Simplicial Multivariable Linear Interpolation

TR91-002

February, 1991

John H. Halton

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

SIMPLICIAL MULTIVARIABLE LINEAR INTERPOLATION

John H. Halton
Computer Science Department
The University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

1. INTRODUCTION

Given a well-behaved multivariable function $F(\xi_1, \xi_2, \ldots, \xi_k)$, whose values are given at the nodes of a cubic lattice

$$L = \Big\{p_1\mu, p_2\mu, \ldots, p_k\mu \;\Big|\; (\forall j \mid 1 \le j \le k)\; p_j \in \mathbb{Z}\Big\}, \tag{1}$$

where μ is the *lattice-constant* of L and $\mathbb{Z}$ is the set of all (positive or negative) integers; we consider the problem of *efficiently* (i.e., with least effort) and *effectively* (i.e., with greatest accuracy) *interpolating* these values at any point $(\xi_1, \xi_2, \ldots, \xi_k)$.

Let us write

$$\lfloor z \rfloor = \max\{q \in \mathbb{Z} \mid q \le z\} \quad \text{and} \quad \langle z \rangle = z - \lfloor z \rfloor \tag{2}$$

(so that $\lfloor z \rfloor$ is what is usually called the *floor function*, or the *integer part*, of z, and $\langle z \rangle$ is what is usually called the *fractional part* of z). Clearly, we have that

$$0 \le \left\langle z \right\rangle < 1. \tag{3}$$

Now, let us define

$$p_j = \left\lfloor \frac{\xi_j}{\mu} \right\rfloor \quad \text{and} \quad x_j = \left\langle \frac{\xi_j}{\mu} \right\rangle; \tag{4}$$

so that

$$(\forall j \mid 1 \le j \le k) \quad 0 \le x_j < 1 \tag{5}$$

and

$$(\forall j \mid 1 \le j \le k) \quad \xi_j = (p_j + x_j)\mu. \tag{6}$$

We limit ourselves to the interpolation of F at $(\xi_1, \xi_2, \ldots, \xi_k)$ as a *multivariable linear* function of $x_1, x_2, \ldots, x_k$, using only the values

$$f_{\boldsymbol{c}} = F_{\boldsymbol{p},\boldsymbol{c}} = F\Big((p_1 + c_1)\mu, (p_2 + c_2)\mu, \ldots, (p_k + c_k)\mu\Big), \tag{7}$$

where

$$(\forall j \mid 1 \le j \le k) \quad c_j \in \{0, 1\}, \tag{8}$$

as coefficients. These are the known values of F at the vertices of the lattice-cell enclosing the point $(\xi_1, \xi_2, \ldots, \xi_k)$.

The usual 'full' multivariable linear interpolation then takes the form

$$\Phi_{\boldsymbol{x}} = \Phi(\xi_1, \xi_2, \ldots, \xi_k)$$

$$= \sum_{c_1=0}^{1} \sum_{c_2=0}^{1} \cdots \sum_{c_k=0}^{1} \left\{ \prod_{j=1}^{k} \left[c_j x_j + (1 - c_j)(1 - x_j) \right] \right\} f_{\boldsymbol{c}}. \tag{9}$$

Thus, it involves a combination of 2^k data for each interpolation.

A k-dimensional *simplex* T is defined by $k + 1$ vertices,

$$T = \mathbb{T}(P_0, P_1, P_2, \ldots, P_k), \tag{10}$$

where

$$(\forall s \mid 0 \le s \le k) \quad P_s = (\alpha_{1s}, \alpha_{2s}, \ldots, \alpha_{ks}), \tag{11}$$

$$\text{and} \quad T = \left\{ (\xi_1, \xi_2, \ldots, \xi_k) \;\middle|\; \begin{array}{r} (\forall s \mid 0 \le s \le k) \quad 0 \le \lambda_s \le 1 \\ 1 = \displaystyle\sum_{s=0}^{k} \lambda_s \\ (\forall j \mid 1 \le j \le k) \quad \xi_j = \displaystyle\sum_{s=0}^{k} \alpha_{js}\lambda_s \end{array} \right\}. \tag{12}$$

We now consider the possibility of obtaining an interpolated approximation to $F(\xi_1, \xi_2, \ldots, \xi_k)$ by a formula of the form

$$\Psi_x = \Psi(\xi_1, \xi_2, \ldots, \xi_k) = \sum_{s=0}^{k} \lambda_s F(P_s), \tag{13}$$

where the P_s are selected from the 2^k lattice points

$$\big((p_1 + c_1)\mu, (p_2 + c_2)\mu, \ldots, (p_k + c_k)\mu\big) \tag{14}$$

defined as in (4)–(9) above.

