Triangulation-based L_1-fitting of Terrain Surfaces

Javier Bernal and Christoph Witzgall

National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, U. S. A. [1]

[1]Partially supported by DARPA/TEC MIPR 97-5039

Abstract. Given a planar triangulation, the goal is to select elevations at its vertices so that resulting piecewise-linear triangulated surface approximates specified elevations using the L_1-norm as the primary measure-of-fit. Several suboptimal algorithms relating to that problem have been devised and implemented, and are described here, as part of TIN prototype software for terrain modeling in the context of distributed simulation as well as elevation data editing. L_1-approximation acts as a median filter and thus provides advantages for "bare earth" representation. For added flexibility, a "sign-oriented scaling" scheme has been developed, which generalizes standard norm-based measures-of-fit. L_1-approximation is either tackled directly or by iterating suitably weighted L_2-approximations. Numerical experiments are reported.

Key words. bare earth, distributed simulation, L_1-approximation, L_2-approximation, linear programming, median filter, terrain surface modeling, TIN, triangulation

1. Introduction

"Triangulated irregular networks (TINs)" are playing an increasingly prominent role in modeling terrain surfaces for purposes such as animation, distributed simulation, and geodetic volumetrics (e.g. [3]). Typically, there is a rectangular *map* area together with a set R of location points $p = (\hat{x}_p, \hat{y}_p)$ on that map each with elevation $\hat{z}_p$. Terrain is to be modeled as a surface based on those elevation data.

A *triangulated irregular network* is a planar triangulation which arises as the footprint of a triangulated surface $z = z(x, y)$, that is, a piecewise linear surface consisting of flat triangles joined continuously together in space. That surface is referred to as TIN-*surface*. Under *triangulation*, we understand any covering of the map by triangles such that the triangle interiors do not overlap and any two triangle edges which meet in more than one point are identical.

Typically, the vertices of the triangulation have previously been selected as *critical points* based on "greedy" criteria. We denote the set of those vertices by S. The TIN surface $z = z(x, y)$ is uniquely determined by the elevations z_s at the critical points $s = (x_s, y_s) \in S$ and the footprint triangulation, the TIN. Indeed, the elevations at the three corners of any triangle in the TIN uniquely determine a plane through the corresponding elevation points and, therefore, a triangle in space.

The object then is to approximate given elevation data by a TIN surface – as accurately as possible with the number of triangles in the TIN not exceeding a prescribed upper limit, the "budget". Such restrictions are necessary, for instance, to meet the high speed requirements of real-time visualization.

Generally, there are two aspects of this optimization. The first is the selection of the TIN itself, that is, of the critical points and their triangulation. The second is to assign elevations at those critical points so as to optimize some measure-of-fit to the given elevations while keeping the triangulation fixed. This work is concerned with that second aspect of optimization.

There are different measures-of-fit in general use to assess the accuracy of such an approximation. Most are based on the notion of a *residual* at each given elevation location $p = (\hat{x}_p, \hat{y}_p) \in R$,

$$r_p \;=\; \hat{z}_p - \bar{z}_p \;=\; \hat{z}_p - z(\hat{x}_p, \hat{y}_p),$$

where $\bar{z}_p$ denotes the elevation $z(\hat{x}_p, \hat{y}_p)$ which the TIN surface assumes at location $(\hat{x}_p, \hat{y}_p)$.

Two commonly considered measures-of-fit are the *maximum deviation*

$$\mathrm{MAX} \;=\; \mathrm{max}_{p\in R}|r_p|$$

and the *root mean square*

$$\mathrm{RMS} \;=\; \sqrt{\frac{1}{|R|}\sum_{p\in R} r_p^2},$$

where $|R|$ is the number of elevation locations. Both quantities relate to norms of vectors. MAX is an instance of the

$$L_\infty - norm,$$

also called Chebychev norm. RMS is a scaled version of the

$$L_2 - norm$$

or least-squares norm.

There is, however, a problem with L_2- and, in particular, with L_∞-norms, in that outliers among the elevation data caused by terrain artifacts such as rocks, single trees and bushes, as well as processing artifacts, distort the shape of the TIN surface – make it conform less to bare earth –, during the process of selecting elevations at critical points. In some approaches, the selection of a norm may also affect the construction of the TIN itself.

Those drawbacks of the commonly considered norms have led the authors to consider the

$$L_1 - norm$$

as a measure-of-fit for terrain surface approximation because it is more robust with respect to artifacts, and may even be used for their detection (see [4]). The L_1-norm is defined as the sum of the absolute values of the residuals. One might actually prefer the mean absolute deviations

$$\text{MAD} \;=\; \frac{1}{|R|}\sum_{p\in R}|r_p|.$$

Note that the elevation $z = z(x,y)$ of the TIN surface at any map location (x,y) is a linear function of the critical elevations z_s, $s \in S$. Consequently, the residuals r_p, $p \in R$ are also linear functions of the critical elevations. For this reason, the L_1-approximation problem is a linear programming problem, and the L_2-approximation problem is a linear least squares problem. The connection between L_1-approximation and linear programming is well established (e.g.[1]).

More precisely, $z = z(x,y)$ is a linear combination of three elevations

$$z_{s_1},\ z_{s_2},\ z_{s_3},$$

which belong to vertices s_1, s_2, s_3 of a TIN triangle containing location (x,y). The coefficients of that linear combination,

$$z(x,y) \;=\; \lambda_1 z_{s_1} + \lambda_2 z_{s_2} + \lambda_3 z_{s_3},$$

are the barycentric coordinates of location (x,y) with respect to $s_i = (x_{s_i}, y_{s_i}),\ i = 1,2,3$ and are defined by

$$\begin{aligned} x &= \lambda_1 x_{s_1} + \lambda_2 x_{s_2} + \lambda_3 x_{s_3} \\ y &= \lambda_1 y_{s_1} + \lambda_2 y_{s_2} + \lambda_3 y_{s_3} \\ 1 &= \lambda_1 + \lambda_2 + \lambda_3. \end{aligned}$$

The barycentric coordinates assume values between 0 and 1 since the location (x,y) lies in the triangle in question. The barycentric coordinates play a major role in approximation by triangulated surfaces.

While linear programming problems are well understood and excellent solution methods exist even for large numbers of variables, the L_1-approximation problem considered here typically exceeds the size that can be handled by those methods. Suboptimal solution methods were, therefore, developed. Straight L_2-approximation, however, can be solved fully optimally on an iterative basis described below.

One method of elevation optimization with respect to a specific measure-of-fit is

$$vertex - based\ elevation\ optimization.$$

With any critical point – vertex in the TIN –, we associate the union of all triangles in the triangulation which contain that vertex, calling it the *star* of that critical point. The star is a simply-connected polygon bounded by the edges between adjacent "neighbor" vertices of the critical point that centers the star.

We will require the triangulation to be

$$star - complete,$$

that is, every star in the triangulation contains at least one elevation location $(\hat{x}_p, \hat{y}_p)$, $p \in R$ in its interior. Without this condition, the elevation at the center of an empty star could be chosen arbitrarily without affecting any of the norm-based measures-of-fit considered here. In that sense, the approximation problem would be ill-defined.

