256.0	12584.0	21.663	30400	11200	375
3272.0	12584.0	21.063	30400	12800	315
3288.0	12584.0	21.080	30400	14400	317
3304.0	12584.0	21.202	30400	16000	329
3320.0	12584.0	21.343	30400	17600	343
3336.0	12584.0	21.420	30400	19200	351
3352.0	12584.0	21.448	30400	20800	354
3368.0	12584.0	21.495	30400	22400	358
3384.0	12584.0	21.225	30400	24000	331
3400.0	12584.0	20.689	30400	25600	278
3416.0	12584.0	20.134	30400	27200	222
3432.0	12584.0	19.586	30400	28800	167
3448.0	12584.0	19.910	30400	30400	200
3464.0	12584.0	19.585	30400	32000	167
3480.0	12584.0	18.616	30400	33600	70
3496.0	12584.0	17.926	30400	35200	1
3512.0	12584.0	17.926	30400	36800	1
3528.0	12584.0	18.986	30400	38400	107
3544.0	12584.0	20.293	30400	40000	238
3560.0	12584.0	21.574	30400	41600	366
3576.0	12584.0	22.671	30400	43200	476
3592.0	12584.0	22.983	30400	44800	507
3608.0	12584.0	22.675	30400	46400	476
3624.0	12584.0	22.417	30400	48000	451
3144.0	12568.0	22.670	28800	0	476
3160.0	12568.0	22.850	28800	1600	494
3176.0	12568.0	23.100	28800	3200	519
3192.0	12568.0	23.390	28800	4800	548
3208.0	12568.0	23.635	28800	6400	572
3224.0	12568.0	23.755	28800	8000	584

3240.0	12568.0	23.600	28800	9600	569
3256.0	12568.0	22.298	28800	11200	439
3272.0	12568.0	21.372	28800	12800	346
3288.0	12568.0	21.095	28800	14400	318
3304.0	12568.0	21.093	28800	16000	318
3320.0	12568.0	21.270	28800	17600	336
3336.0	12568.0	21.392	28800	19200	348
3352.0	12568.0	21.510	28800	20800	360
3368.0	12568.0	21.565	28800	22400	365
3384.0	12568.0	21.379	28800	24000	347
3400.0	12568.0	20.906	28800	25600	299
3416.0	12568.0	20.358	28800	27200	245
3432.0	12568.0	19.790	28800	28800	188
3448.0	12568.0	19.385	28800	30400	147
3464.0	12568.0	19.694	28800	32000	178
3480.0	12568.0	18.985	28800	33600	107
3496.0	12568.0	17.928	28800	35200	2
3512.0	12568.0	17.928	28800	36800	2
3528.0	12568.0	18.946	28800	38400	103
3544.0	12568.0	20.225	28800	40000	231
3560.0	12568.0	21.508	28800	41600	360
3576.0	12568.0	22.658	28800	43200	475
3592.0	12568.0	23.040	28800	44800	513
3608.0	12568.0	22.862	28800	46400	495
3624.0	12568.0	22.655	28800	48000	474
3144.0	12552.0	21.135	27200	0	322
3160.0	12552.0	21.983	27200	1600	407
3176.0	12552.0	23.097	27200	3200	519
3192.0	12552.0	23.420	27200	4800	551
3208.0	12552.0	23.705	27200	6400	579
3224.0	12552.0	23.737	27200	8000	583
3240.0	12552.0	23.680	27200	9600	577
3256.0	12552.0	22.860	27200	11200	495
3272.0	12552.0	21.810	27200	12800	390
3288.0	12552.0	21.215	27200	14400	330

3304.0	12552.0	21.100	27200	16000	319
3320.0	12552.0	21.210	27200	17600	330
3336.0	12552.0	21.427	27200	19200	352
3352.0	12552.0	21.580	27200	20800	367
3368.0	12552.0	21.612	27200	22400	370
3384.0	12552.0	21.498	27200	24000	359
3400.0	12552.0	21.095	27200	25600	318
3416.0	12552.0	20.556	27200	27200	264
3432.0	12552.0	19.964	27200	28800	205
3448.0	12552.0	19.378	27200	30400	147
3464.0	12552.0	19.482	27200	32000	157
3480.0	12552.0	18.745	27200	33600	83
3496.0	12552.0	17.929	27200	35200	2
3512.0	12552.0	17.929	27200	36800	2
3528.0	12552.0	18.919	27200	38400	101
3544.0	12552.0	20.162	27200	40000	225
3560.0	12552.0	21.457	27200	41600	355
3576.0	12552.0	22.662	27200	43200	475
3592.0	12552.0	23.140	27200	44800	523
3608.0	12552.0	22.903	27200	46400	499
3624.0	12552.0	22.663	27200	48000	475
3144.0	12536.0	22.978	25600	0	507
3160.0	12536.0	22.907	25600	1600	500
3176.0	12536.0	23.120	25600	3200	521
3192.0	12536.0	23.380	25600	4800	547
3208.0	12536.0	23.573	25600	6400	566
3224.0	12536.0	23.630	25600	8000	572
3240.0	12536.0	23.622	25600	9600	571
3256.0	12536.0	23.010	25600	11200	510
3272.0	12536.0	21.890	25600	12800	398
3288.0	12536.0	21.343	25600	14400	343
3304.0	12536.0	21.243	25600	16000	333
3320.0	12536.0	21.243	25600	17600	333
3336.0	12536.0	21.403	25600	19200	349
3352.0	12536.0	21.550	25600	20800	364

3368.0	12536.0	21.608	25600	22400	370
3384.0	12536.0	21.695	25600	24000	378
3400.0	12536.0	21.320	25600	25600	341
3416.0	12536.0	20.764	25600	27200	285
3432.0	12536.0	20.131	25600	28800	222
3448.0	12536.0	19.500	25600	30400	159
3464.0	12536.0	18.901	25600	32000	99
3480.0	12536.0	17.931	25600	33600	2
3496.0	12536.0	17.931	25600	35200	2
3512.0	12536.0	17.931	25600	36800	2
3528.0	12536.0	18.843	25600	38400	93
3544.0	12536.0	20.053	25600	40000	214
3560.0	12536.0	21.321	25600	41600	341
3576.0	12536.0	22.509	25600	43200	460
3592.0	12536.0	23.029	25600	44800	512
3608.0	12536.0	22.800	25600	46400	489
3624.0	12536.0	22.522	25600	48000	461
3144.0	12520.0	23.395	24000	0	548
3160.0	12520.0	22.910	24000	1600	500
3176.0	12520.0	23.077	24000	3200	517
3192.0	12520.0	23.278	24000	4800	537
3208.0	12520.0	23.565	24000	6400	565
3224.0	12520.0	23.670	24000	8000	576
3240.0	12520.0	23.715	24000	9600	580
3256.0	12520.0	23.430	24000	11200	552
3272.0	12520.0	22.583	24000	12800	467
3288.0	12520.0	21.530	24000	14400	362
3304.0	12520.0	21.392	24000	16000	348
3320.0	12520.0	21.327	24000	17600	342
3336.0	12520.0	21.365	24000	19200	345
3352.0	12520.0	21.465	24000	20800	355
3368.0	12520.0	21.612	24000	22400	370
3384.0	12520.0	21.905	24000	24000	399
3400.0	12520.0	21.591	24000	25600	368
3416.0	12520.0	20.990	24000	27200	308

