# IMPROVED ROUNDING FOR SPLINE COEFFICIENTS AND KNOTS

ERIC GROSSE AND JOHN D. HOBBY

Abstract. When representing the coefficients and knots of a spline using only small integers, independently rounding each infinite precision value is not the best strategy. We show how to build an affine model for the error expanded about the optimal full-precision free-knot or parameterized spline, then use the Lovász basis reduction algorithm to select a better rounding. The technique could be used for other situations in which a quadratic error model can be computed.

## 1. Introduction

We introduce a technique that improves on componentwise rounding in the context of compact representation of approximations. Although the method is potentially applicable to a variety of approximation schemes, for concreteness we look at free-knot and parametric splines. How can one convert a spline with real coefficients $x_0$ to one with (small) integer coefficients $x_I$, efficiently and with little distortion? Our solution is:

(1) Find the $x_0 \in \mathbb{R}^n$ minimizing $\|r(x)\|_2$. Parameters such as spline coefficients and knots are denoted by $x$, the residual vector or function by $r(x)$, and the corresponding discrete or integral $L_2$ norm by $\|r(x)\|_2$. (This minimization may itself be a challenging problem, but is not the principal focus of this paper.)
(2) Construct a matrix $R$ so that $\|R(x - x_0)\| \approx \|r(x) - r(x_0)\|$ about the minimum.
(3) Use the Lovász basis reduction algorithm to find an integer matrix $M$ such that $RM$ is approximately orthogonal and $M^{-1}$ is also integer.
(4) Using a heuristic, snap $y_0 = M^{-1}x_0$ to a nearby $y_I \in \mathbb{Z}^n$ and let $x_I = My_I$.

Here we denote the set of integers by $\mathbb{Z}$ and the set of reals by $\mathbb{R}$.

The method takes advantage of the relative insensitivity of splines to their knot locations. It compensates for a rounding error introduced in a coefficient by rounding a knot in the balancing direction. Suppose a scalar function on the interval $[0, 1]$ is to be approximated by a spline with coefficients and knots represented in

1991 *Mathematics Subject Classification.* Primary 65D07; Secondary 11H55, 52C07, 65G05, 68R99, 68U05.

*Key words and phrases.* free-knot splines; parameterized splines; rounding; closest lattice point; compression.

Submitted August 9, 1991 and revised July 9, 1993. To appear in *Mathematics of Computation, 1994*. This copy was typeset January 30, 1994.

12-bit fixed-point values. The order and number of degrees of freedom of the spline are assumed given; a dozen piecewise cubics would be typical. In such a case, the $l_2$ error for the best full-precision floating point spline might be $10^{-6}$, the error for our procedure $10^{-5}$, and the error for simple rounding $10^{-4}$. More generally, the method is worthwhile whenever there is a gap between the full-precision spline error and the error for simple rounding.

The paper is organized as follows. To make apparent which part of the technique is specific to splines and which might be applied to other situations in which floating point structures are to be printed, we postpone to Section 4 the specific approximation problem. In Section 2, we show how to locally approximate a generic nonlinear least squares problem $\min \|r(x)\|_2$ by an affine model $\|Rx - b\|_2$. In Section 3, algorithms are described for minimizing this affine model subject to the constraint that $x \in \mathbb{Z}^n$. In Section 4, the affine model is adjusted to better fit the region which the integer optimization has identified as relevant. In Section 5, we show that the method pays by applying it to several test functions and observing the improvement over simple rounding. Section 6 uses the same idea on parametric splines, for which a different construction of the model $\|Rx - b\|_2$ is necessary. Finally, Section 7 applies this to the problem of compressing maps and fonts.

Because this problem may be of interest to distinct audiences from continuous and discrete mathematics, we have attempted to include enough details to make the paper reasonably self-contained. The issue of nontrivial rounding strategies in representing geometric objects has not had much attention in the literature. The simple decomposition of Section 2 is surely not new, but we could not find it in the literature. Section 3 relates the literature on closest lattice point problems [1, 10, 13, 14, 19] to the rounding application. Determining the optimal free-knot spline has received considerable attention [3, 7, 16] and we take advantage of that work. Some details of our spline optimization, B-spline differentiation, and Hausdorff distance computation may be of interest even to people working without integer constraints.