A single step of a vertex-based iteration then consists of adjusting the elevation at the center of a star so as to minimize the norm of choice of the residuals in the interior of the star while keeping the elevations at all other vertices fixed. Each vertex elevation is adjusted in turn in a pass through all critical points. Such a pass is then repeated iteratively until the resulting improvements become acceptably small.

For L_1- and L_2-norms, the basic vertex-based optimization step can be executed readily in closed form. In the L_2 case, a convex quadratic of one variable has to be minimized. In the L_1 case, minimization of a piecewise linear convex function of one variable is required. This task is equivalent to determining a suitably weighted median of the breakpoints.

In the case of the L_2-norm, furthermore, it can be shown that the vertex-based iterative procedure converges to the true optimum (e.g. [2]). That, however, does not hold in the case of the L_1-norm. For this reason, two additional algorithms for L_1-optimization have been developed in order to improve the degree of optimality that can be achieved.

The first of those algorithms is

$$edge - based\ elevation\ optimization,$$

where the basic step is to L_1-optimize the elevations at the two end-vertices of a triangulation edge simultaneously while keeping the elevations at all other vertices of the triangulation fixed. That local optimization problem turns out to be a special linear programming problem, and a special algorithm has been designed to solve it.

The second algorithm utilizes the fact that, for $p \in R$,

$$\frac{r_p^2}{|r_p|} = \frac{(\hat{z}_p - \bar{z}_p)^2}{|\hat{z}_p - \bar{z}_p|} = |\hat{z}_p - \bar{z}_p| = |r_p|,$$

to formulate a

$$residual - weighted\ L_2 - procedure$$

for L_1-approximation. Roughly speaking, we start the L_2-approximation with equal weights, and adjust these weights as we proceed on the basis of observed residuals. The challenge here is to handle the cases of very small residuals which result in very large weights. There is also the question how far to iterate the L_2 procedure before updating the weights. Our current approach is to adjust weights after one L_2-pass through the vertices of the triangulation.

Both algorithms are still suboptimal and, as the numerical results presented in this report show, best results so far have been achieved by alternating both algorithms. We know that these results are suboptimal, because it is well known that, for any fully optimal solution of any L_1 approximation problem, the number of

$$sharp$$

elevation location points $(\hat{x}_p, \hat{y}_p) \in R$, that is, points for which $\hat{z}_p = \bar{z}_p$, and which are thus represented exactly by the TIN surface, essentially equals the number of degrees of freedom provided by the optimization parameters. Here that number is the number $|S \setminus S'|$ of nonfixed critical points. For our best results, the number of sharp points does not quite reach that goal.

The algorithms and routines described here were devised and developed in the context of particular applications to terrain modeling. For these applications, the bulk of the elevation data points were provided as regular grid points with square unit cells covering a rectangular map exactly. A set S' of optional additional elevation points – "*feature points*" – were also specified with the proviso that they be included into the set of critical points S and that their given elevations are not to be changed. If such a feature point happens to be located on the grid, then its given elevation takes precedence over the specified grid elevation. Furthermore, all critical points other than feature points had to be grid points. This ensured that all corresponding stars were complete, as defined above, because their respective center vertices are elevation points. In addition, the specified grid elevations at such critical points provided good initial values for iterations.

For the purpose of identifying terrain artifacts, it is useful to generalize L_1 approximation by scaling residuals differently depending on their sign. Our software thus includes an option to assign scale factors $u > 0$ if $r_p > 0$ and $v > 0$ if $r_p < 0$. We call that option

$$sign - oriented\ scaling.$$

If, for instance, the object is to identify artifacts above bare earth, then one would scale positive residuals lower than negative residuals. That would reduce the influence of artifacts above bare earth on the construction of the TIN surface. Those artifacts will then show up more clearly if the data are compared to the latter. Sign-oriented scaling can be specified for any norm.

The algorithms reported here are as follows. For quantities not defined earlier, we ask the reader to refer to the detailed descriptions provided later.

$$\mathrm{L2FIT}(R, S, S', T, F, Z, \hat{Z}, n, tol)$$

is a vertex-based iterative procedure for L_2-approximation. It is known to actually converge to the optimum. It has been in use as a post-optimization option in the prototype NIST TIN package for several years.

$$\text{L1FIT1}(R, S, S', T, F, Z, \hat{Z}, n, tol, npmax, u, v)$$

is a vertex-based iterative procedure for L_1-approximation featuring sign-oriented scaling. It is suboptimal. It was the first L_1 procedure to be considered.

$$\text{L1FIT2}(R, S, S', T, F, E, Z, \hat{Z}, n, tol, npmax, u, v)$$

is an edge-based iterative procedure for L_1-approximation featuring sign-oriented scaling. It is suboptimal but achieves results that are closer to optimality than the vertex-based version. It was the second L_1 procedure to be considered.

$$\text{L1FIT3}(R, S, S', T, F, Z, \hat{Z}, n, tol, npmax, u, v, wscl)$$

aims to achieve an L_1 optimum by iterating a residual-weighted vertex-based L_2 procedure. Sign-oriented scaling is included. The procedure is still suboptimal but by itself achieves a higher degree of optimality than the previous procedure.

Comparative numerical results are reported for L_1-approximation of a representative set of terrain elevation data without feature points and without sign-oriented scaling. As was mentioned earlier, best results were found when procedures L1FIT2 and L1FIT3 were alternated. We feel that those results are close to optimal, particularly, because the number 19558 of sharp points approaches the number 20000 of critical points.

2. Numerical results

The procedures presented here for computing approximate L_1 solutions have been implemented at NIST. They were applied to a terrain data set consisting of a regular grid of 229,761 (521×441) points with square $25 \times 25m$ unit cells covering a rectangular $11 \times 13km$ map. A triangulation of 20,000 critical points – selected from among the grid points and covering the map – was also given. None of the critical points were feature points with fixed elevations, so $S' = \emptyset$. Sign-oriented scaling was not used, so $u = v = 1$.

Table 2.1, Table 2.2, and Table 2.3 show some numerical results obtained when the vertex-based procedure, the edge-based procedure, and the L_2 with weights procedure, respectively, were used on the test problem. Table 2.4 shows some numerical results obtained when the edge-based procedure was used on the test problem after 300 passes through the vertices by the L_2 with weights procedure. In all cases, where it applies, variable tol was set to 6.058e-04, and variable $wscl$ to 1.05. Given an integer i, $1 \leq i \leq n$, where it applies, $E(i)$ was set to the

pass	*CPUsec*	*objective*	*residual tolerance*	*MAX residual*	*# sharp points*
1	121.9	2508140.342		158.750	14976
2	121.6	2466989.055		158.750	15015
5	121.2	2461191.750		155.750	17990
10	121.2	2460970.594		155.750	18604
20	121.1	2460872.057		155.750	18638
30	121.2	2460863.037		155.750	18639
50	121.2	2460857.512		155.750	18637
75	121.9	2460852.957		155.750	18639
100	121.8	2460850.886		155.750	18640
125	121.8	2460848.991		155.750	18644
150	121.8	2460847.192		155.750	18649

Table 2.1: Results for vertex-based procedure L1FIT1

point in S which was the first point in the data structure of the implementation that was an endpoint of an edge in T for which $F(i)$ was the other endpoint.

