Matrix and Curve Mesh Interpolation

Andrew E. Long and Clifford A. Long

December 20, 2000

Abstract

Applications of mathematics and statistics often require fitting a smooth approximating function to a finite set of data points, or interpolating the data points by fitting the points exactly. In the event that interpolation of a grid of points is required, we recently proposed a method (which we call the *SVD method*[10]) for creating a function of two variables which interpolates the points. While there are many other procedures in the literature for interpolating and approximating 3-D data sets [8, 12, 13], the SVD method is straightforward and very easy to program. The process of interpolating a matrix we call *skinning* the matrix; we speak of creating a *skin* of the matrix.

Sometimes a mesh or grid of interlocking functions (which we call a *curve mesh*) may be considered for interpolation. Curve mesh interpolation, in which this grid of interlocking functions is interpolated by a single function of two variables, has been carried out in various ways ([2, 3, 4, 14]). We use a variant of the SVD method, based on the fact that the intersections of this grid of functions create a matrix (a *node matrix*). We simultaneously fit this matrix and the mesh which connects them. As might be expected, we speak of a skin of a mesh.

This method works for arbitrary curve meshes. We begin with the simple case of a square, invertible node-matrix meshes, and extend the technique to square, non-invertible meshes, and rectangular meshes. We present here two examples from the literature for comparison with another recent approach[2], as well as a famous example from a paper by Franke [5].

Keywords: Singular Value Decomposition, matrix interpolation, curve mesh interpolation, rank of a function

1 The Singular Value Decomposition

We begin by stating a theorem essential in all that follows:

Theorem 1.1 *(Singular Value Decomposition theorem) Let $A \neq 0$ be an $m \times n$ real matrix. There exist orthogonal matrices $U_{m\times m}$ and $V_{n\times n}$, and integer $r \leq min(m,n)$ (called the* **rank** *of A), such that*

$$U^T AV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}_{m\times n}$$

where $D = \begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_r \end{bmatrix}$ *is diagonal, with $d_i \geq d_{i+1} > 0 \, \forall \, i < r$.*

This says that any real matrix A can be decomposed as a matrix product,

$$A = U \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} V^T.$$

It is obvious that the last $n - r$ columns of U and the last $m - r$ columns of V do not contribute anything to the matrix product, and for that reason U is sometimes reduced to $U_{m\times r}$ and V to $V_{n\times r}$. Then we can write

$$A = UDV^T. \tag{1.1}$$

product is called the **singular value decomposition** (or *SVD*) of A. A proof can be found in many references, including [7]. The elements of the diagonal matrix D are the **singular values**, and the columns of U and V are the **singular vectors**.

The SVD has proven to be a powerful concept in many different areas of mathematics[1, 6, 9, 11], and has been essential to the development of the methods described herein.

2 Matrix Interpolation

Definition 2.1 *An* **interpolant** *is a function which passes through a set of points P. For example, given $P = \{(x_i, y_i) | i = 1, \cdots, n\}$, if $f(x_i) = y_i \forall i = 1, \cdots, n$ and f is a function, then f is an interpolant of P.*

For example, cubic splines are standard interpolants used in industry. To give a trivial example, every function interpolates its own graph.

Definition 2.2 *An* **interpolant of a vector** $\underline{u}$ *of length* n *is an interpolant of the points* $\{(k, u_k)|k = 1, \cdots, n\}$*, defined on* $[1, n]$ *(the indices are chosen as the x-coordinates by default); if a vector of abscissas* $\underline{x}$ *is specified,* $x_1 < x_2 < \ldots < x_n$*, then we will speak of an interpolant of* $\underline{u}$ *on abscissas* $\underline{x}$*, and mean an interpolant of the points* $\{(x_k, u_k)|k = 1, \cdots, n\}$*, defined on* $[x_1, x_n]$*.*

