THE COMPUTATIONAL COST OF SIMPLEX SPLINE FUNCTIONS

Thomas A. Grandine

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

March 1986

(Received March 12, 1986)

Approved for public release
Distribution unlimited

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina  27709

National Science Foundation
Washington, DC  20550

86520127
THE COMPUTATIONAL COST OF SIMPLEX SPLINE FUNCTIONS

Thomas A. Grandine\textsuperscript{1,2}

Technical Summary Report \#2926

March 1986

ABSTRACT

This paper presents a negative result concerning the stable evaluation of simplex spline functions. It has been conjectured that a great deal of computational effort might be saved by implementing the recurrence relation for these functions in a clever way. The main result of this paper is that there is no clever way to implement the recurrence relation once the standard recipe for constructing spaces of simplex spline functions has been followed.

AMS(MOS) Subject Classifications: 41A15, 65D07

Key Words: B-spline, simplex spline, spline function, multivariate, recurrence relation

Work Unit Number 3 – Numerical Analysis and Scientific Computing

\textsuperscript{1}Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.
\textsuperscript{2}This material is based upon work supported by the National Science Foundation under Grant No. DMS-8210950, Mod. 4.
SIGNIFICANCE AND EXPLANATION

All computational schemes for simplex spline functions to date rely both on the recurrence relation for these functions and the standard construction of the simplex spline basis. Under these conditions, for numerical methods for computing simplex spline functions to be as useful as possible, it is necessary to find ways of implementing the recurrence relation as efficiently as possible. This paper shows that "as efficiently as possible" is still not very efficient. This implies that truly fast algorithms (which have not yet been developed) will have to either abandon the recurrence relation or the standard construction of the simplex spline basis.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.
THE COMPUTATIONAL COST OF SIMPLEX SPLINE FUNCTIONS

Thomas A. Grandine\textsuperscript{1,2}

Since de Boor defined the multivariate simplex spline in [B76], many of its mathematical properties have been studied and reported ([B82], [D79], [M80], ...). However, numerical methods for evaluating these functions have not been studied. The goal of this paper is to produce an algorithm which evaluates simplex spline functions, i.e., linear combinations of simplex splines. This can be done in a straightforward manner by using the evaluation algorithm for individual simplex splines, described in [G84], to evaluate all non-zero simplex splines at any point \( x \), then summing up the values using the spline function coefficients as weights. However, since this algorithm is based on the recurrence relation for simplex splines, [M80], it seems a wasteful approach. In particular, many of the lower degree simplex splines which need to be evaluated during this process may well be evaluated multiple times. It seems reasonable (and has been conjectured by Dahmen and Micchelli [DM81] and perhaps others) that a great deal of computational effort might be saved by taking advantage of this multiplicity. This paper makes an attempt to do just that.

In order to begin this attempt, the recurrence relation is needed. This is given by

\[ M(x; \sigma_0, \sigma_1, \ldots, \sigma_n) = \sum_{i=0}^{n} a_i M(x; \sigma_0, \sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n). \]  

\textsuperscript{1}Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

\textsuperscript{2}This material is based upon work supported by the National Science Foundation under Grant No. DMS-8210950, Mod. 4.
where
\[ x = \sum_{r=0}^{n} \alpha_r P_{\sigma_r}. \]

Here \( P_z \) means the first \( m \) components of the vector \( z \).

The second necessary step is to clarify just which simplex splines are the correct ones to deal with when applying this recurrence relation. This requires the construction of an appropriate simplex spline basis. Höllig \cite{H82} and Dahmen and Micchelli \cite{DM82} have simultaneously and independently given versions of one such construction. The Höllig version of the construction is summarized here:

To construct a simplex spline basis over \( \Omega \subseteq \mathbb{R}^m \), start with a triangulation of \( \Omega \).