3432.0	12520.0	20.297	24000	28800	239
3448.0	12520.0	19.612	24000	30400	170
3464.0	12520.0	18.968	24000	32000	106
3480.0	12520.0	17.932	24000	33600	2
3496.0	12520.0	17.932	24000	35200	2
3512.0	12520.0	17.932	24000	36800	2
3528.0	12520.0	18.789	24000	38400	88
3544.0	12520.0	19.929	24000	40000	202
3560.0	12520.0	21.124	24000	41600	321
3576.0	12520.0	22.190	24000	43200	428
3592.0	12520.0	22.875	24000	44800	496
3608.0	12520.0	22.710	24000	46400	480
3624.0	12520.0	22.448	24000	48000	454
3144.0	12504.0	23.718	22400	0	581
3160.0	12504.0	22.817	22400	1600	491
3176.0	12504.0	23.173	22400	3200	526
3192.0	12504.0	23.330	22400	4800	542
3208.0	12504.0	23.563	22400	6400	565
3224.0	12504.0	23.688	22400	8000	578
3240.0	12504.0	23.740	22400	9600	583
3256.0	12504.0	23.628	22400	11200	572
3272.0	12504.0	22.913	22400	12800	500
3288.0	12504.0	21.757	22400	14400	385
3304.0	12504.0	21.507	22400	16000	360
3320.0	12504.0	21.433	22400	17600	352
3336.0	12504.0	21.347	22400	19200	344
3352.0	12504.0	21.427	22400	20800	352
3368.0	12504.0	21.680	22400	22400	377
3384.0	12504.0	21.855	22400	24000	394
3400.0	12504.0	21.879	22400	25600	397
3416.0	12504.0	21.204	22400	27200	329
3432.0	12504.0	20.444	22400	28800	253
3448.0	12504.0	19.706	22400	30400	179
3464.0	12504.0	19.020	22400	32000	111
3480.0	12504.0	17.933	22400	33600	2

3496.0	12504.0	17.933	22400	35200	2
3512.0	12504.0	17.933	22400	36800	2
3528.0	12504.0	18.345	22400	38400	43
3544.0	12504.0	19.847	22400	40000	194
3560.0	12504.0	21.002	22400	41600	309
3576.0	12504.0	22.082	22400	43200	417
3592.0	12504.0	22.942	22400	44800	503
3608.0	12504.0	22.823	22400	46400	491
3624.0	12504.0	22.462	22400	48000	455
3144.0	12488.0	23.323	20800	0	541
3160.0	12488.0	22.830	20800	1600	492
3176.0	12488.0	23.138	20800	3200	523
3192.0	12488.0	23.343	20800	4800	543
3208.0	12488.0	23.552	20800	6400	564
3224.0	12488.0	23.645	20800	8000	573
3240.0	12488.0	23.805	20800	9600	589
3256.0	12488.0	23.597	20800	11200	569
3272.0	12488.0	23.080	20800	12800	517
3288.0	12488.0	22.150	20800	14400	424
3304.0	12488.0	21.708	20800	16000	380
3320.0	12488.0	21.610	20800	17600	370
3336.0	12488.0	21.513	20800	19200	360
3352.0	12488.0	21.470	20800	20800	356
3368.0	12488.0	21.677	20800	22400	377
3384.0	12488.0	21.917	20800	24000	401
3400.0	12488.0	22.003	20800	25600	409
3416.0	12488.0	21.342	20800	27200	343
3432.0	12488.0	20.548	20800	28800	264
3448.0	12488.0	19.768	20800	30400	186
3464.0	12488.0	19.048	20800	32000	114
3480.0	12488.0	17.935	20800	33600	2
3496.0	12488.0	17.935	20800	35200	2
3512.0	12488.0	17.935	20800	36800	2
3528.0	12488.0	18.338	20800	38400	43
3544.0	12488.0	19.812	20800	40000	190

3560.0	12488.0	20.958	20800	41600	305
3576.0	12488.0	22.082	20800	43200	417
3592.0	12488.0	23.034	20800	44800	512
3608.0	12488.0	22.940	20800	46400	503
3624.0	12488.0	22.632	20800	48000	472
3144.0	12472.0	22.795	19200	0	488
3160.0	12472.0	22.782	19200	1600	487
3176.0	12472.0	23.080	19200	3200	517
3192.0	12472.0	23.208	19200	4800	530
3208.0	12472.0	23.392	19200	6400	548
3224.0	12472.0	23.740	19200	8000	583
3240.0	12472.0	23.905	19200	9600	599
3256.0	12472.0	23.710	19200	11200	580
3272.0	12472.0	23.170	19200	12800	526
3288.0	12472.0	22.495	19200	14400	458
3304.0	12472.0	22.125	19200	16000	421
3320.0	12472.0	21.725	19200	17600	381
3336.0	12472.0	21.570	19200	19200	366
3352.0	12472.0	21.520	19200	20800	361
3368.0	12472.0	21.700	19200	22400	379
3384.0	12472.0	21.970	19200	24000	406
3400.0	12472.0	22.111	19200	25600	420
3416.0	12472.0	21.440	19200	27200	353
3432.0	12472.0	20.610	19200	28800	270
3448.0	12472.0	19.786	19200	30400	187
3464.0	12472.0	19.037	19200	32000	113
3480.0	12472.0	17.936	19200	33600	2
3496.0	12472.0	17.936	19200	35200	2
3512.0	12472.0	17.936	19200	36800	2
3528.0	12472.0	18.343	19200	38400	43
3544.0	12472.0	19.803	19200	40000	189
3560.0	12472.0	20.941	19200	41600	303
3576.0	12472.0	22.078	19200	43200	417
3592.0	12472.0	23.047	19200	44800	514
3608.0	12472.0	22.975	19200	46400	506

3624.0	12472.0	22.862	19200	48000	495
3144.0	12456.0	22.490	17600	0	458
3160.0	12456.0	22.595	17600	1600	468
3176.0	12456.0	22.987	17600	3200	508
3192.0	12456.0	23.195	17600	4800	528
3208.0	12456.0	23.395	17600	6400	548
3224.0	12456.0	23.770	17600	8000	586
3240.0	12456.0	23.913	17600	9600	600
3256.0	12456.0	23.830	17600	11200	592
3272.0	12456.0	23.507	17600	12800	560
3288.0	12456.0	22.895	17600	14400	498
3304.0	12456.0	22.480	17600	16000	457
3320.0	12456.0	21.862	17600	17600	395
3336.0	12456.0	21.620	17600	19200	371
3352.0	12456.0	21.580	17600	20800	367
3368.0	12456.0	21.843	17600	22400	393
3384.0	12456.0	22.120	17600	24000	421
3400.0	12456.0	22.254	17600	25600	434
3416.0	12456.0	21.515	17600	27200	360
3432.0	12456.0	20.627	17600	28800	272
3448.0	12456.0	19.743	17600	30400	183
3464.0	12456.0	18.954	17600	32000	104
3480.0	12456.0	17.937	17600	33600	3
3496.0	12456.0	17.937	17600	35200	3
3512.0	12456.0	17.937	17600	36800	3
3528.0	12456.0	18.137	17600	38400	23
3544.0	12456.0	19.788	17600	40000	188
3560.0	12456.0	20.920	17600	41600	301
3576.0	12456.0	22.052	17600	43200	414
3592.0	12456.0	23.023	17600	44800	511
3608.0	12456.0	22.955	17600	46400	504
3624.0	12456.0	22.958	17600	48000	505
3144.0	12440.0	22.372	16000	0	446
3160.0	12440.0	22.503	16000	1600	459
3176.0	12440.0	22.888	16000	3200	498