2. EXISTENCE

Our first problem is to determine whether, indeed, given any point $(\xi_1, \xi_2, \ldots, \xi_k)$, there always exists a simplex T, defined as in (10)–(12), whose vertices are a subset of the vertices of the lattice-cell enclosing the point $(\xi_1, \xi_2, \ldots, \xi_k)$, which contains the given point. Equivalently, we ask whether there is a subset of $k + 1$ of the 2^k points $\boldsymbol{c} = (c_1, c_1, \ldots, c_k)$ satisfying (8), which form the vertices of a simplex containing any given point $(x_1, x_2, \ldots, x_k)$ satisfying (5).

We observe that, by (12), we require that there be $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_k$, such that $(\forall s \mid 0 \le s \le k)$ $0 \le \lambda_s \le 1$, and

$$\left\{ \begin{array}{rr} & \sum_{s=0}^{k} \lambda_s = 1 \\ (\forall j \mid 1 \le j \le k) & \sum_{s=0}^{k} c_{js}\lambda_s = x_j \end{array} \right\}. \tag{15}$$

Dropping the conditions on the λ_s, we see that this is equivalent to the matrix equation

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ c_{10} & c_{11} & c_{12} & \dots & c_{1k} \\ c_{20} & c_{21} & c_{22} & \dots & c_{2k} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{k0} & c_{k1} & c_{k2} & \dots & c_{kk} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_k \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_k \end{bmatrix}, \tag{16a}$$

or

$$\boldsymbol{M L} = \boldsymbol{X}. \tag{16b}$$

THEOREM 1. *For any* $(x_1, x_2, \dots, x_k)$ *satisfying (5), there is a selection of values* c_{js} $(1 \le j \le k,\ 0 \le s \le k)$, *such that (16) has a unique solution* $\lambda_0, \lambda_1, \lambda_2, \dots, \lambda_k$, *and* $(\forall s \mid 0 \le s \le k)\ 0 \le \lambda_s \le 1$.

Proof. We can certainly put the $x_j \in \{0, 1\}$ into non-increasing order: there is a permutation $[r_1, r_2, \dots, r_k]$ of $[1, 2, \dots, k]$, such that

$$\{r_1, r_2, \dots, r_k\} = \{1, 2, \dots, k\}, \tag{17}$$

and

$$1 \ge x_{r_1} \ge x_{r_2} \ge \dots \ge x_{r_k} \ge 0. \tag{18}$$

Let us now define a corresponding permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \varpi_{11} & \varpi_{12} & \dots & \varpi_{1k} \\ 0 & \varpi_{21} & \varpi_{22} & \dots & \varpi_{2k} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & \varpi_{k1} & \varpi_{k2} & \dots & \varpi_{kk} \end{bmatrix}, \tag{19}$$

by

$$\varpi_{ij} = \left\{ \begin{matrix} 1 & \text{if} & \left\{ \begin{matrix} \text{either} & i = j = 0 \\ \text{or} & 1 \le j = r_i \end{matrix} \right\} \\ 0 & \text{otherwise} & \end{matrix} \right\}. \tag{20}$$

As is easily verified (and well-known),

$$P^{\mathsf{T}} P = P P^{\mathsf{T}} = \mathbf{I}, \tag{21a}$$

where $\mathbf{I}$ is the *identity matrix* and P^{T} denotes the *transpose* of P; so that

$$P^{-1} = P^{\mathsf{T}}. \tag{21b}$$

Now, from (16),

$$P M L = P X, \tag{22a}$$

which, with a little thought, reduces to the form

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ c_{r_1 0} & c_{r_1 1} & c_{r_1 2} & \dots & c_{r_1 k} \\ c_{r_2 0} & c_{r_2 1} & c_{r_2 2} & \dots & c_{r_2 k} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ c_{r_k 0} & c_{r_k 1} & c_{r_k 2} & \dots & c_{r_k k} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ . \\ . \\ . \\ \lambda_k \end{bmatrix} = \begin{bmatrix} 1 \\ x_{r_1} \\ x_{r_2} \\ . \\ . \\ . \\ x_{r_k} \end{bmatrix}. \tag{22b}$$