In all tables, the first column gives the sequence number of the particular pass. The second column displays its CPU running time in seconds. The third column contains the value of the objective function at the completion of the pass. Where applicable (Table 2.3), the fourth column contains the value of the residual tolerance used during the pass. The fifth column contains the maximum of the absolute values of the residuals at the completion of the pass. The last column contains the number of sharp grid points at the completion of the pass, i.e. the number of residuals whose absolute values were less than 0.01 at the completion of the pass.

We note that the numerical results in Table 2.1, Table 2.2, and Table 2.3 seem to indicate that the edge-based procedure works better than the vertex-based procedure, and that the L_2 with weights procedure works better than the edge-based procedure. However, the results in Table 2.4 seem to indicate that using the edge-based procedure after the execution of the L_2 with weights procedure is a better approach.

pass	*CPUsec*	*objective*	*residual tolerance*	*MAX residual*	*# sharp points*
1	558.4	2476235.062		158.750	15552
2	566.4	2460725.514		155.750	16867
5	555.0	2457409.175		155.750	18986
10	554.6	2457036.057		155.750	19573
20	558.8	2457001.234		155.750	19683
30	554.9	2456996.365		155.750	19649
50	561.0	2456984.264		155.750	19595
75	554.5	2456977.839		155.750	19552
100	551.4	2456973.258		155.750	19536
125	558.9	2456969.810		155.750	19505
150	552.5	2456966.380		155.750	19494

Table 2.2: Results for edge-based procedure L1FIT2

pass	*CPUsec*	*objective*	*residual tolerance*	*MAX residual*	*# sharp points*
1	121.5	2588582.203	146.825	121.619	268
5	121.5	2557018.183	120.793	126.196	280
10	122.3	2556963.799	94.645	129.640	278
20	122.1	2554668.834	58.104	135.601	282
30	122.1	2542027.536	35.671	137.130	279
50	122.0	2499697.025	13.444	142.532	262
75	122.0	2469877.300	3.970	150.890	321
100	121.9	2460298.750	1.172	154.523	446
125	121.9	2457317.776	0.346	155.564	900
150	121.9	2456400.229	1.022e-01	155.702	2117
175	121.9	2456122.316	3.019e-02	155.737	6701
200	121.9	2456038.092	8.915e-03	155.745	18046
225	121.9	2456015.632	2.633e-03	155.749	18589
250	121.9	2456013.435	7.774e-04	155.749	18631
275	123.1	2456013.310	6.058e-04	155.749	18603
300	122.2	2456013.128	6.058e-04	155.749	18584

Table 2.3: Results for L_2-based procedure L1FIT3

pass	*CPUsec*	*objective*	*residual tolerance*	*MAX residual*	*# sharp points*
1	545.8	2455998.827		155.748	19161
2	553.8	2455994.372		155.748	19318
5	537.3	2455988.608		155.750	19545
10	539.4	2455987.781		155.750	19580
20	539.8	2455987.481		155.750	19576
30	539.9	2455987.377		155.750	19578
50	547.7	2455987.331		155.750	19570
70	540.8	2455987.276		155.750	19569
150	539.7	2455987.265		155.750	19558

Table 2.4: Results for edge-based L1FIT2 after L1FIT3

3. The 2-dimensional L_1 fitting problem

Let R be a regular grid with rectangular cell units covering a rectangular map in the plane. Let S be a finite set of critical points in the map that contains the four corners of the map, and let T be a triangulation for S. Let S' be the set of feature points, that is, of critical points with prescribed fixed elevations. For each grid point p in R assume that an elevation $\hat{z}_p$ is associated with p. Given p in R, let t be a triangle in T that contains p, and define $s_i(p)$, $i = 1, 2, 3$, as the points in S that are the vertices of t. Given p in R, define $\lambda_i(p)$, $i = 1, 2, 3$, as the barycentric coordinates of p relative to T so that p equals $\sum_{i=1}^{3} \lambda_i(p) s_i(p)$ and $0 \leq \lambda_i(p) \leq 1$ for each i, $i = 1, 2, 3$. For each point s in S' assume that an elevation z_s is associated with s. We search then for a set of elevations $\{z_s : s \in S \setminus S'\}$ so that the following function is minimized:

$$\sum_{p \in R} |\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)}|.$$

More generally, given u, $v > 0$, and if for each p in R we let $\overline{z}_p$ equal $\sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)}$, we search then for a set of elevations $\{z_s : s \in S \setminus S'\}$ so that the following function is minimized:

$$u \sum_{p \in R^+} (\hat{z}_p - \overline{z}_p) - v \sum_{p \in R^-} (\hat{z}_p - \overline{z}_p), \tag{3.1}$$

where $R^+ = \{p \in R : \hat{z}_p - \overline{z}_p \geq 0\}$ and $R^- = \{p \in R : \hat{z}_p - \overline{z}_p < 0\}$.

4. The 2-dimensional L_2 fitting problem

Of related interest is the quadratic version of the problem. Here we search for a set of elevations $\{z_s : s \in S \setminus S'\}$ so that the following function is minimized:

$$\sum_{p \in R} (\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)})^2. \tag{4.1}$$

Since this function is quadratic it has a unique minimum point which is the point at which its gradient equals zero [2].

Given a point s in S we define the *star* of s as the union of the triangles in T that have s as a vertex, and denote by R_s the set of grid points that it contains.

For a fixed point s in S, we assume without any loss of generality that for each point p in R_s, $s_1(p)$ equals s. Under this assumption the partial derivative of Function 4.1 with respect to z_s is

$$2 \sum_{p \in R_s} (\hat{z}_p - \lambda_1(p) z_s - \lambda_2(p) z_{s_2(p)} - \lambda_3(p) z_{s_3(p)})(-\lambda_1(p)).$$

Setting this partial derivative to zero we are then able to solve for z_s and obtain

$$z_s = (\sum_{p \in R_s} (\hat{z}_p - \lambda_2(p) z_{s_2(p)} - \lambda_3(p) z_{s_3(p)}) \lambda_1(p)) / \sum_{p \in R_s} (\lambda_1(p))^2.$$