3192.0	12440.0	23.155	16000	4800	524
3208.0	12440.0	23.468	16000	6400	556
3224.0	12440.0	23.772	16000	8000	586
3240.0	12440.0	23.798	16000	9600	589
3256.0	12440.0	23.690	16000	11200	578
3272.0	12440.0	23.445	16000	12800	553
3288.0	12440.0	22.990	16000	14400	508
3304.0	12440.0	22.630	16000	16000	472
3320.0	12440.0	22.130	16000	17600	422
3336.0	12440.0	21.735	16000	19200	382
3352.0	12440.0	21.630	16000	20800	372
3368.0	12440.0	21.815	16000	22400	390
3384.0	12440.0	22.317	16000	24000	441
3400.0	12440.0	22.241	16000	25600	433
3416.0	12440.0	21.521	16000	27200	361
3432.0	12440.0	20.588	16000	28800	268
3448.0	12440.0	19.622	16000	30400	171
3464.0	12440.0	18.756	16000	32000	84
3480.0	12440.0	17.939	16000	33600	3
3496.0	12440.0	17.939	16000	35200	3
3512.0	12440.0	17.939	16000	36800	3
3528.0	12440.0	17.939	16000	38400	3
3544.0	12440.0	19.762	16000	40000	185
3560.0	12440.0	20.877	16000	41600	297
3576.0	12440.0	21.998	16000	43200	409
3592.0	12440.0	22.987	16000	44800	508
3608.0	12440.0	22.955	16000	46400	504
3624.0	12440.0	23.020	16000	48000	511
3144.0	12424.0	23.188	14400	0	528
3160.0	12424.0	22.545	14400	1600	463
3176.0	12424.0	22.980	14400	3200	507
3192.0	12424.0	23.140	14400	4800	523
3208.0	12424.0	23.435	14400	6400	552
3224.0	12424.0	23.847	14400	8000	594
3240.0	12424.0	23.765	14400	9600	585

3256.0	12424.0	23.535	14400	11200	562
3272.0	12424.0	23.302	14400	12800	539
3288.0	12424.0	22.965	14400	14400	505
3304.0	12424.0	22.708	14400	16000	480
3320.0	12424.0	22.250	14400	17600	434
3336.0	12424.0	21.960	14400	19200	405
3352.0	12424.0	21.907	14400	20800	400
3368.0	12424.0	22.038	14400	22400	413
3384.0	12424.0	22.400	14400	24000	449
3400.0	12424.0	22.324	14400	25600	441
3416.0	12424.0	21.528	14400	27200	362
3432.0	12424.0	20.521	14400	28800	261
3448.0	12424.0	19.448	14400	30400	154
3464.0	12424.0	18.444	14400	32000	53
3480.0	12424.0	17.940	14400	33600	3
3496.0	12424.0	17.940	14400	35200	3
3512.0	12424.0	17.940	14400	36800	3
3528.0	12424.0	17.940	14400	38400	3
3544.0	12424.0	19.725	14400	40000	181
3560.0	12424.0	20.810	14400	41600	290
3576.0	12424.0	21.882	14400	43200	397
3592.0	12424.0	22.792	14400	44800	488
3608.0	12424.0	23.035	14400	46400	512
3624.0	12424.0	23.065	14400	48000	515
3144.0	12408.0	22.665	12800	0	475
3160.0	12408.0	22.390	12800	1600	448
3176.0	12408.0	23.167	12800	3200	526
3192.0	12408.0	23.233	12800	4800	532
3208.0	12408.0	23.532	12800	6400	562
3224.0	12408.0	23.840	12800	8000	593
3240.0	12408.0	23.708	12800	9600	580
3256.0	12408.0	23.455	12800	11200	554
3272.0	12408.0	23.237	12800	12800	533
3288.0	12408.0	22.833	12800	14400	492
3304.0	12408.0	22.567	12800	16000	466

3320.0	12408.0	22.340	12800	17600	443
3336.0	12408.0	22.150	12800	19200	424
3352.0	12408.0	22.125	12800	20800	421
3368.0	12408.0	22.243	12800	22400	433
3384.0	12408.0	22.475	12800	24000	456
3400.0	12408.0	22.395	12800	25600	448
3416.0	12408.0	21.555	12800	27200	364
3432.0	12408.0	20.444	12800	28800	253
3448.0	12408.0	19.300	12800	30400	139
3464.0	12408.0	18.298	12800	32000	39
3480.0	12408.0	17.942	12800	33600	3
3496.0	12408.0	17.941	12800	35200	3
3512.0	12408.0	17.941	12800	36800	3
3528.0	12408.0	17.941	12800	38400	3
3544.0	12408.0	19.685	12800	40000	177
3560.0	12408.0	20.748	12800	41600	284
3576.0	12408.0	21.796	12800	43200	388
3592.0	12408.0	22.723	12800	44800	481
3608.0	12408.0	23.110	12800	46400	520
3624.0	12408.0	23.130	12800	48000	522
3144.0	12392.0	22.353	11200	0	444
3160.0	12392.0	22.323	11200	1600	441
3176.0	12392.0	22.983	11200	3200	507
3192.0	12392.0	23.337	11200	4800	543
3208.0	12392.0	23.647	11200	6400	574
3224.0	12392.0	23.802	11200	8000	589
3240.0	12392.0	23.680	11200	9600	577
3256.0	12392.0	23.522	11200	11200	561
3272.0	12392.0	23.360	11200	12800	545
3288.0	12392.0	22.745	11200	14400	483
3304.0	12392.0	22.403	11200	16000	449
3320.0	12392.0	22.280	11200	17600	437
3336.0	12392.0	22.235	11200	19200	432
3352.0	12392.0	22.290	11200	20800	438
3368.0	12392.0	22.442	11200	22400	453