Now let us construct the particular matrix

$$\boldsymbol{PM} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 1 & \dots & 1 \\ \cdot & \cdot & \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot\ \cdot & \cdot \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}, \tag{23a}$$

so that

$$c_{r_i j} = \begin{Bmatrix} 1 & \text{if} \quad j \ge i \ge 1 \\ 0 & \text{otherwise} \end{Bmatrix}. \tag{23b}$$

It then follows from (22), by the usual process of *Gaussian elimination*, that

$$\begin{Bmatrix} \lambda_0 = 1 - x_{r_1} \\ \lambda_1 = x_{r_1} - x_{r_2} \\ \lambda_2 = x_{r_2} - x_{r_3} \\ \lambda_3 = x_{r_3} - x_{r_4} \\ \dots \\ \dots \\ \dots \\ \lambda_{k-1} = x_{r_{k-1}} - x_{r_k} \\ \lambda_k = x_{r_k} \end{Bmatrix}. \tag{24}$$

Since, by (18), each of the above $\lambda_s \in \{0, 1\}$, our theorem follows. ▣

It should be observed that, not only have we proved the existence of a suitable simplex in every case, but the proof of the existence theorem shows how the simplex and the coefficients λ_s can be explicitly constructed.

THEOREM 2. *The k! simplices corresponding to the matrices (23) for all k! possible permutations* $\boldsymbol{P}$ *or* ϖ (i.e., $i \to r_i$) *have no common interior points.*

Proof. Consider the simplex described in the proof of Theorem 1. Its vertices are the $k+1$ points $\boldsymbol{c}_s = (c_{1s}, c_{2s}, \ldots, c_{ks})$, with $s = 0, 1, 2, \ldots, k$ (i.e., the columns of $\boldsymbol{M}$, below the first row), satisfying (23). Its *interior* consists of the points with coordinates satisfying

$$1 > x_{r_1} > x_{r_2} > \ldots > x_{r_k} > 0, \tag{25}$$

as is easily seen from the condition that all $0 < \lambda_s < 1$, with (18) and (24).

Now, it is clear that any point $\boldsymbol{x}$ with all its coordinates different can only satisfy an inequality (25) for just one permutation ϖ, given by (20). In other words, any $\boldsymbol{x}$ interior to one simplex can *only* be interior to that one simplex and cannot be in the boundary of any such simplex. This proves our theorem. ▣

Thus we have demonstrated that the simplices defined above constitute a $k!$-fold *dissection* of the hypercube.

3. The Interpolation

The next question we consider is the accuracy of the simplicial interpolation (13), as compared with the full linear interpolation (9). We shall suppose that the function $F(\xi_1, \xi_2, \ldots, \xi_k)$, and the derived function $f(x_1, x_2, \ldots, x_k)$ in the lattice cell enclosing the given point $(\xi_1, \xi_2, \ldots, \xi_k)$, have *Taylor expansions* of sufficient length: by (6) and (7),

$$\begin{aligned} f_{\boldsymbol{x}} &= f(x_1, x_2, \ldots, x_k) = F(p_1\mu + x_1\mu, p_2\mu + x_2\mu, \ldots, p_k\mu + x_k\mu) \\ &= \left\{ 1 + \mu\left(\sum_{i=1}^{k} x_i \frac{\partial}{\partial \xi_i}\right) + \frac{\mu^2}{2}\left(\sum_{i=1}^{k} x_i \frac{\partial}{\partial \xi_i}\right)^2 + \frac{\mu^2}{3!}\left(\sum_{i=1}^{k} x_i \frac{\partial}{\partial \xi_i}\right)^3 + \ldots \right\} f_0, \end{aligned} \tag{26}$$

where $$f_0 = f(0, 0, \ldots, 0) = F(p_1\mu, p_2\mu, \ldots, p_k\mu). \tag{27}$$

It follows that

$$f_c = \left\{ 1 + \mu\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right) + \frac{\mu^2}{2}\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right)^2 + \frac{\mu^2}{3!}\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right)^3 + \ldots \right\} f_0, \quad (28)$$

whence, the full interpolation (9) yields

$$\Phi_x = \sum_{c_1=0}^{1}\sum_{c_2=0}^{1}\cdots\sum_{c_k=0}^{1}\left(\prod_{j=1}^{k}\left[c_j x_j + (1-c_j)(1-x_j)\right]\right)$$

$$\times \left\{ 1 + \mu\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right) + \frac{\mu^2}{2}\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right)^2 + \frac{\mu^2}{3!}\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right)^3 + \ldots \right\} f_0. \quad (29)$$