A triangulation of a set \( \Omega \) is a collection \( T \) of sets \( \tau \) such that

1. \( \text{vol}[\tau] > 0 \)
2. \( \cup_{\tau \in T} [\tau] = \Omega \)
3. \( [\tau] \cap [\tau'] = [\tau \cap \tau'], \quad \tau, \tau' \in T \)
4. \( \# \tau = m + 1, \quad \tau \in T, \)

where \( [A] \) is the convex hull of \( A \) and \( \# A \) is the cardinality of \( A \). Thus, each \( [\tau] \) is a simplex in \( \Omega \). Consider the slab \( \Omega \cdot \Sigma_k := \{(u,v)|u \in \Omega, v \in \Sigma_k\} \), where \( \Sigma_k \) is the standard unit simplex in \( \mathbb{R}^k \). The construction boils down to triangulating this slab and using the vertices of each of the resulting simplices as knot sets for the simplex splines.

The slab can be triangulated as follows: For each \( \tau \in T \), consider the following grid:

\[
\begin{array}{cccc}
\tau_{0,1} & \tau_{1,1} & \cdots & \tau_{m,1} \\
\vdots & \vdots & \ddots & \vdots \\
\tau_{0,m} & \tau_{1,m} & \cdots & \tau_{m,m} \\
\end{array}
\]
Here, \( t_{ij} \) is intended to represent \((t_i, e_j)\), where \( t_i \) is a point in \( \tau \) and \( e_j \) is the \( j \)-th standard unit vector (with \( e_0 \) being 0). Hence \( t_{ij} \) is a vertex of \( \tau \times \Sigma_k \). (This is not quite true. In the actual construction, the knots are pulled apart. This means that \( t_{ij} \) is not really \((t_i, e_j)\), but rather some point which is close to \((t_i, e_j)\). The actual details are quite interesting and useful, but not of importance here). Consider all non-decreasing paths from \( t_{0,0} \) to \( t_{m,k} \). Such a path is denoted by \( \sigma_0, \sigma_1, ..., \sigma_n \) where \( \sigma_0 = t_{0,0}, \sigma_n = t_{m,k}, \) and, if \( \sigma_\ell = t_{ij} \), then \( \sigma_{\ell+1} \) is either \( t_{i+1,j} \) or \( t_{i,j+1} \). The set of all paths through the grid for \( \tau \) for all \( \tau \in T \) give rise to a simplex spline basis.

Thus, each of the knots, \( \sigma_\ell \), can be classified according to where it was found along the path from \( t_{0,0} \) to \( t_{m,k} \). Specifically, given \( \sigma_\ell = t_{ij} \), then \( \sigma_\ell \) is called a horizontal knot if \( \sigma_{\ell-1} = t_{i-1,j} \) and \( \sigma_{\ell+1} = t_{i+1,j} \). If \( \sigma_{\ell-1} = t_{i,j-1} \) and \( \sigma_{\ell+1} = t_{i,j+1} \), then \( \sigma_\ell \) is called a vertical knot. All other knots are called corner knots. Note that these names are specific to a given simplex spline, i.e. a knot which is of one type in one simplex spline may be of a different type in a different simplex spline. This classification of knots has no significance except to relate the simplex splines to the paths from which they were derived.

The paths provide the key to piecing lower degree simplex splines together, as will be made clear shortly.

**Definition:** The simplex spline \( M_\ell := M(\cdot|\sigma_0, ..., \sigma_{\ell-1}, \sigma_{\ell+1}, ..., \sigma_n) \) is said to contribute to the simplex spline \( M(\cdot|\sigma_0, ..., \sigma_n) \) if, in (1), \( M_\ell \) necessarily has a non-zero coefficient corresponding to it.

In order to prove the upcoming theorem in this paper, two more concepts are needed. The first notion is that of a cut corner. A path with a cut corner is obtained from a path \( \sigma_0, ..., \sigma_n \) by removing a corner point. This amounts to finding a path through the grid
consisting of only horizontal and vertical moves, except for one place, arbitrarily chosen, where a diagonal move is allowed. The second notion has to do with the grids themselves. In addition to the grid for \( r \), a grid may also be considered for \( r \cap r' \). If \( r \neq r' \), then this grid will have fewer than \( m - 1 \) columns, and a path through the grid will only produce a simplex spline of degree \( k - c - m - 1 \), where \( c \) is the number of columns of the grid.

**Theorem 1:** Assume that \( t_{i,j} \neq t_{i',j'} \) unless \( i = i' \) and \( j = j' \). Then the only simplex splines of degree \( k - 1 \) which can contribute to two simplex splines of degree \( k \) are those which come from paths with cut corners and those which come from paths corresponding to grids for \( r \cap r' \), where \( r \cap r' = m \). There are no simplex splines of degree \( k - 1 \) which contribute to three or more simplex splines of degree \( k \).