3384.0	12392.0	22.603	11200	24000	469
3400.0	12392.0	22.723	11200	25600	481
3416.0	12392.0	21.587	11200	27200	368
3432.0	12392.0	20.303	11200	28800	239
3448.0	12392.0	19.122	11200	30400	121
3464.0	12392.0	18.095	11200	32000	18
3480.0	12392.0	17.943	11200	33600	3
3496.0	12392.0	17.943	11200	35200	3
3512.0	12392.0	17.942	11200	36800	3
3528.0	12392.0	17.942	11200	38400	3
3544.0	12392.0	19.656	11200	40000	174
3560.0	12392.0	20.708	11200	41600	280
3576.0	12392.0	21.758	11200	43200	385
3592.0	12392.0	22.719	11200	44800	481
3608.0	12392.0	23.223	11200	46400	531
3624.0	12392.0	23.085	11200	48000	517
3144.0	12376.0	22.420	9600	0	451
3160.0	12376.0	22.442	9600	1600	453
3176.0	12376.0	23.118	9600	3200	521
3192.0	12376.0	23.355	9600	4800	544
3208.0	12376.0	23.593	9600	6400	568
3224.0	12376.0	23.712	9600	8000	580
3240.0	12376.0	23.680	9600	9600	577
3256.0	12376.0	23.595	9600	11200	568
3272.0	12376.0	23.400	9600	12800	549
3288.0	12376.0	22.733	9600	14400	482
3304.0	12376.0	22.288	9600	16000	438
3320.0	12376.0	22.337	9600	17600	443
3336.0	12376.0	22.378	9600	19200	447
3352.0	12376.0	22.468	9600	20800	456
3368.0	12376.0	22.595	9600	22400	468
3384.0	12376.0	22.705	9600	24000	479
3400.0	12376.0	22.649	9600	25600	474
3416.0	12376.0	21.347	9600	27200	344
3432.0	12376.0	19.977	9600	28800	207

3448.0	12376.0	18.770	9600	30400	86
3464.0	12376.0	17.945	9600	32000	3
3480.0	12376.0	17.944	9600	33600	3
3496.0	12376.0	17.944	9600	35200	3
3512.0	12376.0	17.944	9600	36800	3
3528.0	12376.0	17.943	9600	38400	3
3544.0	12376.0	19.644	9600	40000	173
3560.0	12376.0	20.692	9600	41600	278
3576.0	12376.0	21.738	9600	43200	383
3592.0	12376.0	22.687	9600	44800	478
3608.0	12376.0	23.192	9600	46400	528
3624.0	12376.0	23.045	9600	48000	513
3144.0	12360.0	22.632	8000	0	472
3160.0	12360.0	22.673	8000	1600	476
3176.0	12360.0	23.173	8000	3200	526
3192.0	12360.0	23.390	8000	4800	548
3208.0	12360.0	23.540	8000	6400	563
3224.0	12360.0	23.635	8000	8000	572
3240.0	12360.0	23.743	8000	9600	583
3256.0	12360.0	23.603	8000	11200	569
3272.0	12360.0	23.305	8000	12800	539
3288.0	12360.0	22.638	8000	14400	473
3304.0	12360.0	22.343	8000	16000	443
3320.0	12360.0	22.517	8000	17600	461
3336.0	12360.0	22.603	8000	19200	469
3352.0	12360.0	22.673	8000	20800	476
3368.0	12360.0	22.785	8000	22400	487
3384.0	12360.0	22.887	8000	24000	498
3400.0	12360.0	22.083	8000	25600	417
3416.0	12360.0	20.780	8000	27200	287
3432.0	12360.0	19.486	8000	28800	157
3448.0	12360.0	18.165	8000	30400	25
3464.0	12360.0	17.946	8000	32000	3
3480.0	12360.0	17.946	8000	33600	3
3496.0	12360.0	17.945	8000	35200	3

3512.0	12360.0	17.945	8000	36800	3
3528.0	12360.0	18.320	8000	38400	41
3544.0	12360.0	19.670	8000	40000	176
3560.0	12360.0	20.714	8000	41600	280
3576.0	12360.0	21.748	8000	43200	384
3592.0	12360.0	22.685	8000	44800	477
3608.0	12360.0	23.035	8000	46400	512
3624.0	12360.0	23.022	8000	48000	511
3144.0	12344.0	22.823	6400	0	491
3160.0	12344.0	22.903	6400	1600	499
3176.0	12344.0	23.120	6400	3200	521
3192.0	12344.0	23.313	6400	4800	540
3208.0	12344.0	23.540	6400	6400	563
3224.0	12344.0	23.590	6400	8000	568
3240.0	12344.0	23.665	6400	9600	575
3256.0	12344.0	23.712	6400	11200	580
3272.0	12344.0	23.325	6400	12800	541
3288.0	12344.0	22.663	6400	14400	475
3304.0	12344.0	22.478	6400	16000	457
3320.0	12344.0	22.610	6400	17600	470
3336.0	12344.0	22.728	6400	19200	482
3352.0	12344.0	22.872	6400	20800	496
3368.0	12344.0	23.114	6400	22400	520
3384.0	12344.0	22.547	6400	24000	464
3400.0	12344.0	21.366	6400	25600	345
3416.0	12344.0	20.046	6400	27200	213
3432.0	12344.0	18.896	6400	28800	98
3448.0	12344.0	17.948	6400	30400	4
3464.0	12344.0	17.947	6400	32000	4
3480.0	12344.0	17.947	6400	33600	4
3496.0	12344.0	17.946	6400	35200	3
3512.0	12344.0	17.946	6400	36800	3
3528.0	12344.0	18.733	6400	38400	82
3544.0	12344.0	19.739	6400	40000	183
3560.0	12344.0	20.790	6400	41600	288

3576.0	12344.0	21.806	6400	43200	389
3592.0	12344.0	22.663	6400	44800	475
3608.0	12344.0	22.888	6400	46400	498
3624.0	12344.0	22.907	6400	48000	500
3144.0	12328.0	22.985	4800	0	507
3160.0	12328.0	23.048	4800	1600	514
3176.0	12328.0	23.210	4800	3200	530
3192.0	12328.0	23.370	4800	4800	546
3208.0	12328.0	23.542	4800	6400	563
3224.0	12328.0	23.555	4800	8000	564
3240.0	12328.0	23.700	4800	9600	579
3256.0	12328.0	23.820	4800	11200	591
3272.0	12328.0	23.495	4800	12800	558
3288.0	12328.0	22.795	4800	14400	488
3304.0	12328.0	22.597	4800	16000	469
3320.0	12328.0	22.760	4800	17600	485
3336.0	12328.0	22.927	4800	19200	502
3352.0	12328.0	23.103	4800	20800	519
3368.0	12328.0	23.171	4800	22400	526
3384.0	12328.0	22.065	4800	24000	415
3400.0	12328.0	20.631	4800	25600	272
3416.0	12328.0	19.248	4800	27200	134
3432.0	12328.0	17.949	4800	28800	4
3448.0	12328.0	17.949	4800	30400	4
3464.0	12328.0	17.948	4800	32000	4
3480.0	12328.0	17.948	4800	33600	4
3496.0	12328.0	17.947	4800	35200	4
3512.0	12328.0	17.947	4800	36800	4
3528.0	12328.0	18.781	4800	38400	87
3544.0	12328.0	19.851	4800	40000	194
3560.0	12328.0	20.931	4800	41600	302
3576.0	12328.0	21.957	4800	43200	405
3592.0	12328.0	22.826	4800	44800	491
3608.0	12328.0	22.790	4800	46400	488
3624.0	12328.0	22.872	4800	48000	496