Theorem 2.1 *Given* A *as in Theorem 1.1, and two sets of vector interpolants* $\{u_k(x)|k = 1, \cdots, r\}$ *on abscissas* $\underline{x}$ *and* $\{v_k(y)|k = 1, \cdots, r\}$ *on abscissas* $\underline{y}$ *of the first* r *columns of* U *and* V*, respectively. Then*

$$S(x, y) = \sum_{k=1}^{r} u_k(x) d_k v_k(y) \tag{2.2}$$

is an interpolant of matrix A*, in the sense that* $S(x_i, y_j) = a_{ij}$*. We call this interpolant* S *of* A *a* **skin** *of* A*.*

Note: the graph of S is just a surface (the skin) which passes through the points (x_i, y_j, a_{ij}).

Proof: We claim that

$$S(x, y) = \sum_{k=1}^{n} u_k(x) d_k v_k(y)$$

"skins" the matrix, in the sense that $S(x_i, y_j) = a_{ij}$. The proof is simply a calculation: since $u_k(x)$ and $v_k(y)$ are vector interpolants,

$$S(x_i, y_j) = \sum_{k=1}^{n} u_k(x_i) d_k v_k(y_j) = \sum_{k=1}^{n} u_{ik} d_k v_{jk} = \sum_{k=1}^{n} u_{ik} d_k v_{jk} = a_{ij}$$

by equation 1.1. Denote the i^{th} row of a matrix M as $\underline{M_{i.}}$ (represented as a column vector). Then in matrix form,

$$S(x_i, y_j) = \underline{U_{i.}}^T D \underline{V_{j.}} = a_{ij}.$$

Q.E.D.

We call this the **SVD method** of matrix interpolation, since it relies on the SVD of the matrix. As a shorthand for this method, we may write

$$S(x, y) = \underline{U}(x)^T D\underline{V}(y),$$

where $\underline{U}(x)$ represents the vector of vector interpolants of the columns of matrix U (and similarly for $\underline{V}(y)$).

The SVD method is extraordinarily flexible: evey choice of vector interpolants $\underline{U}(x)$ and $\underline{V}(y)$ yields a matrix interpolant. The differentiability of the skin S is an immediate consequence of the differentiability of the singular vector interpolants, as we see from (2.2). For example, to obtain C^2 smoothness of the skinning surface, which is typically required for industrial milling processes, use twice-differentiable interpolants of the singular vectors.

Definition 2.3 *We define the* **rank** *of a function $F(x, y)$ relative to coordinates (x, y) as the smallest integer n such that F can be represented as*

$$F(x, y) = \sum_{i=1}^{n} \alpha_i(x)\beta_i(y).$$

Note that the rank of a function is not coordinate-invariant: for example, while

$$F(x, y) = x^2 - y^2 \, (= x^2 * 1 + (-1) * y^2)$$

is rank-two in coordinates (x, y), if we define $u = x + y$ and $v = x - y$, then

$$F(u, v) = uv,$$

which is rank-one in (u, v).

A lower bound on the rank of a function relative to a given coordinate system can be obtained by calculating the rank of the matrix A of function values of F applied to the nodes of any grid in $\Re^2$; that is, A such that $a_{ij} = F(x_i, y_j)$. Thus, if F can generate a rank-two node-matrix, then the rank of F is at least two.

One can verify that $S(x, y)$ of equation 2.2 is of (at most) rank-n by setting $\alpha_i(x) \equiv u_i(x)$ and $\beta_i(y) \equiv d_i v_i(y)$.

3 Curve Meshes: Introduction

Quite informally, a curve mesh is an interlocking grid of functions (refer to figure (1)). More formally,

Definition 3.1 *A* **curve mesh** M *is the union of graphs of functions on* $[a,b] \times [c,d]$ *in* $\Re^2$,

$$M = \bigcup_{j=1}^{n} \{(x, y_j, g_j(x)) | x \in [a,b]\} \cup \bigcup_{i=1}^{m} \{(x_i, y, h_i(y)) | y \in [c,d]\},$$

where

- $a \equiv x_1 < x_2 < \ldots < x_m \equiv b$ *and* $c \equiv y_1 < y_2 < \ldots < y_n \equiv d$, *and*
- $g_j(x_i) = h_i(y_j)$ *(which makes the functions* *interlocking*).