The following iterative procedure, which is based on the above formula, can be used for computing the minimum point of Function 4.1. That it terminates in a finite number of steps follows again from the fact that Function 4.1 is quadratic [2]. Here we assume that there are n points in S, n a positive integer, and that the n points are ordered in some fashion. A one-to-one function F from $\{1, \ldots, n\}$ onto S is defined by setting $F(i)$ to the i^{th} point in S for each i, $i = 1, \ldots, n$. A real-valued elevation function Z with domain S is defined by setting $Z(s)$ to z_s for each s in S', and to $\hat{z}_s$ otherwise. At the end of the execution of the procedure Z will contain the minimum point. Another real-valued elevation function $\hat{Z}$ with domain R is defined by setting $\hat{Z}(p)$ to $\hat{z}_p$ for each p in R. The procedure, called L2FIT, requires an input variable called *tol* which must be set to a positive number and is used as a tolerance or adjustment constant during the execution of the procedure. Two procedures, called STARGRIDPOINTS and BARYCENTRIC are used as primitives in L2FIT. Given integer i, $1 \leq i \leq n$, with $F(i) \in S \setminus S'$, for some positive integer m, STARGRIDPOINTS locates points $G(j)$, $j = 1, \ldots, m$, that are the points in $R_{F(i)}$. Given an integer j, $1 \leq j \leq m$, BARYCENTRIC locates a triangle in T with $F(i)$ as a vertex that contains $G(j)$, identifies the other two vertices Q_2, Q_3 of the triangle, and computes the barycentric coordinates W_1, W_2, W_3 of $G(j)$ relative to the triangle so that $G(j)$ equals $W_1 \cdot F(i) + W_2 \cdot Q_2 + W_3 \cdot Q_3$. The outline of the procedure follows.

```
procedure L2FIT(R, S, S', T, F, Z, Ẑ, n, tol)
    begin
    flag := 0;
    while (flag = 0) do
        begin
        flag := 1;
        for i := 1 until n do
            begin
            if (F(i) ∈ S \ S') then
                begin
                (G, m):=STARGRIDPOINTS(F(i), R, T);
                numerator := 0.0; denominator := 0.0;
                for j := 1 until m do
                    begin
                    (Q_2, Q_3, W_1, W_2, W_3):=BARYCENTRIC(T, G(j), F(i));
                    numerator := numerator+
                    (Ẑ(G(j)) − W_2 · Z(Q_2) − W_3 · Z(Q_3)) · W_1;
                    denominator := denominator + (W_1)^2
                    end
                znew := numerator/denominator;
                if (| Z(F(i)) − znew |> tol) then
                    begin
                    flag := 0;
                    Z(F(i)) := znew
    endendendendend
```

5. Vertex-based iterative procedure for L_1 approximation

An approach similar to the one in the previous section can be used for obtaining approximate solutions to L_1 fitting problems. Given s in $S \setminus S'$, for each point $\hat{s}$ in $S \setminus \{s\}$ we assume that an elevation $z_{\hat{s}}$ is associated with $\hat{s}$. We search then for an elevation z_s so that the function

$$\sum_{p \in R} |\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)}|$$

is minimized.

More generally, given u, $v > 0$, we search for an elevation z_s so that the function

$$u \sum_{p \in R^+} (\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)}) - v \sum_{p \in R^-} (\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)})$$

is minimized, where
$R^+ = \{p \in R : \hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)} \geq 0\}$ and $R^- = \{p \in R : \hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)} < 0\}$.

Assume without any loss of generality that for each point p in R_s, $s_1(p)$ equals s, and for some positive integer m, let p_j, $j = 1, \ldots, m$ be the points in R_s. The above problem is then equivalent to searching for an elevation z_s so that the function

$$\sum_{j=1}^{m} w_j |\hat{z}_{p_j} - \lambda_1(p_j) z_s - \lambda_2(p_j) z_{s_2(p_j)} - \lambda_3(p_j) z_{s_3(p_j)}| \tag{5.1}$$

is minimized, where w_j equals u if $\hat{z}_{p_j} - \lambda_1(p_j) z_s - \lambda_2(p_j) z_{s_2(p_j)} - \lambda_3(p_j) z_{s_3(p_j)} \geq 0$, and v otherwise, for each j, $j = 1, \ldots, m$. This function is convex, continuous, piece-wise linear, and has at least one minimum point.

We show how a minimum point of Function 5.1 can be found. For each j, $j = 1, \ldots, m$, assume without any loss of generality that $\lambda_1(p_j)$ is not equal to zero, and define c_j as follows:

$$c_j \equiv (\hat{z}_{p_j} - \lambda_2(p_j) z_{s_2(p_j)} - \lambda_3(p_j) z_{s_3(p_j)}) / \lambda_1(p_j).$$

Assume without any loss of generality that $m > 1$, and that for each j, $j = 1, \ldots, m-1$, $c_j < c_{j+1}$. For each j, $j = 1, \ldots, m-1$, let I_j be the interval $[c_j, c_{j+1}]$. Define I_0 as the interval $(-\infty, c_1]$, and I_m as the interval $[c_m, \infty)$. In the interval I_0 the slope of Function 5.1 is $-u \sum_{k=1}^{m} \lambda_1(p_k)$, and in I_m it is $v \sum_{k=1}^{m} \lambda_1(p_k)$. For each j, $j = 1, \ldots, m-1$, in the interval I_j the slope is $v \sum_{k=1}^{j} \lambda_1(p_k) - u \sum_{k=j+1}^{m} \lambda_1(p_k)$. Clearly, the slope must change signs at an endpoint of one the intervals, and this is then a minimum point of Function 5.1.

The following iterative procedure for attempting to minimize approximately Function 3.1, is based on the ideas presented above for minimizing Function 5.1. The procedure, called L1FIT1, requires an input variable called *npmax* which must be set to a positive integer and is used as the maximum number of passes allowed through the vertices, thus guaranteeing that the execution of the procedure terminates. Besides STARGRIDPOINTS and BARYCENTRIC, a procedure called INCREASORT is used as a primitive in L1FIT1. Given an integer mw, $mw > 1$, a one-to-one function M from $\{1, \ldots, mw\}$ onto a subset I of the positive integers, and a function C from I into the set of real numbers, INCREASORT rearranges M so that for each j, $j = 1, \ldots, mw-1$, $C(M(j)) \leq C(M(j+1))$. The outline of the procedure follows.

```
procedure L1FIT1(R, S, S', T, F, Z, Ẑ, n, tol, npmax, u, v)
  begin
  flag := 0; npass := 0; w := u + v;
  while (flag = 0 and npass < npmax) do
    begin
    flag := 1; npass := npass + 1;
    for i := 1 until n do
      begin
      if (F(i) ∈ S \ S') then
        begin
        (G, m):=STARGRIDPOINTS(F(i), R, T);
        mw := 0; slope := 0.0;
        for j := 1 until m do
          begin
          (Q_2, Q_3, W_1, W_2, W_3):=BARYCENTRIC(T, G(j), F(i));
          if (W_1 > 0.0) then
            begin
            mw := mw + 1;
            C(mw) := (Ẑ(G(j)) − W_2 · Z(Q_2) − W_3 · Z(Q_3))/W_1;
            D(mw) := W_1;
            M(mw) := mw;
            slope := slope − u · W_1
          endend
        if (mw > 1) then INCREASORT(M, C, mw);
        j := 0;
        while (slope < 0.0) do
          begin
          j := j + 1;
          slope := slope + w · D(M(j))
          end
        znew := C(M(j));
        if (| Z(F(i)) − znew |> tol) then
          begin
          flag := 0;
          Z(F(i)) := znew
  endendendendend
```

6. Edge-based iterative procedure for L_1 approximation

Given s, s' in $S \setminus S'$, s and s' the endpoints of an edge of a triangle in T, and assuming that an elevation $z_{\hat{s}}$ is associated with each point $\hat{s}$ in $S \setminus \{s, s'\}$, we now search for elevations z_s and $z_{s'}$ so that the function

$$u \sum_{p \in R^+} (\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)}) - v \sum_{p \in R^-} (\hat{z}_p - \sum_{i=1}^{3} \lambda_i(p) z_{s_i(p)})$$

is minimized. Here R^+, R^-, u, v are as in the previous section.