We are interested in calculating the *error* in this interpolation,

$$\Phi_x - f_x = \sum_{t=0}^{\infty} \frac{\mu^t}{t!}\left\{\sum_{c_1=0}^{1}\sum_{c_2=0}^{1}\cdots\sum_{c_k=0}^{1}\right.$$

$$\left.\left(\prod_{j=1}^{k}\left[c_j x_j + (1-c_j)(1-x_j)\right]\right)\left(\sum_{i=1}^{k} c_i \frac{\partial}{\partial \xi_i}\right)^t f_0 - \left(\sum_{i=1}^{k} x_i \frac{\partial}{\partial \xi_i}\right)^t f_0\right\}. \quad (30)$$

A little algebraic manipulation shows that the first two terms ($t = 0$ and $t = 1$) vanish identically, and the remaining terms yield

$$\Phi_{\boldsymbol{x}} - f_{\boldsymbol{x}} = \frac{\mu^2}{2}\left\{\sum_{c_1=0}^{1}\sum_{c_2=0}^{1}\cdots\sum_{c_k=0}^{1}\left(\prod_{j=1}^{k}\left[c_j x_j + (1-c_j)(1-x_j)\right]\right)\right.$$

$$\times\left(\sum_{i=1}^{k} c_i^2 \frac{\partial^2 f_0}{\partial \xi_i^2} + \sum_{i=1}^{k}\sum_{\text{All } h\neq i} c_h c_i \frac{\partial^2 f_0}{\partial \xi_h \partial \xi_i}\right)$$

$$\left. - \left(\sum_{i=1}^{k} x_i^2 \frac{\partial^2 f_0}{\partial \xi_i^2} + \sum_{i=1}^{k}\sum_{\text{All } h\neq i} x_h x_i \frac{\partial^2 f_0}{\partial \xi_h \partial \xi_i}\right)\right\} + O(\mu^3)$$

$$= \frac{\mu^2}{2}\left\{\sum_{i=1}^{k}(x_i - x_i^2)\frac{\partial^2 f_0}{\partial \xi_i^2}\right\} + O(\mu^3). \tag{31}$$

Here, we make the usual and justifiable assumption that the lattice constant μ is *small*.

Now, turning to the simplicial interpolation (13), in the particular case determined by (23) and (24), we see that (for $s = 0, 1, 2, \ldots, k$) P_s has components '1' in coordinates with indices r_t ($1 \le t \le s$) and components '0' in all other coordinates [e.g., $P_0 = (0, 0, \ldots, 0)$; $P_1 = (0, \ldots, 1, \ldots, 0)$, with the single '1' in position r_1; $P_2 = (0, \ldots, 1, \ldots, 1, \ldots, 0)$, with the two '1' in positions r_1 and r_2; and $P_k = (1, 1, \ldots, 1)$]. Thus, by (28),

$$\Psi_{\boldsymbol{x}} = F(P_0) + \sum_{s=1}^{k} x_{r_s}\left[F(P_s) - F(P_{s-1})\right] = f_0 + \sum_{s=1}^{k} x_{r_s}\left(f_{\boldsymbol{c}_s} - f_{\boldsymbol{c}_{s-1}}\right)$$

$$= f_0 + \sum_{s=1}^{k}\left\{\mu\, x_{r_s}\frac{\partial f_0}{\partial \xi_{r_s}} + \frac{\mu^2}{2} x_{r_s}\left(\frac{\partial^2 f_0}{\partial \xi_{r_s}{}^2} + \sum_{\text{All } t\neq s}\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right)\right\} + O(\mu^3)$$

$$= f_0 + \sum_{s=1}^{k}\left\{\mu\, x_{r_s}\frac{\partial f_0}{\partial \xi_{r_s}} + \frac{\mu^2}{2} x_{r_s}\left(\frac{\partial^2 f_0}{\partial \xi_{r_s}{}^2} + 2\sum_{t=1}^{s-1}\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right)\right\} + O(\mu^3). \tag{32}$$