Each curve mesh defines a **node matrix** A, *the nodes of the function intersections:*

$$a_{ij} \equiv h_i(y_j) \, (= g_j(x_i)).$$

In practice, a curve mesh often starts as a matrix of what might be called *control points* located on a grid in space; these points are then joined to each other along the grid directions (creating the functions g_j and h_i on the respective domains).

The idea of curve mesh interpolation is really very simple: a curve mesh may be considered a restriction of an unknown function (f) to a grid. We seek to create an admissible f function, in the sense that $g_j(x) = f(x, y_j)$ and $h_i(x) = f(x_i, y)$. We thus seek skin f of A, which simultaneously interpolates the curves g and h in their respective directions.

4 Curve Mesh Interpolation: the square, invertible matrix case

We begin by considering curve mesh M such that $m = n$, and such that the associated node matrix A is invertible.

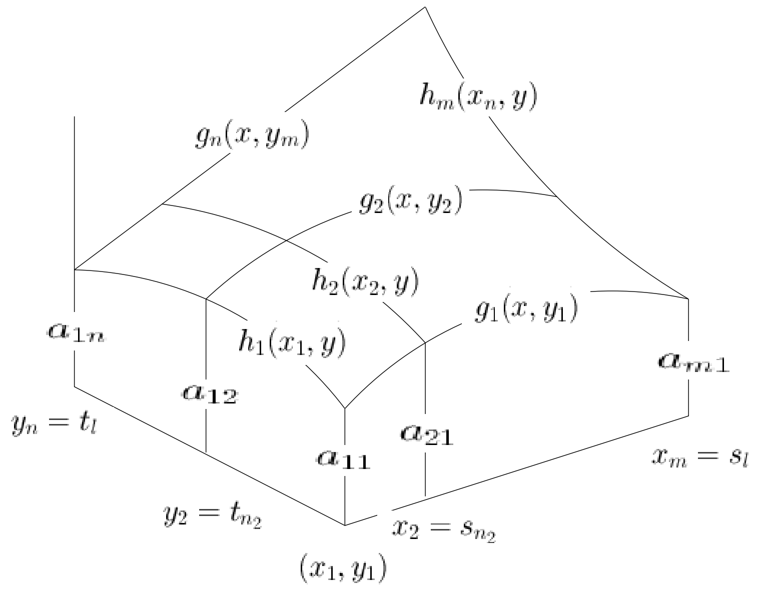

Figure 1: The graph of a curve mesh to help clarify notation. **We need to fix this figure, so that it shows the functions g and h with only single arguments, perhaps writing $g_1(x) = f(x, y_1)$ in place of $g_1(x, y_1)$.**

Theorem 4.1 *(Curve Mesh Theorem) Given curve mesh M as in Definition 3.1 such that $n = m$, and such that the associated node matrix $A_{m\times m}$ is invertible. Then*

$$S(x,y) = \begin{bmatrix} g_1(x) \\ \vdots \\ g_m(x) \end{bmatrix}^T A^{-1} \begin{bmatrix} h_1(y) \\ \vdots \\ h_m(y) \end{bmatrix} \equiv \underline{G}(x)^T A^{-1} \underline{H}(y)$$

is a skin of A which simultaneously interpolates the functions of vectors $\underline{G}(x)$ and $\underline{H}(y)$. We call this the **inverse method** *of curve mesh interpolation.*