Assume without any loss of generality that for each point p in R_s, $s_1(p)$ equals s, that for each point p in $R_{s'}$, $s_2(p)$ equals s', and that the sets $R_s \setminus R_{s'}$ and $R_{s'} \setminus R_s$ are nonempty. For positive integers m_1, m_2, m_3, let p_{j1}, $j = 1, \ldots, m_1$ be the points in $R_s \cap R_{s'}$, let p_{j2}, $j = 1, \ldots, m_2$ be the points in $R_s \setminus R_{s'}$, and let p_{j3}, $j = 1, \ldots, m_3$ be the points in $R_{s'} \setminus R_s$. The above problem is then equivalent to searching for elevations z_s and $z_{s'}$ so that the following function is minimized:

$$\begin{aligned}
(6.1) \qquad & \sum_{j=1}^{m_1} w_{1j} |\hat{z}_{p_{j1}} - \lambda_1(p_{j1}) z_s - \lambda_2(p_{j1}) z_{s'} - \lambda_3(p_{j1}) z_{s_3(p_{j1})}| + \\
& \sum_{j=1}^{m_2} w_{2j} |\hat{z}_{p_{j2}} - \lambda_1(p_{j2}) z_s - \lambda_2(p_{j2}) z_{s_2(p_{j2})} - \lambda_3(p_{j2}) z_{s_3(p_{j2})}| + \\
& \sum_{j=1}^{m_3} w_{3j} |\hat{z}_{p_{j3}} - \lambda_1(p_{j3}) z_{s_1(p_{j3})} - \lambda_2(p_{j3}) z_{s'} - \lambda_3(p_{j3}) z_{s_3(p_{j3})}|.
\end{aligned}$$

Here w_{kj} equals u if $\hat{z}_{p_{jk}} - \lambda_1(p_{jk}) z_{s_1(p_{jk})} - \lambda_2(p_{jk}) z_{s_2(p_{jk})} - \lambda_3(p_{jk}) z_{s_3(p_{jk})} \geq 0$, and v otherwise, for each k, j, $k = 1, 2, 3$, $j = 1, \ldots, m_k$. In what follows we think of Function 6.1 as a two-variable real-valued function whose first variable is $z_{s'}$ and whose second variable is z_s. We denote it by f so that if z_s and $z_{s'}$ are set to values, $f(z_{s'}, z_s)$ then denotes the corresponding value of the function. Finally, we denote by $H(s', s)$ the set of triplets of the form $(z_{s'}, z_s, f(z_{s'}, z_s))$. $H(s', s)$ is a convex, continuous, piece-wise linear surface in 3-dimensional space, and z_s, $z_{s'}$ exist at which f achieves its minimum value, and $(z_{s'}, z_s, f(z_{s'}, z_s))$ is a vertex of $H(s', s)$.

For each j, $j = 1, \ldots, m_1$, assume without any loss of generality that $\lambda_1(p_{j1})$ is not equal to zero, and define c_j, b_j, d_j as follows:

$$\begin{aligned}
c_j &\equiv (\hat{z}_{p_{j1}} - \lambda_3(p_{j1}) z_{s_3(p_{j1})}) / \lambda_1(p_{j1}), \\
b_j &\equiv -\lambda_2(p_{j1}) / \lambda_1(p_{j1}), \\
d_j &\equiv \lambda_1(p_{j1}).
\end{aligned}$$

For each j, $j = 1, \ldots, m_2$, assume without any loss of generality that $\lambda_1(p_{j2})$ is not equal to zero, and define c_{j+m_1}, b_{j+m_1}, d_{j+m_1} as follows:

$$c_{j+m_1} \equiv (\hat{z}_{p_{j2}} - \lambda_2(p_{j2}) z_{s_2(p_{j2})} - \lambda_3(p_{j2}) z_{s_3(p_{j2})}) / \lambda_1(p_{j2}),$$

$$b_{j+m_1} \equiv 0.0,$$

$$d_{j+m_1} \equiv \lambda_1(p_{j2}).$$

For each j, $j = 1, \ldots, m_3$, assume without any loss of generality that $\lambda_2(p_{j3})$ is not equal to zero, and define c'_j, d'_j as follows:

$$c'_j \equiv (\hat{z}_{p_{j3}} - \lambda_1(p_{j3}) z_{s_1(p_{j3})} - \lambda_3(p_{j3}) z_{s_3(p_{j3})}) / \lambda_2(p_{j3}),$$

$$d'_j \equiv \lambda_2(p_{j3}).$$

Setting mw to $m_1 + m_2$, and mv to m_3, for each j, $j = 1, \ldots, mw$, denote by l_j the straight line in the $z_{s'} - z_s$ plane defined by the linear equation $z_s = b_j z_{s'} + c_j$, and for each j, $j = 1, \ldots, mv$, denote by l'_j the straight line in the same plane defined by the linear equation $z'_s = c'_j$. Clearly, from the polyhedral nature of $H(s', s)$ it follows that the perpendicular projections onto the $z_{s'} - z_s$ plane of the 1-dimensional faces of $H(s', s)$ are contained in these lines, and those of the vertices are points where these lines intersect. Thus, the problem of minimizing Function 6.1 reduces to that of finding among the points at which at least two of the lines l_j, $j = 1, \ldots, mw$, l'_j, $j = 1, \ldots, mv$, intersect, one at which the function achieves its smallest value.

Given an integer *mopt*, $1 \leq mopt \leq mw$, we show how a minimum point of f, i. e. Function 6.1, restricted to line l_{mopt} can be found in a manner similar to the one used in the previous section for minimizing Function 5.1. Without any loss of generality we assume that at any point in the $z_{s'} - z_s$ plane at most two of the lines l_j, $j = 1, \ldots, mw$, l'_j, $j = 1, \ldots, mv$, intersect. Thus, if two of these lines intersect, and Function 6.1 restricted to the two lines achieves its smallest value at the point at which the two lines intersect, then from the convexity and the polyhedral nature of $H(s', s)$ it follows that among the points at which any two of the lines intersect, at this point the function achieves its smallest value. Therefore, as pointed above, this then implies that at this point the function also achieves its minimum value without any restrictions on z_s and $z_{s'}$.