3144.0	12312.0	23.112	3200	0	520
3160.0	12312.0	23.160	3200	1600	525
3176.0	12312.0	23.335	3200	3200	542
3192.0	12312.0	23.462	3200	4800	555
3208.0	12312.0	23.580	3200	6400	567
3224.0	12312.0	23.610	3200	8000	570
3240.0	12312.0	23.698	3200	9600	579
3256.0	12312.0	23.958	3200	11200	605
3272.0	12312.0	23.642	3200	12800	573
3288.0	12312.0	23.007	3200	14400	510
3304.0	12312.0	22.747	3200	16000	484
3320.0	12312.0	22.885	3200	17600	497
3336.0	12312.0	23.175	3200	19200	526
3352.0	12312.0	23.333	3200	20800	542
3368.0	12312.0	22.690	3200	22400	478
3384.0	12312.0	21.381	3200	24000	347
3400.0	12312.0	19.870	3200	25600	196
3416.0	12312.0	18.432	3200	27200	52
3432.0	12312.0	17.951	3200	28800	4
3448.0	12312.0	17.950	3200	30400	4
3464.0	12312.0	17.949	3200	32000	4
3480.0	12312.0	17.949	3200	33600	4
3496.0	12312.0	17.949	3200	35200	4
3512.0	12312.0	17.948	3200	36800	4
3528.0	12312.0	18.882	3200	38400	97
3544.0	12312.0	20.019	3200	40000	211
3560.0	12312.0	21.129	3200	41600	322
3576.0	12312.0	22.141	3200	43200	423
3592.0	12312.0	22.944	3200	44800	503
3608.0	12312.0	22.702	3200	46400	479
3624.0	12312.0	22.895	3200	48000	498
3144.0	12296.0	23.190	1600	0	528
3160.0	12296.0	23.263	1600	1600	535
3176.0	12296.0	23.423	1600	3200	551
3192.0	12296.0	23.520	1600	4800	561

3208.0	12296.0	23.622	1600	6400	571
3224.0	12296.0	23.630	1600	8000	572
3240.0	12296.0	23.665	1600	9600	575
3256.0	12296.0	23.975	1600	11200	606
3272.0	12296.0	23.890	1600	12800	598
3288.0	12296.0	23.235	1600	14400	532
3304.0	12296.0	22.878	1600	16000	497
3320.0	12296.0	22.940	1600	17600	503
3336.0	12296.0	23.335	1600	19200	542
3352.0	12296.0	23.001	1600	20800	509
3368.0	12296.0	22.019	1600	22400	411
3384.0	12296.0	20.692	1600	24000	278
3400.0	12296.0	19.239	1600	25600	133
3416.0	12296.0	17.952	1600	27200	4
3432.0	12296.0	17.952	1600	28800	4
3448.0	12296.0	17.951	1600	30400	4
3464.0	12296.0	17.951	1600	32000	4
3480.0	12296.0	17.950	1600	33600	4
3496.0	12296.0	17.950	1600	35200	4
3512.0	12296.0	18.016	1600	36800	10
3528.0	12296.0	19.109	1600	38400	120
3544.0	12296.0	20.279	1600	40000	237
3560.0	12296.0	21.398	1600	41600	349
3576.0	12296.0	22.394	1600	43200	448
3592.0	12296.0	23.054	1600	44800	514
3608.0	12296.0	22.638	1600	46400	473
3624.0	12296.0	22.788	1600	48000	488
3144.0	12280.0	23.235	0	0	532
3160.0	12280.0	23.370	0	1600	546
3176.0	12280.0	23.528	0	3200	562
3192.0	12280.0	23.685	0	4800	577
3208.0	12280.0	23.698	0	6400	579
3224.0	12280.0	23.740	0	8000	583
3240.0	12280.0	23.858	0	9600	595
3256.0	12280.0	23.962	0	11200	605

3272.0	12280.0	23.872	0	12800	596
3288.0	12280.0	23.275	0	14400	536
3304.0	12280.0	22.935	0	16000	502
3320.0	12280.0	22.997	0	17600	509
3336.0	12280.0	23.257	0	19200	535
3352.0	12280.0	22.592	0	20800	468
3368.0	12280.0	21.426	0	22400	351
3384.0	12280.0	20.103	0	24000	219
3400.0	12280.0	18.863	0	25600	95
3416.0	12280.0	17.953	0	27200	4
3432.0	12280.0	17.953	0	28800	4
3448.0	12280.0	17.952	0	30400	4
3464.0	12280.0	17.952	0	32000	4
3480.0	12280.0	17.951	0	33600	4
3496.0	12280.0	17.950	0	35200	4
3512.0	12280.0	18.556	0	36800	64
3528.0	12280.0	19.551	0	38400	164
3544.0	12280.0	20.655	0	40000	274
3560.0	12280.0	21.727	0	41600	382
3576.0	12280.0	22.707	0	43200	480
3592.0	12280.0	23.151	0	44800	524
3608.0	12280.0	22.573	0	46400	466
3624.0	12280.0	22.650	0	48000	474

Appendix K. Listing of Output Files for the Hypothetical LiDAR Example

The bathymetry and (or) LiDAR genetic algorithm (GA) program outputs two files: stats_out.csv and bi_out.txt. These files are created each time the program is ran and will overwrite if they exist. If these files are needed, it is suggested that they be renamed before the program is reran. The file stats_out.csv is a comma delimited text file containing fitness values for the best, average, median, worst, and RMSE (Root Mean Square Error) at every generation. Because the file has a csv suffix, it can easily be read into a spreadsheet. The file bi_out.txt is a space delimited text file containing the final GA-produced bathymetry and (or) LiDAR dataset (x, y, and z). The first column in file bi_out.txt represents the x-value (easting), the second column represents the y-value (northing), and the third column represents the z-value (elevation or height).

Tables K.1 and K.2 are computer listings of output files for run 8 of the hypothetical LiDAR example (Chapter 4.5.1), which was ran for 100 generations. Table K.1 is a listing of the stat_out.csv file and Table K.2 is a listing of the bi_out.csv.