Proof: The proof makes use of the SVD method of matrix interpolation. We seek a skin S of A,

$$S(x,y) = \underline{U}(x)^T D \underline{V}(y) \tag{4.3}$$

constrained such that $S(x_i, y) = h_i(y)$ and $S(x, y_i) = g_i(x)$ for all $i = 1, \ldots, m$.

$$S(x_i, y) = \underline{U_{i.}}^T D \underline{V}(y),$$

and hence

$$S(\underline{x}, y) = UD\underline{V}(y)$$

(understanding $S(\underline{x}, y)$ as the vector of elements $S(x_i, y)$). Imposing the condition that

$$S(\underline{x}, y) = UD\underline{V}(y) \equiv \underline{H}(y)$$

gives

$$\underline{V}(y) \equiv D^{-1}U^T\underline{H}(y).$$

Similarly

$$S(x, y_i) = \underline{U}(x)^T D\underline{V_{i.}}$$

and hence

$$S(x, \underline{y})^T = \underline{U}(x)^T DV^T.$$

Imposing the constraint that

$$S(x, \underline{y})^T = \underline{U}(x)^T DV^T \equiv \underline{G}(x)^T$$

yields

$$\underline{U}(x)^T \equiv \underline{G}(x)^T VD^{-1}.$$

Combining these results, we obtain

$$S(x, y) = \underline{U}(x)^T D\underline{V}(y) = \underline{G}(x)^T VD^{-1}DD^{-1}U^T\underline{H}(y),$$

or

$$S(x, y) = \underline{G}(x)^T VD^{-1}U^T\underline{H}(y) = \underline{G}(x)^T A^{-1}\underline{H}(y),$$

as one can see by inverting A from equation 1.1. Q.E.D.

An example from the literature due to Franke[5] is illustrated in figure (2). One observation to make at this point is that, in contrast to the SVD method of interpolation of a matrix, which permits a high degree of creativity in the interpolation process, the skin obtained using the inverse method of curve mesh interpolation is unique.

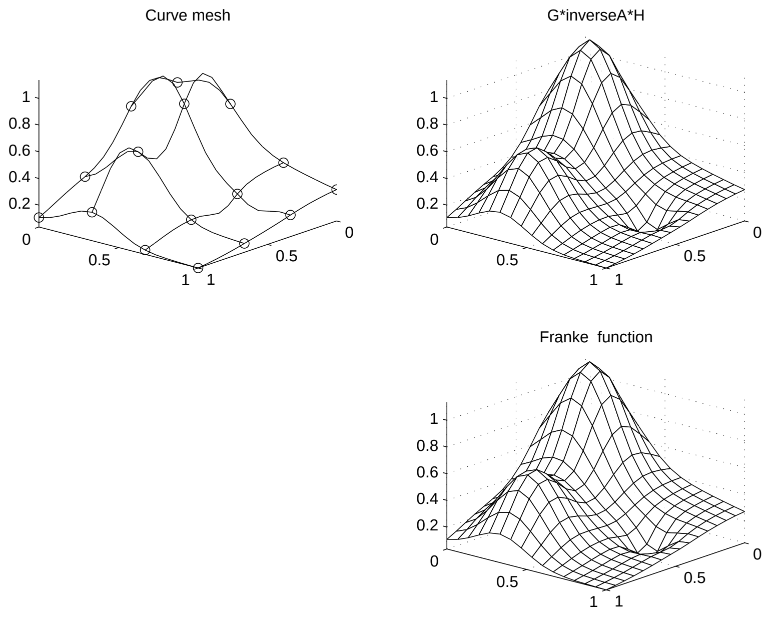

Figure 2: Franke function. The mesh in the upper-left defines a 4×4 *node-matrix* of elevations (indicated by the circles at the intersections of the mesh functions). This matrix is invertible, and the function represented in the upper right is the result of using the inverse method of curve mesh interpolation (Theorem (4.1)). Note, however, that we have not used the default domain ([1,4]), but rather vector interpolants on abscissa vector $[0\,\frac{1}{3}\,\frac{2}{3}\,1]^T$.