## 2. Improved affine model

Given a nonlinear least squares problem with residual vector $r(x)$, the traditional local linear least squares, or "affine", model is the Gauss-Newton approximation

$$\frac{1}{2}\|r(x_0+\delta)\|_2^2 \approx \frac{1}{2}\|J\delta + r(x_0)\|_2^2 \tag{1}$$

where $J = Dr(x_0)$ is the Jacobian matrix of first derivatives. The full Taylor quadratic expansion about $x_0$ requires a matrix of second partial derivatives, $H_i = D^2 r_i(x_0)$, for each component of the residual vector $r(x)$. If we let

$$A = J^T J + \sum_i r_i(x_0) H_i \tag{2}$$

$$b = r^T(x_0) J \tag{3}$$

$$c = \frac{1}{2}\|r(x_0)\|_2^2 \tag{4}$$

then the quadratic model is

$$\frac{1}{2}||r(x_0+\delta)||_2^2 \approx c + b^T\delta + \frac{1}{2}\delta^T A\delta \tag{5}$$

We intend to expand near a minimum of $r$, so $b$ should be small and $A$ positive semidefinite. The point of this section is to show how this higher order model can also be written in affine form.

Let

$$A = QS^2Q^T \tag{6}$$

where $S$ is a diagonal matrix containing the eigenvalues and $Q$ is an orthogonal matrix whose columns are eigenvectors of $A$. Define

$$R = SQ^T \tag{7}$$

$$d = -S^{-1}Q^T b. \tag{8}$$

Then the quadratic model can be rewritten as

$$c + b^T\delta + \frac{1}{2}\delta^T A\delta = \frac{1}{2}||R\delta - d||_2^2 + c - \frac{1}{2}||d||_2^2. \tag{9}$$

This affine model has a higher order of accuracy than the Gauss-Newton one.

Spline fitting with free-knots leads to very ill-conditioned systems, hence some of the eigenvalues may be negative in practice. We return to this issue in Section 4.2. We use eigenvalues because they present full information about the model in a convenient form, though the matrix square root could be computed less expensively by a Cholesky factorization. In our application, the cost of the linear algebra is small compared to the cost of finding $x_0$.

## 3. Adding integer constraints

The interesting part of the problem is to find an integer vector $x$ that, given an $n$ by $n$ matrix $R$ and an $n$-vector $x_0$, minimizes

$$||R(x - x_0)||_2 \tag{10}$$

or at least comes close to the minimum. (The requirement $x \in \mathbb{Z}^n$ is equivalent, after scaling, to the requirement that $x \in \mathbb{R}^n$ be represented in limited precision.) In other words, we need to find an integer combination of the columns of $R$ that is close to $Rx_0$. The set of all such linear combinations is a subset of real $n$-space $\mathbb{R}^n$ closed under addition. Such subsets are called *integer lattices* and the problem of minimizing $||R(x - x_0)||_2$ is called the *closest lattice point problem.*

Van Emde Boas has shown that this problem is NP-complete [19], but good approximate solutions can often be found using the Lovász lattice basis reduction algorithm[10]. The idea is to use the Lovász algorithm to find an alternative representation of the lattice that makes it easier to find a lattice point close to $Rx_0$. Originally, the lattice is represented by the basis formed by the columns of $R$. The result of the Lovász algorithm is a matrix whose columns form a new basis for the same lattice. Thus there is an integer matrix $M$ such that the new matrix is $RM$ and $M^{-1}$ is also an integer matrix. This means that a lattice vector $Rx$ can be written $RMy$ where $y = M^{-1}x$ is an integer vector if and only if $x$ is. The output

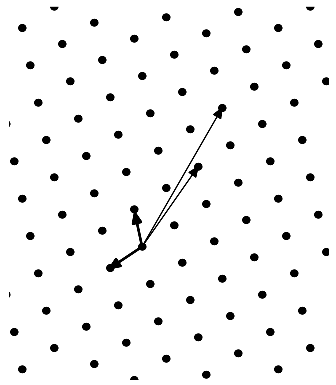

FIGURE 1. Before (long thin arrows) and after (short thick arrows) the Lovász lattice basis reduction.

of the algorithm applied to a simple example with $n = 2$ is shown in Figure 1. By "improved basis", we mean that the set $R\mathbb{Z}^n$ is the same as $RM\mathbb{Z}^n$ (the dots in the figure) and the columns of $RM$ (the thick arrows) are shorter and more orthogonal than those of $R$ (the thin arrows).