Recalling that the r_t form a permutation of $\{1, 2, \ldots, k\}$—see (17)—we observe that, by (26), reordered as in (18),

$$\Psi_x - f_x = \frac{\mu^2}{2}\sum_{s=1}^{k} x_{r_s}\left\{\left[\frac{\partial^2 f_0}{\partial \xi_{r_s}{}^2} + 2\sum_{t=1}^{s-1}\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right] - \left[x_{r_s}\frac{\partial^2 f_0}{\partial \xi_{r_s}{}^2} + 2\sum_{t=1}^{s-1} x_{r_t}\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right]\right\} + O(\mu^3)$$

$$= \frac{\mu^2}{2}\sum_{s=1}^{k} x_{r_s}\left\{(1 - x_{r_s})\frac{\partial^2 f_0}{\partial \xi_{r_s}{}^2} + 2\sum_{t=1}^{s-1}(1 - x_{r_t})\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right\} + O(\mu^3), \tag{33}$$

where we note that, by (18),

$$(\forall t \mid t < s) \quad 0 \le 1 - x_{r_t} \le 1 - x_{r_s} \le 1. \tag{34}$$

Comparing (31) with (33), we see that the error produced by the simplicial interpolation is of the same order as that produced by the full interpolation on the lattice-cell, namely, $O(\mu^2)$, and if we have a bound

$$(\forall i, j \mid 1 \le i \le k,\ 1 \le j \le k) \quad \left|\frac{\partial^2 f_0}{\partial \xi_i \partial \xi_j}\right| < K, \tag{35}$$

on the second derivatives of F, then, by (31), since, as is easily verified,

$$\max_{0 \le x \le 1} x(1 - x) = \frac{1}{4}, \tag{36}$$

$$|\Phi_x - f_x| < \frac{K\mu^2}{2}\sum_{i=1}^{k} x_i(1 - x_i) + O(\mu^3) \le \frac{kK\mu^2}{8} + O(\mu^3); \tag{37}$$

while, by (33), with (36),

$$
\begin{aligned}
|\Psi_x - f_x| &< \frac{K\mu^2}{2}\sum_{s=1}^{k} x_{r_s}\left[(1-x_{r_s}) + 2\sum_{t=1}^{s-1}(1-x_{r_t})\right] + O(\mu^3) \\
&\le \frac{K\mu^2}{2}\sum_{s=1}^{k} x_{r_s}\left[(1-x_{r_s}) + 2\sum_{t=1}^{s-1}(1-x_{r_s})\right] + O(\mu^3) \\
&= \frac{K\mu^2}{2}\sum_{s=1}^{k}(2s-1)x_{r_s}(1-x_{r_s}) + O(\mu^3) \\
&\le \frac{k^2K\mu^2}{8} + O(\mu^3). \qquad (38)
\end{aligned}
$$

Summing up our results in (31), (33), (37), and (38) we have:

THEOREM 3. *Applying the bounds (35), we have that*

$$
|\Phi_x - f_x| = \frac{\mu^2}{2}\left|\sum_{i=1}^{k}(x_i - x_i^2)\frac{\partial^2 f_0}{\partial \xi_i^2}\right| + O(\mu^3) < \frac{kK\mu^2}{8} + O(\mu^3) \qquad (39)
$$

and

$$
\begin{aligned}
|\Psi_x - f_x| &= \frac{\mu^2}{2}\left|\sum_{s=1}^{k} x_{r_s}\left\{(1-x_{r_s})\frac{\partial^2 f_0}{\partial \xi_{r_s}^2} + 2\sum_{t=1}^{s-1}(1-x_{r_t})\frac{\partial^2 f_0}{\partial \xi_{r_t}\partial \xi_{r_s}}\right\}\right| \\
&< \frac{k^2K\mu^2}{8} + O(\mu^3). \qquad (40)
\end{aligned}
$$

The conclusion is, therefore, that we will obtain comparable accuracy with the much quicker simplicial interpolation if we scale the lattice constant μ by a factor $\sqrt{k}$:

$$
\mu_{\text{SIMPLEX}} = \frac{\mu_{\text{LATTICE}}}{\sqrt{k}}. \qquad (41)
$$

Of course, if this means actually computing $k^{k/2}$ times as many data, there is little advantage in the simplicial interpolation, but this is not always a requirement; it may be that already-assembled data suffice to yield the required accuracy, and then, as we shall see, the simplicial interpolation is faster to execute.