5 Curve Mesh Interpolation: the square, non-invertible matrix case

In the Franke example (Figure 2), the node-matrix A was square and invertible, and so the inverse method worked well to reproduce the Franke function (which is of rank four). But suppose that the square matrix A were non-invertible? How might one proceed?

One option is to simply replace the inverse with the pseudo-inverse:

$$S(x,y) = \underline{G}(x)A^{+}\underline{H}(y) \qquad (5.4)$$

Interestingly enough, this method will suffice to reproduce the curve mesh, provided that the dimensions of the spans of the functions of $\underline{G}$ and $\underline{H}$ are at most the rank of A. If either of the spans is of higher rank, then we require another strategy.

Two examples from the literature illustrate the use of the pseudo-inverse method. Figure 3 shows the results of utilizing this technique on an example Bos, *et al.*[2], based on the function $F(x, y) = \frac{1}{1+(x_2+y_2)^{10}}$ and on mesh curves defined on an 11×11 grid. In this case A is not of numerical full rank (rank six). The function F itself is obviously not of finite rank (since at best it can only be expanded in an infinite power series of product functions).

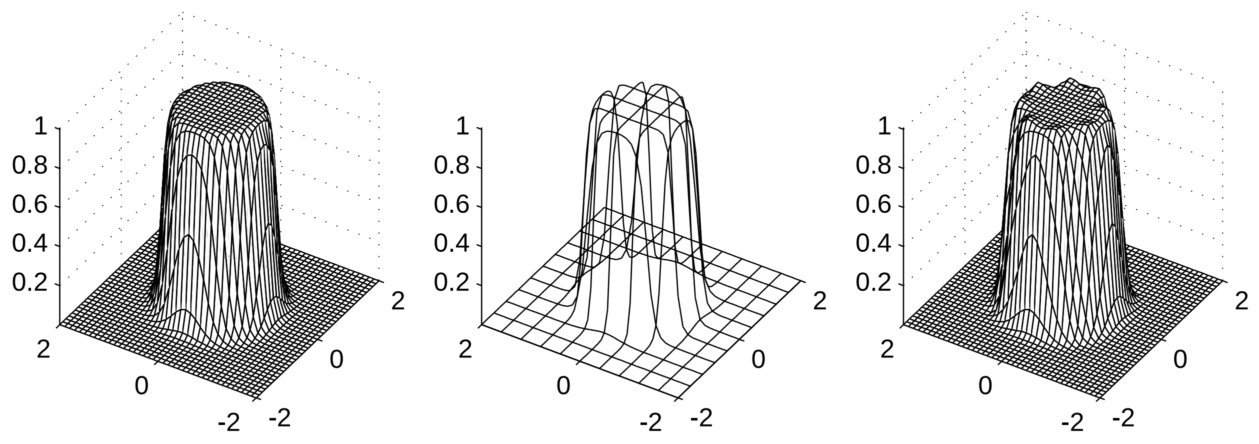

Figure 3: At left is the surface of $F(x, y) = \frac{1}{1+(x_2+y_2)^{10}}$, studied in [2]; in the center is the 11×11 mesh, and at right the skin of the curve mesh using the pseudo-inverse method.

It is quite possible that an ill-conditioned A matrix might yield extraneous pits and peaks in the process under consideration. Most software packages have pseudo-inverse routines which allow the user to adjust the pseudo-inverses in such cases. For example, the MATLAB call, pinv(A,tol), allows a tolerance adjustment which essentially avoids the difficulty caused by relatively small eigenvalues of the matrix A when computing the pseudoinverse. The MATLAB call used in the example of Figure 4 was "pinv(A,1e-3)" in order to avoid these peaks, which essentially reduced the condition number of the resulting matrix to five thousand. Note that the polygonal paths in Figure 4 are just linear interpolants of smooth curves. This folded parabolic cylinder example was studied by Bos, *et al.*[2] and is represented by $F(x,y) = |y - x_2 + 0.4317|$, with an 11×11 grid used to generate the original mesh curves. We believe that our interpolating surface compares favorably to theirs, and does not require prior knowlege of the kink on the parabolic base curve (as does their method). In this case the numerical rank of A is only 5.