Table K.1. Listing of output file stat_out.csv.

```
Generation,Best,Best,Average,Median,Median,Worst,Worst,RMSE
No.,Weight,Fitness,Fitness,Weight,Fitness,Weight,Fitness,Fitness
   1,  141,    0.299E+03,    0.768E+34,  139,  0.113E+06,  176,  0.614E+36,   0.68238E+35
   2,  141,    0.299E+03,    0.112E+09,   41,  0.956E+08,   40,  0.232E+09,   0.74070E+08
   3,  141,    0.299E+03,    0.676E+08,   63,  0.787E+08,   49,  0.201E+09,   0.57818E+08
   4,  141,    0.299E+03,    0.276E+08,   70,  0.442E+07,   64,  0.164E+09,   0.39962E+08
   5,  141,    0.299E+03,    0.323E+08,   79,  0.458E+07,   84,  0.111E+09,   0.33269E+08
   6,  141,    0.299E+03,    0.158E+08,  100,  0.953E+06,  106,  0.778E+08,   0.21660E+08
   7,  141,    0.299E+03,    0.125E+08,  108,  0.445E+06,   98,  0.479E+08,   0.16108E+08
   8,  141,    0.299E+03,    0.187E+07,  130,  0.140E+06,  113,  0.325E+08,   0.61743E+07
   9,  144,    0.196E+03,    0.760E+11,  128,  0.652E+05,  153,  0.608E+13,   0.67498E+12
  10,  144,    0.196E+03,    0.142E+17,  137,  0.848E+05,  158,  0.881E+18,   0.10043E+18
  11,  144,    0.166E+03,    0.110E+34,  162,  0.266E+22,  175,  0.813E+35,   0.90543E+34
  12,  144,    0.166E+03,    0.222E+39,  163,  0.105E+24,  180,  0.122E+41,   0.14791E+40
  13,  144,    0.166E+03,    0.346E+53,  174,  0.498E+34,  195,  0.276E+55,   0.34615E+53
  14,  144,    0.166E+03,    0.171E+56,  177,  0.902E+37,  197,  0.130E+58,   0.17129E+56
  15,  144,    0.166E+03,    0.182E+74,  186,  0.137E+47,  216,  0.146E+76,   0.18209E+74
  16,  144,    0.166E+03,    0.319E+79,  196,  0.201E+56,  221,  0.251E+81,   0.31861E+79
  17,  144,    0.166E+03,    0.107E+93,  211,  0.120E+71,  235,  0.498E+94,   0.10719E+93
  18,  144,    0.166E+03,    0.995+112,  231,  0.186E+91,  254,  0.796+114,   0.99461+112
  19,  144,    0.166E+03,    0.380+111,  252,  0.201+112,  253,  0.284+113,   0.37992+111
  20,  144,    0.166E+03,    0.621+112,  243,  0.167+103,  255,  0.493+114,   0.62097+112
  21,  144,    0.166E+03,    0.412+124,  245,  0.612+104,  266,  0.329+126,   0.41184+124
  22,  144,    0.166E+03,    0.347+137,  246,  0.148+106,  279,  0.278+139,   0.34720+137
```

23,	144,	0.166E+03,	0.731+141,	256,	0.486+116,	283,	0.444+143,	0.73051+141
24,	144,	0.166E+03,	0.788+148,	254,	0.506+113,	290,	0.630+150,	0.78790+148
25,	144,	0.166E+03,	0.113+157,	252,	0.468+112,	299,	0.905+158,	+Infinity
26,	144,	0.166E+03,	0.343+151,	284,	0.648+143,	293,	0.274+153,	0.34262+151
27,	144,	0.166E+03,	0.535+155,	279,	0.397+139,	297,	0.386+157,	+Infinity
28,	144,	0.166E+03,	0.338+160,	286,	0.341+146,	303,	0.270+162,	+Infinity
29,	144,	0.166E+03,	0.932+175,	291,	0.281+151,	317,	0.736+177,	+Infinity
30,	144,	0.166E+03,	0.118+171,	295,	0.184+155,	312,	0.915+172,	+Infinity
31,	144,	0.166E+03,	0.834+181,	304,	0.634+164,	323,	0.667+183,	+Infinity
32,	144,	0.166E+03,	0.719+188,	301,	0.199+161,	331,	0.576+190,	+Infinity
33,	144,	0.166E+03,	0.317+184,	313,	0.327+173,	326,	0.254+186,	+Infinity
34,	144,	0.166E+03,	0.713+180,	294,	0.919+153,	322,	0.540+182,	+Infinity
35,	144,	0.166E+03,	0.214+184,	308,	0.200+168,	326,	0.171+186,	+Infinity
36,	144,	0.166E+03,	0.478+204,	279,	0.332+139,	346,	0.382+206,	+Infinity
37,	144,	0.166E+03,	0.617+195,	303,	0.817+162,	338,	0.493+197,	+Infinity
38,	144,	0.166E+03,	0.548+203,	323,	0.251+183,	345,	0.438+205,	+Infinity
39,	144,	0.166E+03,	0.506+218,	307,	0.513+167,	360,	0.404+220,	+Infinity
40,	144,	0.166E+03,	0.365+207,	321,	0.207+181,	349,	0.292+209,	+Infinity
41,	144,	0.166E+03,	0.319+202,	324,	0.110+183,	344,	0.240+204,	+Infinity
42,	144,	0.166E+03,	0.102+211,	331,	0.766+190,	353,	0.785+212,	+Infinity
43,	144,	0.166E+03,	0.908+210,	334,	0.313+194,	352,	0.611+212,	+Infinity
44,	144,	0.166E+03,	0.281+214,	321,	0.467+181,	356,	0.209+216,	+Infinity
45,	144,	0.166E+03,	0.104+221,	337,	0.303+197,	363,	0.831+222,	+Infinity
46,	144,	0.166E+03,	0.138+223,	343,	0.431+202,	365,	0.710+224,	+Infinity
47,	144,	0.166E+03,	0.429+227,	320,	0.417+180,	369,	0.343+229,	+Infinity

48,	144,	0.166E+03,	0.143+213,	337,	0.108+197,	354,	0.585+214,	+Infinity
49,	144,	0.166E+03,	0.177+215,	344,	0.536+204,	357,	0.141+217,	+Infinity
50,	144,	0.166E+03,	0.485+218,	343,	0.124+204,	360,	0.352+220,	+Infinity
51,	144,	0.166E+03,	0.410+231,	351,	0.115+211,	373,	0.328+233,	+Infinity
52,	144,	0.166E+03,	0.598+227,	346,	0.390+206,	369,	0.478+229,	+Infinity
53,	144,	0.166E+03,	0.346+227,	338,	0.108+198,	369,	0.277+229,	+Infinity
54,	144,	0.166E+03,	0.524+212,	337,	0.347+197,	355,	0.419+214,	+Infinity
55,	144,	0.166E+03,	0.540+218,	339,	0.486+197,	360,	0.204+220,	+Infinity
56,	144,	0.166E+03,	0.105+224,	337,	0.436+197,	366,	0.832+225,	+Infinity
57,	144,	0.166E+03,	0.105+222,	350,	0.717+210,	363,	0.757+223,	+Infinity
58,	144,	0.166E+03,	0.262+237,	362,	0.704+222,	379,	0.209+239,	+Infinity
59,	144,	0.166E+03,	0.275+232,	333,	0.476+192,	374,	0.220+234,	+Infinity
60,	144,	0.166E+03,	0.614+216,	344,	0.137+204,	358,	0.449+218,	+Infinity
61,	144,	0.166E+03,	0.358+220,	345,	0.167+205,	362,	0.256+222,	+Infinity
62,	144,	0.166E+03,	0.192+240,	354,	0.733+213,	382,	0.154+242,	+Infinity
63,	144,	0.166E+03,	0.232+226,	367,	0.380+226,	368,	0.182+228,	+Infinity
64,	144,	0.166E+03,	0.221+240,	355,	0.176+215,	382,	0.143+242,	+Infinity
65,	144,	0.166E+03,	0.128+252,	364,	0.123+224,	394,	0.102+254,	+Infinity
66,	144,	0.166E+03,	0.625+247,	341,	0.199+201,	389,	0.450+249,	+Infinity
67,	144,	0.166E+03,	0.307+249,	344,	0.422+204,	391,	0.242+251,	+Infinity
68,	144,	0.166E+03,	0.635+241,	377,	0.332+237,	383,	0.508+243,	+Infinity
69,	144,	0.166E+03,	0.283+235,	360,	0.747+219,	377,	0.227+237,	+Infinity
70,	144,	0.166E+03,	0.484+247,	379,	0.916+238,	389,	0.387+249,	+Infinity
71,	144,	0.166E+03,	0.234+256,	358,	0.335+218,	398,	0.188+258,	+Infinity
72,	144,	0.166E+03,	0.464+256,	357,	0.458+217,	398,	0.371+258,	+Infinity