**Proof:** Take any simplex spline of degree \( k \), say \( M \), deriving from a path corresponding to the grid for \( r \). Then the only simplex splines which can contribute to \( M \) are those obtained by deleting a knot, say \( \sigma_j \), along that path. Suppose \( \sigma_j \) is a vertical knot. Then \( M_j := M(\sigma_0, \ldots, \sigma_{j-1}, \sigma_{j+1}, \ldots, \sigma_n) \) can only contribute to \( M \), since the only way to get from \( \sigma_{j-1} \) to \( \sigma_{j+1} \) along a path is through \( \sigma_j \). Suppose \( \sigma_j \) is a corner knot. Assume that \( \sigma_{j-1} = t_{j-\ell, \ell-1} \). Then, since \( \sigma_j \) is a corner knot, \( \sigma_{j+1} = t_{j-\ell+1, \ell} \). Thus, the only way to get from \( \sigma_{j-1} \) to \( \sigma_{j+1} \) along a path is through either \( t_{j-\ell, \ell-1, \ell+1} \) or \( t_{j-\ell, \ell, \ell} \), one of which must be \( \sigma_j \). Thus, \( M_j \) can contribute at most two simplex splines of degree \( d \).

Lastly, suppose \( \sigma_j \) is a horizontal knot. Then \( M_j \) comes from a path along a grid where the column corresponding to \( \sigma_j \) is deleted. Suppose there exists \( r' \) such that \( r \cap r' \) provides just such a grid. Then \( M_j \) contributes both to \( M \) and to the corresponding simplex spline from the grid for \( r' \), but to no others. Otherwise, \( M_j \) just contributes to \( M \). This proves the theorem.
A computer code has been written using this idea. The same approach to implementing the recurrence relation that is employed in G84 is used here. Here, however, it is best to have non-zero as few of the \( \alpha \), corresponding to vertical knots as possible. Unfortunately, including this condition results in a problem whose computational complexity is equivalent to that of the travelling salesman problem. Since the goal here is to write a fast code, the simple heuristic device of steering the method toward corner and horizontal knots whenever possible is employed. This does not behave perfectly, but it does function adequately, and the non-zero \( \alpha \), obtained usually correspond to corner and horizontal knots when such solutions are possible. Additionally, the code keeps track of which lower degree simplex splines have already been computed so that their results can be reused if possible. A hashing scheme has been implemented to provide nearly direct access to the values of these already computed simplex splines.

Unfortunately, the code behaves quite poorly in practice. In actual comparison with the more naive approach, i.e., the one in which no intermediate results are saved and all simplex splines are computed one at a time, the naive approach is the clear and decisive winner. The clever approach takes approximately three times as much CPU time, and this ratio seems not to change with either the degree of the simplex splines or the complexity of the knots, at least through seventh degree polynomials in the bivariate case!

The explanation for this lies in the excessive overhead involved in keeping track of the lower degree intermediate results. The only way to distinguish between values of different simplex splines is to somehow distinguish between the simplex splines themselves. This can only be done by enumerating the knots which make up the simplex spline. Thus, the amount of overhead increases with the degree of the simplex splines. This means that the
payoff that might be expected by making the degree of the simplex splines sufficiently large does not actually occur.

What's worse, Theorem 1 says, in some sense, that this cannot be improved on. While other programmers might be able to improve the code sufficiently to make it competitive, it cannot be done in such a way that the performance of the naive method can be bettered by more than a factor of $2^{k-1}$. Indeed, such a program would have to have some overhead not present in the naive code, and there will usually be lower degree simplex splines which contribute to only one higher degree simplex spline present. Thus, a factor of $2^{k-1}$ improvement is really far too much to hope for.
References


This paper presents a negative result concerning the stable evaluation of simplex spline functions. It has been conjectured that a great deal of computational effort might be saved by implementing recurrence relation for these functions in a clever way. The main result of this paper is that there is no clever way to implement the recurrence relation once the standard recipe for constructing spaces of simplex spline functions has been followed.
END FILMED

6-86

D TIC