We demonstrate the *augmented inverse method* via an example.

Consider the matrix

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

with an associated mesh created by taking sections of the function $F(x,y) = xy$ along the edges of the domain $[0,1] \times [0,1]$. The node matrix A (the values of F at the corners) is non-invertible, and so the inverse method will fail. F is rank-one, however, so the pseudo-inverse method (equation 5.4) interpolates the mesh generated by F. The pseudo-inverse method would not have skinned the mesh generated by the function $F(x,y) = xy + sin(2\pi x)$, however, which has the same node matrix, as it is of rank two.

In the interest of demonstrating the augmented inverse method, we *augment* or *complete* the associated node-matrix A to create A^*, by adding a constant matrix to A, e.g.,

$$A^* = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

We may think of as the node-matrix A^* as corresponding to the function $1 + f(x,y)$.

This matrix A^* is invertible, and so we can apply the inverse method (Theorem 4.1) to create S^*, obtaining the skin $S(x,y)$ as

$$S(x,y) \equiv S^*(x,y) - 1.$$

In effect, the constant (1) added at each node is interpolated by the constant *completion function* $C(x,y) = 1$, which we subsequently remove:

$$S(x,y) \equiv S^*(x,y) - C(x,y).$$

This idea is easily generalized: rather than adding only a constant, we add additional node-matrices, corresponding to known functions, to create an invertible matrix. Thus, for example, we might consider the 3×3 matrix

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

with mesh created using sections of the function

$$F(x,y) = \frac{\sin(2\pi x)}{|y|+1)} + 4xy(x-.5)(y-.5) \tag{5.5}$$

F is rank-two (that is, F can be written as the sum of two functions of the form $F(x,y) = g_1(x)h_1(y) + g_2(x)h_2(y)$, and not fewer). We "complete" A as follows:

$$A^* = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1.00000 & 1.64872 & 2.71828 \\ 1.64872 & 2.71828 & 4.48169 \\ 2.71828 & 4.48169 & 7.38906 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

or

$$A^* = \begin{bmatrix} 2.00000 & 2.64872 & 3.71828 \\ 2.64872 & 3.71828 & 5.48169 \\ 3.71828 & 5.48169 & 9.38906 \end{bmatrix},$$

which we actually derived as the node-matrix corresponding to the function $1 + e^x e^y + F(x,y)$. This matrix A^* is invertible, so we use the inverse method and obtain function S^*; from it, we create S as

$$S(x,y) \equiv S^*(x,y) - 1 - e^x e^y,$$

as shown in Figure 5.

One may object that the completion of A is not unique: any addition of functions which leads to an invertible node matrix produces a skin of the mesh. The choices of the constant (=1) component and the $e^x e^y$ component were simply expeditious: a rank-one function would not suffice; but many choices of rank-two (or higher) functions would work. As in the case of the SVD method of matrix interpolation, we consider this a plus: it provides some design potential. If one choice fails to produce the desired skin, then another choice may be attempted.

6 Curve Mesh Interpolation: the rectangular matrix case

We have thus taken care of the case of "square" curve meshes (that is, those with square node matrices for which $n = m$). In the event that $n \neq m$, there is a relatively simple strategy. WLOG consider the case $m > n$: in this case, there are more row interpolants than column interpolants. Introduce $m - n$ additional "column interpolants", corresponding to "missing columns". That is, select $m - n$ additional abscissas to insert into the vector of abscissas $\underline{y}$ to create $\underline{y}^*$. These may even fall outside the interval $[y_1, y_n]$, provided that the functions $\{h_k(y)|i = 1, \ldots, m\}$ are defined in the larger interval $[y_1^*, y_m^*]$.