For each j, $j = 1, \ldots, mw$, $j \neq mopt$, assume without any loss of generality that l_j is not parallel to l_{mopt}, and define a_j as follows:

$$a_j \equiv (c_j - c_{mopt}) / (b_{mopt} - b_j).$$

Clearly, a_j is the value of $z_{s'}$ at which l_j intersects l_{mopt}. Assume without any loss of generality that $mw > 2$, that *mopt* equals mw, and that for each j, $j = 1, \ldots, mw - 2$,

$a_j < a_{j+1}$. For the sake of simplicity and only for the current discussion, assume that mv equals zero, and for each j, $j = 1, \ldots, mw-2$, let I_j be the interval $[a_j, a_{j+1}]$ in the $z_{s'}$ axis of the $z_{s'} - z_s$ plane. Define I_0 as the interval $(-\infty, a_1]$, and I_{mw-1} as the interval $[a_{mw-1}, \infty)$ in the same axis. Define a one-variable real-valued function f' by setting $f'(z_{s'})$ to $f(z_{s'}, b_{mopt}z_{s'} + c_{mopt})$ for each value of $z_{s'}$. f' is convex, continuous, piece-wise linear, and has at least one minimum point. For each j, set u_j to u if $b_{mopt} - b_j$ is positive, and to $-v$ otherwise, and set v_j to v if $b_{mopt} - b_j$ is positive, and to $-u$ otherwise. With this terminology, in the interval I_0 the slope of f' is $-\sum_{k=1}^{mw-1} u_j(b_{mopt} - b_j)d_j$, and in I_{mw-1} it is $\sum_{k=1}^{mw-1} v_j(b_{mopt} - b_j)d_j$. For each j, $j = 1, \ldots, mw-2$, in the interval I_j the slope is $\sum_{k=1}^{j} v_j(b_{mopt} - b_j)d_j - \sum_{k=j+1}^{mw-1} u_j(b_{mopt} - b_j)d_j$. Clearly, the slope of f' must change signs at an endpoint of one of the intervals, and this is then a minimum point of f'. Thus, given $z_{s'}$ at which f' achieves its minimum value it follows that Function 6.1 restricted to l_{mopt} has a minimum point at $(z_{s'}, b_{mopt}z_{s'} + c_{mopt})$.

A minimum point of Function 6.1 without any restrictions on z_s and $z_{s'}$ can then be found through a two-step iterative procedure as follows:

Step 1. Mark each one of the lines l_j, $j = 1, \ldots, mw$, l'_j, $j = 1, \ldots, mv$ as *not optimized.* Select arbitrarily one of these lines and call it the *current line.*

Step 2. Find a minimum point of Function 6.1 restricted to the current line, and call this point the *current minimum point.* Mark the current line as *optimized*, identify the one line among the lines l_j, $j = 1, \ldots, mw$, l'_j, $j = 1, \ldots, mv$, that intersects the current line at the current minimum point, and call this line the *next line.* If the next line has been marked as optimized then the current minimum point is a minimum point of Function 6.1 without restrictions and the procedure terminates. Else call the next line the current line and go back to the beginning of Step 2.

The following iterative procedure for attempting to minimize approximately Function 3.1, is based on the ideas presented above for minimizing Function 6.1. Here we assume that for each i, $i = 1, \ldots, n$, an edge of a triangle in T is selected so that $F(i)$ is an endpoint of this edge. A function E from $\{1, \ldots, n\}$ into S is defined by setting for each i, $i = 1, \ldots, n$, $E(i)$ to the point in S which is the other endpoint of the edge selected with $F(i)$ as an endpoint. Besides INCREASORT, the procedure, called L1FIT2, requires primitives STARGRIDPOINTS1, STARGRIDPOINTS2, STARGRIDPOINTS3 as well as BARYCENTRIC1, BARYCENTRIC2, and BARYCENTRIC3. Given integer i, $1 \le i \le n$, with $F(i) \in S \setminus S'$, for some positive integer m_1, STARGRIDPOINTS1 locates points $G_1(j)$, $j = 1, \ldots, m_1$, that are the points in $R_{F(i)} \cap R_{E(i)}$. Given that $R_{F(i)} \setminus R_{E(i)}$ is not empty, for some positive integer m_2, STARGRIDPOINTS2 locates points $G_2(j)$, $j = 1, \ldots, m_2$, that are the points in $R_{F(i)} \setminus R_{E(i)}$. It sets m_2 to zero if $R_{F(i)} \setminus R_{E(i)}$ is empty. Given that $R_{E(i)} \setminus R_{F(i)}$ is not empty, for some positive integer m_3, STARGRIDPOINTS3 locates points $G_3(j)$, $j = 1, \ldots, m_3$, that are the points in $R_{E(i)} \setminus R_{F(i)}$. It sets m_3 to zero if $R_{E(i)} \setminus R_{F(i)}$ is empty. Given an integer j, $1 \leq j \leq m_1$, BARYCENTRIC1 locates a triangle in T with $F(i)$ and $E(i)$ as vertices that contains $G_1(j)$, identifies the third vertex Q_3 of the triangle, and computes the barycentric coordinates W_1, W_2, W_3 of $G_1(j)$ relative to the triangle so that $G_1(j)$ equals $W_1 \cdot F(i) + W_2 \cdot E(i) + W_3 \cdot Q_3$. Assuming m_2 is positive, given an integer j, $1 \leq j \leq m_2$, BARYCENTRIC2 locates a triangle in T with $F(i)$ as a vertex that contains $G_2(j)$, identifies the other two vertices Q_2, Q_3 of the triangle, and computes the barycentric coordinates W_1, W_2, W_3 of $G_2(j)$ relative to the triangle so that $G_2(j)$ equals $W_1 \cdot F(i) + W_2 \cdot Q_2 + W_3 \cdot Q_3$. Finally, assuming m_3 is positive, given an integer j, $1 \leq j \leq m_3$, BARYCENTRIC3 locates a triangle in T with $E(i)$ as a vertex that contains $G_3(j)$, identifies the other two vertices Q_1, Q_3 of the triangle, and computes the barycentric coordinates W_1, W_2, W_3 of $G_3(j)$ relative to the triangle so that $G_3(j)$ equals $W_1 \cdot Q_1 + W_2 \cdot E(i) + W_3 \cdot Q_3$.

The outline of the procedure follows. In lines 6-41 of the procedure, the nonvertical straight lines (l_j, $j = 1, \ldots, mw$, above), and the vertical straight lines (l'_j, $j = 1, \ldots, mv$, above) associated with the current edge are identified. In lines 42-60 of the procedure the first straight line to be called the current line (for the current edge) is identified, and it is the one nonvertical line at which Function 6.1 (for the current edge) attains its minimum for $z_{s'}$ arbitrarily close to $-\infty$. In line 66 the $z_{s'}$ values where the vertical lines intersect the $z_{s'}$-axis are sorted in increasing order. The $z_{s'}$ values where the current line intersects nonvertical lines are computed in lines 72-82 of the procedure if the current line is nonvertical, and in lines 111-114 if it is vertical. These values are sorted in increasing order in line 84 if the current line is nonvertical, and in line 115 if it is vertical. In lines 87-99 if the current line is nonvertical, and in lines 117-120 if it is vertical, a straight line to be called the next line is identified which is either a nonvertical or a vertical line whose intersection with the current line is a minimum point of Function 6.1 restricted to the current line (to be called the current minimum point). The current line is marked as optimized in line 106 of the procedure if it is nonvertical and in line 109 if it is vertical. Given that the next line has already been marked as optimized, in lines 122-128 of the procedure if the current line is vertical, and in lines 129-134 if it is nonvertical, another nonvertical line, if any, to be called the current line is identified which is not marked as optimized and that contains the current minimum point. If no such a line exists then the current minimum point is a minimum point for Function 6.1 without restrictions and if necessary the values of z_s and $z_{s'}$ (for the current edge) are corrected in lines 135-140 of the procedure.

procedure L1FIT2$(R, S, S', T, F, E, Z, \hat{Z}, n, tol, npmax, u, v)$
    begin
1. $flag1 := 0$; $npass := 0$; $w := u + v$;
2.   **while** $(flag1 = 0$ and $npass < npmax)$ **do**
        begin