73,	144,	0.166E+03,	0.147+259,	371,	0.290+231,	401,	0.118+261,	+Infinity
74,	144,	0.166E+03,	0.616+256,	359,	0.404+218,	398,	0.452+258,	+Infinity
75,	144,	0.166E+03,	0.575+249,	380,	0.596+239,	391,	0.460+251,	+Infinity
76,	144,	0.166E+03,	0.486+252,	368,	0.202+228,	394,	0.389+254,	+Infinity
77,	144,	0.166E+03,	0.475+262,	377,	0.111+238,	404,	0.377+264,	+Infinity
78,	144,	0.166E+03,	0.343+265,	400,	0.847+260,	407,	0.274+267,	+Infinity
79,	144,	0.166E+03,	0.981+261,	367,	0.166+225,	404,	0.785+263,	+Infinity
80,	144,	0.166E+03,	0.828+268,	372,	0.125+232,	411,	0.662+270,	+Infinity
81,	144,	0.166E+03,	0.265+258,	380,	0.530+240,	401,	0.208+260,	+Infinity
82,	144,	0.166E+03,	0.542+250,	359,	0.131+219,	392,	0.429+252,	+Infinity
83,	144,	0.166E+03,	0.427+257,	386,	0.347+246,	399,	0.318+259,	+Infinity
84,	144,	0.166E+03,	0.272+254,	393,	0.468+253,	397,	0.217+256,	+Infinity
85,	144,	0.166E+03,	0.402+247,	374,	0.299+233,	389,	0.295+249,	+Infinity
86,	144,	0.166E+03,	0.200+245,	361,	0.480+220,	387,	0.153+247,	+Infinity
87,	144,	0.166E+03,	0.161+270,	368,	0.134+228,	412,	0.129+272,	+Infinity
88,	144,	0.166E+03,	0.191+263,	372,	0.973+231,	405,	0.151+265,	+Infinity
89,	144,	0.166E+03,	0.842+245,	360,	0.383+220,	387,	0.452+247,	+Infinity
90,	144,	0.166E+03,	0.153+264,	403,	0.377+263,	406,	0.117+266,	+Infinity
91,	144,	0.166E+03,	0.252+259,	360,	0.260+220,	401,	0.202+261,	+Infinity
92,	144,	0.166E+03,	0.539+267,	379,	0.596+239,	409,	0.392+269,	+Infinity
93,	144,	0.166E+03,	0.116+258,	380,	0.330+240,	400,	0.722+259,	+Infinity
94,	144,	0.166E+03,	0.240+252,	377,	0.444+237,	394,	0.187+254,	+Infinity
95,	144,	0.166E+03,	0.175+278,	371,	0.180+231,	420,	0.117+280,	+Infinity
96,	144,	0.166E+03,	0.347+262,	361,	0.663+221,	404,	0.259+264,	+Infinity
97,	144,	0.166E+03,	0.835+264,	363,	0.569+223,	406,	0.330+266,	+Infinity

98,	144,	0.166E+03,	0.330+272,	391,	0.667+251,	414,	0.264+274,	+Infinity
99,	144,	0.166E+03,	0.869+268,	376,	0.488+236,	410,	0.382+270,	+Infinity
100,	144,	0.166E+03,	0.276+271,	384,	0.347+244,	413,	0.221+273,	+Infinity

Table K.2. Listing of output file bi_out.csv. (The first column represents the x-value, the second column represents the y-value, and the third column represents the z-value.)

BEST INDIVIDUAL

X	Y	Z
0	0	532
0	35200	485
0	36800	495
0	48000	598
1600	0	546
1600	1600	535
1600	9600	453
1600	25600	499
3200	1600	551
3200	14400	507
3200	27200	518
3200	28800	519
3200	36800	534
3200	43200	735
3200	44800	655
4800	6400	540
4800	12800	532
4800	14400	523
4800	48000	628
6400	4800	563
6400	6400	562
6400	11200	573
6400	20800	564
6400	35200	570
6400	44800	676
6400	48000	666
8000	0	582
8000	9600	580
8000	12800	592
8000	24000	576

8000	32000	580
8000	33600	578
8000	38400	504
8000	40000	480
9600	4800	578
9600	41600	449
9600	43200	512
9600	48000	447
11200	12800	554
11200	24000	551
11200	41600	364
11200	44800	342
12800	4800	558
12800	8000	539
12800	11200	544
12800	41600	257
14400	41600	233
16000	1600	496
16000	8000	443
16000	22400	359
16000	25600	333
16000	30400	329
16000	32000	341
16000	48000	361
17600	1600	503
17600	22400	352
17600	24000	341
19200	22400	343
19200	25600	349
19200	27200	351
20800	9600	455
20800	20800	355
20800	36800	248
20800	46400	97
22400	6400	520

22400	19200	378
22400	32000	319
24000	25600	378
24000	28800	346
25600	11200	481
25600	12800	448
25600	16000	433
27200	4800	133
27200	32000	194
27200	48000	0
28800	3200	4
28800	20800	263
28800	22400	253
28800	43200	0
30400	0	4
30400	1600	4
30400	4800	3
30400	6400	3
30400	8000	25
30400	20800	185
30400	36800	94
30400	43200	0
32000	3200	3
32000	25600	99
32000	27200	157
32000	38400	0
32000	44800	31
33600	1600	3
33600	12800	3
33600	28800	107
33600	30400	70
33600	33600	1
35200	8000	3
35200	12800	3
35200	20800	2

35200	30400	1
35200	33600	1
35200	43200	207
36800	6400	3
36800	9600	3
36800	19200	2
36800	44800	385
38400	3200	97
38400	14400	3
38400	17600	22
38400	20800	42
38400	28800	103
38400	40000	336
40000	3200	210
40000	6400	182
40000	17600	187
40000	36800	379
41600	1600	348
41600	6400	287
41600	41600	501
41600	44800	496
43200	20800	417
43200	22400	417
43200	27200	475
43200	44800	513
44800	1600	514
44800	27200	523
44800	35200	532
44800	36800	532
44800	48000	509
46400	14400	512
46400	22400	491
46400	24000	480
46400	38400	533
46400	41600	509

46400	46400	496
48000	0	473
48000	3200	498
48000	8000	511
48000	11200	517
48000	12800	521
48000	22400	455
48000	44800	487
48000	48000	462

Appendix L. Permission to Use isort (Appendix I.4)

ISORT sorts array IX and optionally makes the same interchanges in array IY. The array IX may be sorted in increasing order or decreasing order.

file: slatec/src/isort.f (http://www.netlib.org/slatec/src/)
for: Sort an array and optionally make the same interchanges in
gams: N6A2A
by: Jones, R. E., (SNLA)

SLATEC Common Mathematical Library, Version 4.1, July 1993, a comprehensive software library containing over 1400 general purpose mathematical and statistical routines written in Fortran 77.