The Lovász algorithm reduces the problem of minimizing (10) to minimizing

$$\|RM(y - y_0)\|_2, \tag{11}$$

where $y_0 = M^{-1}x_0$ and $x = My$. This is easier than trying to minimize (10) directly because simple strategies for choosing $y$ produce relatively good values for (11). Even if $y$ is obtained by rounding each component of $y_0$ to the nearest integer without regard to $RM$, Babai is able to give an upper bound on the resulting value of (11).[1]

Babai also analyzes a better strategy called Nearest Plane. He shows that it finds a lattice point $RMy$ where $\|RM(y - y_0)\|_2$ is within a factor of $2^{n/2}$ of the best possible. That is if $RM$ is the reduced basis matrix produced by the Lovász algorithm, Nearest Plane produces an integer vector $y$ such that

$$\|RM(y - y_0)\|_2 \leq 2^{n/2}\|RM(y' - y_0)\|_2$$

for all integer vectors $y'$. Thus $x = My$ is an equally good solution of (10).

Being within a factor of $2^{n/2}$ of optimal may not sound very good, but the bound turns out to be very pessimistic in practice and there are a number of ways to improve on the algorithm that Babai analyzed. These possible improvements are best understood after examining the Lovász algorithm in more detail in the next part of this section. Section 3.2 then presents Babai's Nearest Plane algorithm and discusses the improvements.

**3.1. The Lovász Algorithm.** We need a version of the algorithm that produces the integer matrix $M$ and is appropriate for use with floating point arithmetic. This is significantly different from Schnorr's work on using floating point arithmetic with the Lovász algorithm since he assumes that the matrix $R$ is given in fixed point and the result is to be computed exactly.[14, 15]

**procedure** *reduce*$(l, k)$;
{ $r := \lceil \bar{R}_{l,k} \rfloor$;
$M_{\cdot,k} \mathrel{-}= r M_{\cdot,l}$;
$\bar{R}_{\cdot,k} \mathrel{-}= r \bar{R}_{\cdot,l}$;
}

**procedure** *swap*$(k)$;
{ $\rho := \bar{R}_{k-1,k}$;
Exchange $M_{\cdot,k-1}$ and $M_{\cdot,k}$;
Exchange $\bar{R}_{\cdot,k-1}$ and $\bar{R}_{\cdot,k}$;
$c := d_k + \rho^2 d_{k-1}$;
$\rho' := \rho d_{k-1}/c$;
$d_k := d_k d_{k-1}/c$;
$d_{k-1} := c$;
$\bar{R}_{k-1:k,\cdot} := \left(\begin{smallmatrix} 1 & \rho' \\ 0 & 1 \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 & 1 \\ 1 & -\rho \end{smallmatrix}\right)\bar{R}_{k-1:k,\cdot}$;
}

FIGURE 2. Subroutines used by the Lovász algorithm

The major variables are an index $k$, a unit upper-triangular matrix $\bar{R}$, an integer matrix $M$, and a diagonal matrix $D$ with nonzero entries $d_1, d_2, \ldots, d_n$. These variables satisfy the following invariants: det $|M| = \pm 1$, and there exists an orthogonal matrix $Q$ such that

$$RM = Q\sqrt{D}\,\bar{R}. \tag{12}$$

(This version of the Lovász algorithm does not explicitly compute $Q$ or the product $RM$).

Figure 2 gives a subroutine *reduce*$(l, k)$ that performs a column operation on $M$ and $\bar{R}$ so as to make $|\bar{R}_{l,k}| \le \frac{1}{2}$. It assumes that $l < k$ so that an integer multiple of column $l$ can be subtracted from column $k$ while retaining $\bar{R}$ in unit upper-triangular form. (The meaning of the notation $\lceil \bar{R}_{l,k} \rfloor$ is that $\bar{R}_{l,k}$ is to be rounded the nearest integer).