$flag1 := 1$; $npass := npass + 1$;
for $i := 1$ **until** n **do**
begin
if $(F(i) \in S \setminus S')$ **then**
begin
(G_1, m_1):=STARGRIDPOINTS1$(F(i), E(i), R, T)$;
(G_2, m_2):=STARGRIDPOINTS2$(F(i), E(i), R, T)$;
if $(E(i) \in S \setminus S')$ **then** (G_3, m_3):=STARGRIDPOINTS3$(F(i), E(i), R, T)$
else $m_3 := 0$;
$mw := 0$; $mv := 0$;
for $j := 1$ **until** m_1 **do**
begin
(Q_3, W_1, W_2, W_3):=BARYCENTRIC1$(T, G_1(j), F(i), E(i))$;
if $(W_1 > 0.0$ and $E(i) \in S \setminus S')$ **then**
begin
$mw := mw + 1$;
$C(mw) := (\hat{Z}(G_1(j)) - W_3 \cdot Z(Q_3))/W_1$;
$B(mw) := -W_2/W_1$;
$D(mw) := W_1$
end
elseif $(W_1 > 0.0)$ **then**
begin
$mw := mw + 1$;
$C(mw) := (\hat{Z}(G_1(j)) - W_2 \cdot Z(E(i)) - W_3 \cdot Z(Q_3))/W_1$;
$B(mw) := 0.0$;
$D(mw) := W_1$
end
elseif $(W_2 > 0.0$ and $E(i) \in S \setminus S')$ **then**
begin
$mv := mv + 1$;
$C'(mv) := (\hat{Z}(G_1(j)) - W_3 \cdot Z(Q_3))/W_2$;
$D'(mv) := W_2$
endend
if $(m_2 > 0)$ **then**
begin
for $j := 1$ **until** m_2 **do**
begin
$(Q_2, Q_3, W_1, W_2, W_3)$:=BARYCENTRIC2$(T, G_2(j), F(i))$;

if $(W_1 > 0.0)$ **then**
begin
$mw := mw + 1;$
$C(mw) := (\hat{Z}(G_2(j)) - W_2 \cdot Z(Q_2) - W_3 \cdot Z(Q_3))/W_1;$
$B(mw) := 0.0;$
$D(mw) := W_1$
endendend
if $(m_3 > 0)$ **then**
begin
for $j := 1$ **until** m_3 **do**
begin
$(Q_1, Q_3, W_1, W_2, W_3)$:=BARYCENTRIC3$(T, G_3(j), E(i))$;
if $(W_2 > 0.0)$ **then**
begin
$mv := mv + 1;$
$C'(mv) := (\hat{Z}(G_3(j)) - W_1 \cdot Z(Q_1) - W_3 \cdot Z(Q_3))/W_2;$
$D'(mv) := W_2$
endendend
$slopew := 0.0;$
for $j := 1$ **until** mw **do**
begin
$M(j) := j;\ N(j) := 0;$
$slopew := slopew + v \cdot D(j)$
end
if $(mw > 1)$ **then**
begin
INCREASORT(M, B, mw);
for $j := 1$ **until** $mw - 1$ **do**
begin
$jj := j + 1;$
while $(jj \leq mw$ and $B(M(jj)) = B(M(j)))$ **do**
begin
if $(C(M(jj)) > C(M(j)))$ **then**
begin
$mt := M(j);$
$M(j) := M(jj);$
$M(jj) := mt$
end

```
                jj := jj + 1
            endendend
        j := 0;
        while (slopew > 0.0) do
            begin
            j := j + 1;
            slopew := slopew − w · D(M(j))
            end
        mopt := M(j);
        slopev := 0.0;
        if (mv > 0) then
            begin
            for j := 1 until mv do
                begin
                M'(j) := j; N'(j) := 0;
                slopev := slopev + u · D'(j)
            endend
        if (mv > 1) then INCREASORT(M', C', mv);
        znwa := C(mopt); znwb := Z(E(i));
        flag2 := 0;
        while (flag2 = 0) do
            begin
            flag2 := 1; mb := 0;
            mbc := 1; mvc := 1; slopew := slopev;
            for j := 1 until mw do
                begin
                b := B(mopt) − B(j);
                if (b > 0.0) then
                    slopew := slopew + u · b · D(j)
                elseif (b < 0.0) then
                    slopew := slopew − v · b · D(j);
                if (b ≠ 0.0) then
                    begin
                    c := C(j) − C(mopt);
                    mb := mb + 1;
                    M(mb) := j;
                    A(j) := c/b
                endend
            if (mb > 0 or mv > 0) then
```

```
            begin
84.         if (mb > 1) then INCREASORT(M, A, mb);
85.         M(mb + 1) := mw + 1; A(mw + 1) := 0.0;
86.         M′(mv + 1) := mv + 1; C′(mv + 1) := 0.0;
87.         while (slopew > 0.0) do
                begin
88.             if ((mbc ≤ mb and mvc ≤ mv and A(M(mbc)) < C′(M′(mvc)))
                    or (mbc ≤ mb and mvc > mv)) then
                    begin
89.                 flag3 := 1;
90.                 jb := M(mbc);
91.                 b := B(mopt) − B(jb);
92.                 if (b > 0.0) then
93.                     slopew := slopew − w · b · D(jb)
                    else
94.                     slopew := slopew + w · b · D(jb);
95.                 mbc := mbc + 1
                    end
                else
                    begin
96.                 flag3 := 2;
97.                 jv := M′(mvc);
98.                 slopew := slopew − w · D′(jv);
99.                 mvc := mvc + 1
                endend
100.        if (flag3 = 1) then
                begin
101.            znwb := A(jb);
102.            if (N(jb) ≠ 0) then flag3 := 3
                end
            else
                begin
103.            znwb := C′(jv);
104.            if (N′(jv) ≠ 0) then flag3 := 4
                end
105.        znwa := B(mopt) · znwb + C(mopt);
106.        N(mopt) := 1;
107.        if (flag3 = 1) then mopt := jb; flag2 := 0
108.        elseif (flag3 = 2) then
```

```
                  begin
109.              N'(jv) := 1;
110.              slopew := 0.0;
111.              for j := 1 until mw do
                      begin
112.                  M(j) := j;
113.                  slopew := slopew + u · D(j);
114.                  A(j) := B(j) · znwb + C(j)
                      end
115.              if (mw > 1) then INCREASORT(M, A, mw);
116.              j := 0;
117.              while (slopew > 0.0) do
                      begin
118.                  j := j + 1;
119.                  slopew := slopew − w · D(M(j))
                      end
120.              jb := M(j);
121.              if (N(jb) = 0) then mopt := jb; flag2 := 0
                  else
                      begin
122.                  znwa := A(jb);
123.                  b := 0.0; k := 1; jj := j + 1;
124.                  while (b = 0.0 and k = 1 and jj ≤ mw) do
                          begin
125.                      b := A(M(jj)) − znwa;
126.                      k := N(M(jj));
127.                      jj := jj + 1
                          end
128.                  if (b = 0.0 and k = 0) then mopt := M(jj − 1); flag2 := 0
                  endend
              else
                  begin
129.              b := 0.0; k := 1; jj := mbc;
130.              while (b = 0.0 and k = 1 and jj ≤ mb) do
                      begin
131.                  b := A(M(jj)) − znwb;
132.                  k := N(M(jj));
133.                  jj := jj + 1
                      end
```