*/
* The authors of this software is R.E. Jones
* Copyright (c) 1993 by SNLA.
* Permission to use, copy, modify, and distribute this software for any
* purpose without fee is hereby granted, provided that this entire notice
* is included in all copies of any software which is or includes a copy
* or modification of this software and in all copies of the supporting
* documentation for such software.
* THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
* WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR <org> MAKE ANY
* REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
MERCHANTABILITY
* OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
*/

Here is another notice, from the IJG jpeg library source, that may be a useful model:

In plain English:

1. We don't promise that this software works. (But if you find any bugs, please let us know!)
2. You can use this software for whatever you want. You don't have to pay us.
3. You may not pretend that you wrote this software. If you use it in a program, you must acknowledge somewhere in your documentation that you've used the IJG code.

In legalese:

The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.

This software is copyright (C) 1991-1996, Thomas G. Lane. All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or portions thereof) for any purpose, without fee, subject to these conditions:

1. If any part of the source code for this software is distributed, then this README file must be included, with this copyright and no-warranty notice unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation.
2. If only executable code is distributed, then the accompanying documentation must state that "this software is based in part on the work of the Independent JPEG Group".
3. Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind.

These conditions apply to any software derived from or based on the IJG code, not just to the unmodified library. If you use our work, you ought to acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name in advertising or publicity relating to this software or products derived from it. This software may be referred to only as "the Independent JPEG Group's software".

We specifically permit and encourage the use of this software as the basis of commercial products, provided that all warranty or liability claims are assumed by the product vendor.

Appendix M. Permission to Use locpt (Appendix I.16)

The library locpt.90 determines if a point resides inside a polygon. This code was written by Alan Miller and is released into the public domain (https://jblevins.org/mirror/amiller/, webpage updated 4 February 2004).

Appendix N. Permission to Use GEOMPACK (Appendix I.18)

geompack https://people.sc.fsu.edu/~jburkardt/f77_src/geompack/geompack.f (accessed 10/29/2017)

Subroutines in geompack.f: vbedg, swapec

Functions in geompack.f: diaedg, lrline, i_wrap, i_modp

This code is distributed under the GNU LGPL license. See section 2 about conveying modified versions.

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

https://www.gnu.org/copyleft/lesser.html

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined

Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
    - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
    - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the

Appendix I. Listing of MATLAB file for Spectral Analysis

```
%%
%
%
%  Power Spectral Density (PSD) of a dataset
%
%

% clear workspace...
clear; clc; close all;

% read source data file.
datfile = 'braided_original.txt'; % 2 columns of space separated numbers.
fid = fopen(datfile);
C   = textscan(fid,'%f%f', 'MultipleDelimsAsOne',1);
fclose(fid);

% get data columns:
t = C{1}; % x-value  [distance from left bank, feet]
y = C{2}; % y-value [elevation, feet]

% make sure sorted in x-value....
[t,iSORT] = sort(t);
y = y(iSORT);

% make sure x-value starts at 0...
t = t - min(t);

% get the interval (time step) and make sure all the same...
dt = unique(diff(t));
if numel(dt) ~= 1,
   error('Data is not sampled at a uniform rate (time steps not all the same).');
end


% set interval or time step
dt = 0.5;

% get the sampling frequency...
sampFreqHz = 1/dt;

% get number of values...
n = numel(t);
```

```
% plot time-series...
scr_sz = get(0,'ScreenSize'); % [x,y,w,h].
figure('Color','w', 'Position',[5 5 0.9*scr_sz(3:4)]); axes; hold on;
subplot(2,1,1);
plot(t,y);
title('\bfTime-Series Plot (Time Domain)');
xlabel('TIME, IN SECONDS');
ylabel('VALUE');
box on;


%%
% spectral analysis: this uses technique in Malab documentation.
% type "doc fftdemo.m" for more information.

% compute FFT...
Y = fft(y,n);

% compute the power spectral density...
Pyy = Y.*conj(Y)/n;

% compute the corresponding frequencies...
f = (sampFreqHz/n)*(0:n-1)';

% plot power spectral density...
subplot(2,1,2); hold on;
plot(f,Pyy);
title('\bfPower Spectral Density (Frequency Domain)');
xlim([0,sampFreqHz/2]);
set(gca, 'XScale','log', 'YScale','log');
xlabel('FREQUENCY, IN HERTZ');
box on;

%-------------------------------------------------
% "llfft.m" method from 'Matlab WDS Toolkit'
% See: http://www.nortekusa.com/usa/knowledge-center/table-of-contents/waves
%
% The core code from "llfft.m" was copied here and modified to work with how my variables are
defined.
%
% This starts from scratch (uing their code) to compute the PSD from the raw data, then performs a
smooth.

% nF: nominal number of frequency bands for computing average (the actual number of bands will
be less)
%
%     This essentially just smooths the result by taking the means of logrithmically-spaced partitions:
```

```
%     less 'bands' = more smoothing.
%
nF = 70; % this value used in the 'WDS Toolkit' demo.

x  = y; % reset for new compuatation.
np = length(x);  % number of points.
if mod(np,2)==1, % make even number of points.
   np = np - 1;
   x = x(1:np);
end

Dt = np*dt;          % Dt = total duration of TS.
f  = (1:(np/2))'./Dt; % frequency array up to Nyquist.

xf = abs(fft(x)); % compute spectrum.

xp = xf(2:(np/2+1)).^2; % compute power spectrum & ignore zero freq.
xp = xp*2/np^2/f(1);    % scale power spectrum.

% jvrabel: here comes the log-smoothing part.
% if it is just taking means of log-spaced intervals, it can be done more simply than this.

% logarithmically average spectrum into nF uniformly-spaced "log10" frequency bands.
lf  = log(f);
dlf = 1.000000001*(lf(end) - lf(1)) / nF ; % log frequency increment.
NDX = 1 + floor((lf-lf(1))/dlf);
AA  = [find(diff(NDX)>0)' length(f)]; % array of transitions plus final f.

Cs = cumsum(xp);
Cf = cumsum(f);
F  = [Cf(AA(1))    diff(Cf(AA)') ]./ [AA(1) diff(AA)];
S  = [Cs(AA(1),:); diff(Cs(AA,:))]./([AA(1) diff(AA)]');

% add unsmoothed and log-smooth to plot...
plot(f,xp,'g.');
plot(F,S, 'r' );
legend({ ...
      'Raw PSD (fftdemo.m method)' ...
      'Raw PSD (WDS Toolkit method)' ...
      ['Smoothed (' num2str(nF) ' bands)'] ...
   }, ...
   'Location','EastOutside');
legend boxoff;
return;
```