If Y is one of these new abscissas, then the interlocking constraint implies that the introduced interpolant function $g_(x)$ must satisfy

$$g(x_i) = h_i(Y)\,\forall i = 1, \ldots, m.$$

This can be done in any number of ways, such as by cubic splines. One is limited only by one's imagination!

References

[1] H. C. Andrews and C. L. Patterson. Outer product expansions and their uses in digital image processing. *American Mathematical Monthly*, 82(1):1–13, January 1975.

[2] L. P. Bos, J. E. Grabenstetter, and K. Salkauskas. Pseudo-tensor product interpolation and blending with families of univariate schemes. *Computer Aided Geometric Design*, 13:429–440, 1996.

[3] P. Costantini and C. Manni. A bicubic shape-preserving blending scheme. *Computer Aided Geometric Design*, 13:307–331, 1996.

[4] T. A. Foley and H. S. Ely. Surface interpolation with tension controls using cardinal bases. *Computer Aided Geometric Design*, 6:97–109, 1989.

[5] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comp.*, 38(157):181–200, January 1982.

[6] T. Hern and C. A. Long. Viewing some concepts and applications in linear algebra. *MAA Notes Series*, 19:173–190, May 1991.

[7] K. Hoechsmann. Singular values and the spectral theorem. *The American Mathematical Monthly*, 97(5):413–414, May 1990.

[8] K. Jetter and F. I. Utreras. *Multivariate Approximation.* World Scientific, 1992.

[9] D. Kalman. A singularly valuable decomposition: the SVD of a matrix. *College Math. Journal*, 27(1):2–23, 1996.

[10] A. E. Long and C. A. Long. Surface approximation and interpolation via matrix svd. *The College Mathematics Journal*, January 2000.

[11] C. A. Long. Visualization of matrix singular value decomposition. *Mathematics Magazine*, 56(3):161–167, May 1983.

[12] E. V. Shilkin and A. I. Plis. *Handbook on Splines for the User.* CRC Press, 1995.

[13] H. Spath. *Two Dimensional Spline Interpolation Algorithms.* A. K. Peters, 1994.

[14] X. Ye. Curvature continuous interpolation of curve meshes. *Computer Aided Geometric Design*, 14:169–190, 1997.