134. **if** $(b = 0.0$ and $k = 0)$ **then** $mopt := M(jj-1)$; $flag2 := 0$
endendend
135. **if** $(\mid Z(F(i)) - znwa \mid > tol)$ **then**
begin
136. $flag1 := 0$;
137. $Z(F(i)) := znwa$
end
138. **if** $(\mid Z(E(i)) - znwb \mid > tol)$ **then**
begin
139. $flag1 := 0$;
140. $Z(E(i)) := znwb$
endendendendend

7. Residual-weighted L_2 iterative procedure for L_1 approximation

An approach based on the L_2 iterative procedure presented in Section 4 can be used for obtaining approximate solutions to L_1 fitting problems. This is based on the fact that given a real number r if r does not equal zero then $|r|$ equals $r^2/|r|$.

Given s in $S \setminus S'$, we assume that an elevation $\tilde{z}$ and a positive number $wcut$ are associated with s. In addition, for each point $\hat{s}$ in $S \setminus \{s\}$ we assume that an elevation $z_{\hat{s}}$ is associated with $\hat{s}$. Given u, $v > 0$, and assuming again without any loss of generality that for each point p in R_s, $s_1(p)$ equals s, and that for some positive integer m, p_j, $j = 1, \ldots, m$, are the points in R_s, we search then for an elevation z_s so that the function

$$\sum_{j=1}^{m} (w_j/\tilde{w}_j)(\hat{z}_{p_j} - \lambda_1(p_j)z_s - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)})^2 \tag{7.1}$$

is minimized. Here w_j equals u if $\hat{z}_{p_j} - \lambda_1(p_j)\tilde{z} - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)} \geq 0$, and v otherwise; and $\tilde{w}_j$ equals $|\hat{z}_{p_j} - \lambda_1(p_j)\tilde{z} - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)}|$ if this last number is greater than $wcut$, and $wcut$ otherwise. We notice that given j, $1 \leq j \leq m$, if $\tilde{w}_j$ is greater than $wcut$ and z_s is set to $\tilde{z}$ then the j^{th} term of Function 7.1 becomes $(w_j/\tilde{w}_j)\tilde{w}_j^2$ which reduces to $w_j\tilde{w}_j$, i. e. $w_j|\hat{z}_{p_j} - \lambda_1(p_j)\tilde{z} - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)}|$.

The derivative of Function 7.1 with respect to z_s is

$$2\sum_{j=1}^{m} (w_j/\tilde{w}_j)(\hat{z}_{p_j} - \lambda_1(p_j)z_s - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)})(-\lambda_1(p_j)).$$

Setting this derivative to zero we are then able to solve for z_s and thus obtain the value for

z_s at which Function 7.1 is minimized:

$$z_s = (\sum_{j=1}^{m}(w_j/\tilde{w}_j)(\hat{z}_{p_j} - \lambda_2(p_j)z_{s_2(p_j)} - \lambda_3(p_j)z_{s_3(p_j)})\lambda_1(p_j))/\sum_{j=1}^{m}(w_j/\tilde{w}_j)(\lambda_1(p_j))^2.$$

The following iterative procedure, which is based on the above formula, can be used for attempting to minimize approximately Function 3.1. Besides STARGRIDPOINTS and BARYCENTRIC, the procedure, called L1FIT3, requires a primitive called MAXRESIDUAL. It computes *wcut* which is the largest absolute value of a *residual*, i. e. the largest absolute value of a number of the form $\hat{Z}(p) - W_1 \cdot Z(Q_1) - W_2 \cdot Z(Q_2) - W_3 \cdot Z(Q_3)$, where p is in R, Q_1, Q_2, Q_3 are the vertices of a triangle in T that contains p, and W_1, W_2, W_3 are the barycentric coordinates of p relative to the triangle so that p equals $W_1 \cdot Q_1 + W_2 \cdot Q_2 + W_3 \cdot Q_3$. The variable *wcut*, which we call the *residual tolerance*, is used throughout the execution of the procedure for the purpose of avoiding or postponing division by absolute values of residuals that are less than its current value. This value is progressively reduced by dividing *wcut* before each pass through the vertices by a variable called *wscl*. This last variable must be set on input to a number that is greater than 1.0 and preferably less than or equal to 2.0. The outline of the procedure follows.

procedure L1FIT3($R, S, S', T, F, Z, \hat{Z}, n, tol, npmax, u, v, wscl$)
  begin
  $flag := 0$; $npass := 0$;
  $wcut$:=MAXRESIDUAL($R, Z, \hat{Z}$);
  while ($flag = 0$ and $npass < npmax$) **do**
    begin
    $flag := 1$; $npass := npass + 1$;
    $wcut := wcut/wscl$;
    if ($wcut < tol$) **then** $wcut := tol$;
    for $i := 1$ **until** n **do**
      begin
      if ($F(i) \in S \setminus S'$) **then**
        begin
        (G, m):=STARGRIDPOINTS($F(i), R, T$);
        $zold := Z(F(i))$;
        $numerator := 0.0$; $denominator := 0.0$;
        for $j := 1$ **until** m **do**
          begin
          (Q_2, Q_3, W_1, W_2, W_3):=BARYCENTRIC($T, G(j), F(i)$);
          $\tilde{W} := \hat{Z}(G(j)) - W_1 \cdot zold - W_2 \cdot Z(Q_2) - W_3 \cdot Z(Q_3)$;
          if ($\tilde{W} \geq 0.0$) **then** $W := u$ **else** $W := v$;

$\tilde{W} := |\tilde{W}|;$
if $(\tilde{W} < wcut)$ **then** $\tilde{W} := wcut;$
$numerator := numerator +$
$(W/\tilde{W}) \cdot (\hat{Z}(G(j)) - W_2 \cdot Z(Q_2) - W_3 \cdot Z(Q_3)) \cdot W_1;$
$denominator := denominator + (W/\tilde{W}) \cdot (W_1)^2$
end
$znew := numerator/denominator;$
if $(\mid zold - znew \mid > tol)$ **then**
begin
$flag := 0;$
$Z(F(i)) := znew$
endendendendend

Acknowledgement. Thanks to Douglas R. Caldwell, Topographic Engineering Center (TEC), for his interest and support.

BIBLIOGRAPHY

[1] I.Barrodale and F. D. K. Roberts, An Improved Algorithm for Discrete l_1 Approximation, SIAM J. Numer. ANAL. 10 (1993), pp 839-848.

[2] D. G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass. (1973).

[3] D. M. Mark, The History of Geographic Information Systems: Invention and Re-invention of Triangulated Irregular Networks (TINs), Proceedings GIS/LIS'97, in press. (http://www.geog.buffalo.edu/~dmark/)

[4] C. Witzgall and D. R. Caldwell, An L_1-Optimized TIN Approach to Locating Terrain Data Artifacts. (In preparation)