4. Timing Estimates

At this point, we assume that the values of the function F at the lattice points are already computed and given. The full interpolation (9) then takes three operations:

(i) Determination of the p_j and x_j, for $j = 1, 2, \ldots, k$, by (4); i.e., identification of the particular lattice cell in which the point $(\xi_1, \xi_1, \ldots, \xi_k)$ lies.

(ii) Computation of the 2^k coefficients

$$\prod_{j=1}^{k} \left[c_j x_j + (1 - c_j)(1 - x_j)\right], \tag{42}$$

as defined in (8) and (9). The factors are each either x_j or $1 - x_j$, taken in all combinations.

(iii) Evaluation of the interpolated value Φ_x from (9).

Let **a** denotes the time required for an *addition* or *subtraction*, **m** denotes the time required for a *multiplication* or *division*, and **t** denotes the relatively short time required for a *test*, *shift*, *store*, or *retrieve* operation. Then the operation (i) requires, for each coordinate, two retrievals, one division, two split operations, and two storage operations (obtaining the integer and fractional parts), for a total time

$$\mathbb{T}_{k\text{-CUBE(i)}} = k(\mathbf{m} + 6\mathbf{t}). \tag{43}$$

To perform the operation (ii), we must, for greatest efficiency, proceed inductively. First, we retrieve and store x_1, subtract it from 1 and store $1 - x_1$: this takes time $\mathbf{a} + 3\mathbf{t}$. If the total time for the operation (ii) for $k = j - 1$ is $\mathbb{T}_{(j-1)\text{-CUBE(ii)}}$, then we know that $\mathbb{T}_{1\text{-CUBE(ii)}} = \mathbf{a} + 3\mathbf{t}$. Also, to get $\mathbb{T}_{j\text{-CUBE(ii)}}$, we must first retrieve x_j, subtract it from 1, and, for each of the 2^{j-1} stored coefficients, multiply it by $1 - x_j$ and store the result; this takes time $2^{j-1}\mathbf{m} + \mathbf{a} + (2^{j-1} + 1)\mathbf{t}$. Then we must retrieve x_j again and repeat the multiplications by it, taking time $2^{j-1}\mathbf{m} + (2^{j-1} + 1)\mathbf{t}$ more. Thus,

$$\left.\begin{aligned} \mathbb{T}_{1\text{-CUBE(ii)}} &= \mathbf{a} + 3\mathbf{t} \\ \mathbb{T}_{j\text{-CUBE(ii)}} &= \mathbb{T}_{(j-1)\text{-CUBE(ii)}} + 2^j\mathbf{m} + \mathbf{a} + (2^j + 2)\mathbf{t} \end{aligned}\right\}. \tag{44}$$

It follows that

$$\begin{aligned} \mathbb{T}_{k\text{-CUBE(ii)}} &= \sum_{j=2}^{k} [2^j\mathbf{m} + \mathbf{a} + (2^j + 2)\mathbf{t}] \ + \ \mathbf{a} + 3\mathbf{t} \\ &= (2^{k+1} - 4)\mathbf{m} + k\mathbf{a} + (2^{k+1} + 2k - 3)\mathbf{t} \\ &\sim 2^{k+1}(\mathbf{m} + \mathbf{t}). \end{aligned} \tag{45}$$