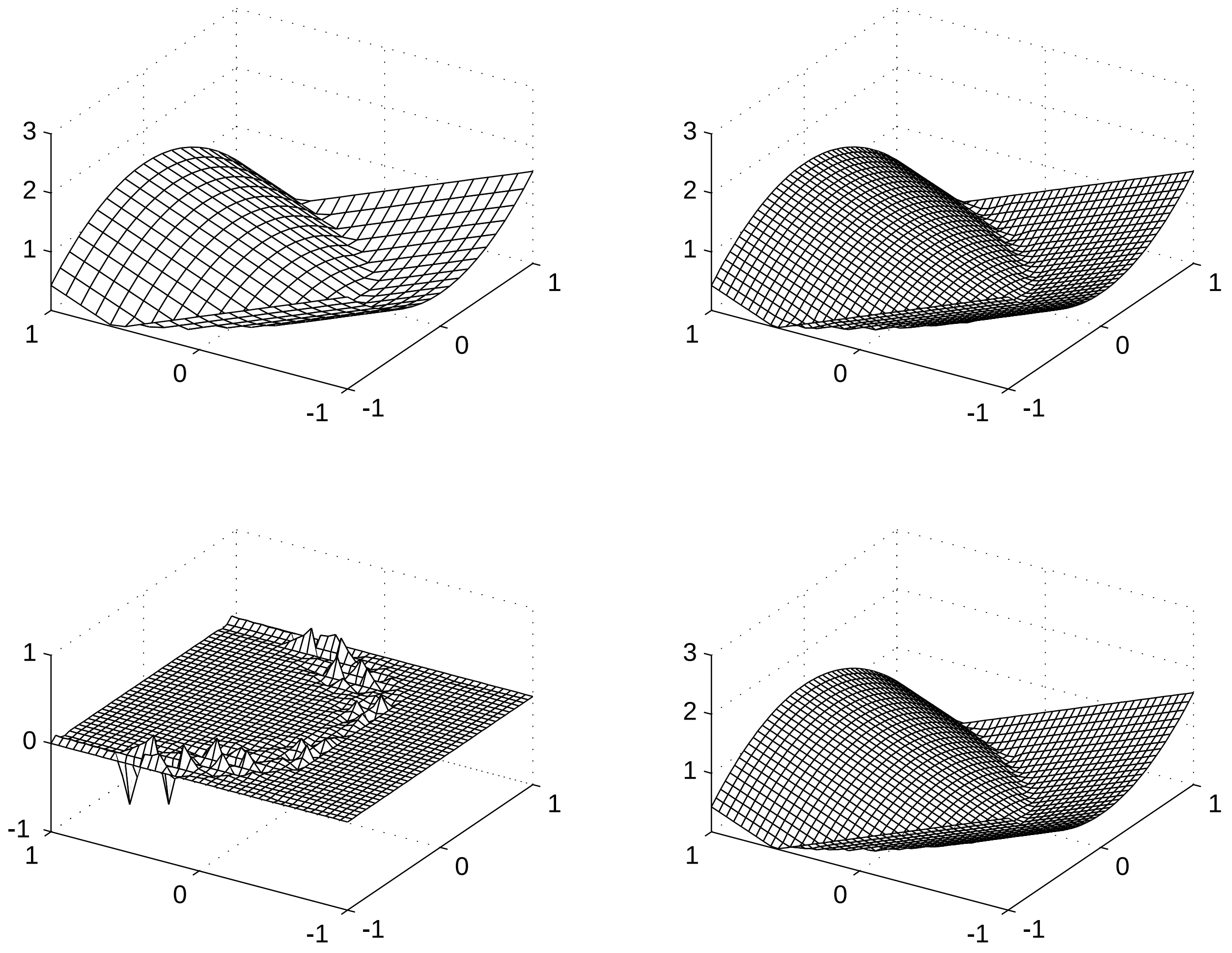

Figure 4: Another function from Bos, *et al.*[2]: $F(x, y) = |y - x_2 + 0.4317|$.
Dad: what's this one again?

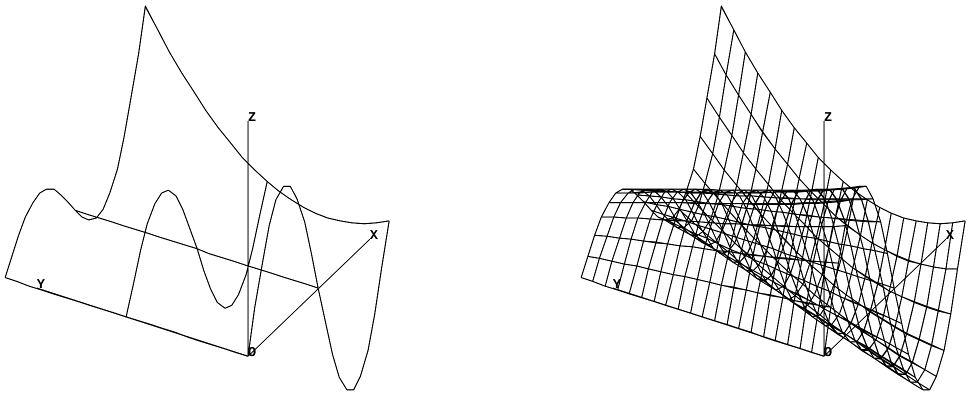

Figure 5: Mesh and skin for the curve mesh created from the function of equation (5.5).