Finally, to perform the operation (iii) takes additional time

$$\begin{aligned} \mathbb{T}_{k\text{-CUBE(iii)}} &= 2^k(\mathbf{m} + \mathbf{t}) + (2^k - 1)(2\mathbf{t} + \mathbf{a}) \\ &\sim 2^k(\mathbf{m} + 3\mathbf{t}). \end{aligned} \tag{46}$$

as is easily verified. Thus, the total time for full interpolation is

$$\begin{aligned} \mathbb{T}_{k\text{-CUBE}} &= \mathbb{T}_{k\text{-CUBE(i)}} + \mathbb{T}_{k\text{-CUBE(ii)}} + \mathbb{T}_{k\text{-CUBE(iii)}} \\ &= 2^k(3\mathbf{m} + 5\mathbf{t}) + k(\mathbf{m} + \mathbf{a} + 8\mathbf{t}) - (4\mathbf{m} + 3\mathbf{t}) \\ &\sim 2^k(3\mathbf{m} + 5\mathbf{t}). \end{aligned} \tag{47}$$

Turning to simplicial interpolation (13), we see that there are also three operations:

(i) Determination of the p_j and x_j, for $j = 1, 2, \ldots, k$, by (4), exactly as for full interpolation.

(ii) Ordering of the coordinates x_j, as indicated in (18).

(iii) Evaluation of the interpolated value Ψ_x from (13).

Proceeding as for full interpolation, we first see that

$$\mathbb{T}_{k\text{-SIMPLEX(i)}} = k(\mathbf{m} + 6\mathbf{t}). \tag{48}$$

It is well-known that, if we use an efficient sorting method (e.g., heap-sort, merge-sort, or quick sort), we can achieve [see D. E. KNUTH, *The Art of Computer Programming, Vol. 3, Sorting and Searching* (Addison-Wesley, Reading, MA, 1973) p. 149]

$$\mathbb{T}_{k\text{-SIMPLEX(ii)}} = (18 \log_2 k + 38)k\,\mathbf{t}. \tag{49}$$

Finally, the first line of (32) indicates that

$$\mathbb{T}_{k\text{-SIMPLEX(iii)}} = k(\mathbf{m} + 2\mathbf{a} + 5\mathbf{t}). \tag{50}$$

Thus, the total time for simplicial interpolation is

$$\begin{aligned}\mathbb{T}_{k\text{-SIMPLEX}} &= \mathbb{T}_{k\text{-SIMPLEX(i)}} + \mathbb{T}_{k\text{-SIMPLEX(ii)}} + \mathbb{T}_{k\text{-SIMPLEX(iii)}} \\ &= 18k(\log_2 k)\mathbf{t} + k(2\mathbf{m} + 2\mathbf{a} + 49\mathbf{t}) \\ &\sim 18k(\log_2 k)\mathbf{t}. \end{aligned} \tag{51}$$

Summing up these results, we have:

THEOREM 4. *With timing constants,* $\mathbf{t}$, $\mathbf{a}$, *and* $\mathbf{m}$, *defined as above, we have that*

$$\begin{aligned}\mathbb{T}_{k\text{-CUBE}} &= 2^k(3\mathbf{m} + 5\mathbf{t}) + k(\mathbf{m} + \mathbf{a} + 8\mathbf{t}) - (4\mathbf{m} + 3\mathbf{t}) \\ &\sim 2^k(3\mathbf{m} + 5\mathbf{t}) \end{aligned} \tag{52}$$

and

$$\begin{aligned}\mathbb{T}_{k\text{-SIMPLEX}} &= 18k(\log_2 k)\mathbf{t} + k(2\mathbf{m} + 2\mathbf{a} + 49\mathbf{t}) \\ &\sim 18k(\log_2 k)\mathbf{t}. \end{aligned} \tag{53}$$

Evidently, for large values of k, the simplicial interpolation is far more efficient than the full lattice-cell interpolation. Some comparative examples, for somewhat typical timing constants $\mathbf{t} = 1$, $\mathbf{a} = 8$, and $\mathbf{m} = 24$, and for dimensions $k = 2, 3, 4, 6, 8, 16, 32, 128, 1024$, are given below.

k	$\mathbb{T}_{k\text{-CUBE}}$	$\mathbb{T}_{k\text{-SIMPLEX}}$	RATIO
2	289	262	0.907
3	637	424.6	0.667
4	1,293	596	0.461
6	5,069	957.2	0.189
8	19,933	1,336	0.067
16	5,046,813	2,960	5.865×10^{-4}
32	3.307×10^{11}	6,496	1.964×10^{-8}
128	2.620×10^{40}	30,592	1.168×10^{-36}
1024	1.384×10^{310}	300,032	2.168×10^{-305}