Figure 2 also gives a subroutine *swap*$(k)$ that exchanges columns $k$ and $k-1$ in both $M$ and $\bar{R}$. To prevent this from affecting elements of $\bar{R}$ on or below the diagonal, rows $k$ and $k-1$ of $\bar{R}$ are then left-multiplied by a matrix

$$\begin{pmatrix} 1 & \rho' \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -\rho \end{pmatrix} \tag{13}$$

chosen so that $\rho = \bar{R}_{k-1,k}$ at the beginning of *swap*$(k)$ and there is an orthogonal matrix of the form

$$\begin{pmatrix} \sqrt{d'_{k-1}} & 0 \\ 0 & \sqrt{d'_k} \end{pmatrix}\begin{pmatrix} 1 & \rho' \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -\rho \end{pmatrix}\begin{pmatrix} 1/\sqrt{d_{k-1}} & 0 \\ 0 & 1/\sqrt{d_k} \end{pmatrix} \tag{14}$$

where $d'_k$ and $d'_{k-1}$ are the new values of $d_k$ and $d_{k-1}$ in *swap*$(k)$. Thus left-multiplying $\sqrt{D}\,\bar{R}$ by (14) does not affect the invariant that there is an orthogonal matrix $Q$ satisfying (12). This is equivalent to left-multiplying $\bar{R}$ by (13) and updating $d_k$ and $d_{k-1}$.

```
procedure Lovász(R, α);
{   Use modified Gram-Schmidt to find D and R̄ with R = Q√D R̄;
    Initialize M to be the identity matrix;
    k := 2;
    while k ≤ n
    do { reduce(k − 1, k);
         if d_k < (α − R̄²_{k−1,k}) d_{k−1}
         then { swap(k); k := max(k − 1, 2); }
         else { for l := k − 2, k − 3, ... , 1
                do reduce(l, k);
                k := k + 1;
              }
       }
}
```

FIGURE 3. A version of the Lovász algorithm that computes matrices $M$, $D$, and $\bar{R}$ as in (12).

The version of the Lovász algorithm in Figure 3 uses the *reduce* and *swap* routines to manipulate the matrices $D$, $\bar{R}$, and $M$ while maintaining the invariants. Its arguments are an $n \times n$ matrix $R$ and a real parameter $\alpha$ that can be chosen in the range $\frac{1}{4} < \alpha < 1$ to control the tradeoff between running time and the quality of the results.

When the Lovász algorithm stops, all off-diagonal entries of $\bar{R}$ are between $-\frac{1}{2}$ and $\frac{1}{2}$ and the entries of $D$ satisfy

$$d_{k-1} \leq \frac{d_k}{\alpha - \frac{1}{4}}$$

for $k = 2, 3, \ldots, n$. (See [10].) Hence the ratio $d_{k-i}/d_k$ is at most $(\alpha - \frac{1}{4})^{-i}$ for all $i$ less than $k$. This upper bound is $2^i$ when $\alpha = \frac{3}{4}$, but it is tightened to $(\frac{4}{3})^i$ when $\alpha$ approaches one.

The cost of tightening the bounds by increasing $\alpha$ is that this increases the running time. Lovász *et al.* bound the running time by showing that each call to $swap(k)$ reduces the product

$$\prod_{i=1}^{n} d_i^{n-i} \tag{15}$$

to at most $\alpha$ times its former value. In practice, the reduction factor is likely to be better than this, especially if $\alpha \approx 1$ so the running time for $\alpha = 0.9999$ is only about three times that for $\alpha = \frac{3}{4}$.

**3.2. Babai's Nearest Plane Algorithm.** The nearest lattice point problem involves finding an integer vector $y$ that $RMy$ is near $RMy_0$ in Euclidean $n$-space, given a vector $y_0$ and matrices $R$ and $M$. Babai suggests that if $RM$ is produced by the Lovász algorithm, good results can be obtained by fixing the components of $y$ one at a time starting with $y_n$. When choosing $y_k$, it is a good idea to take the choices for $y_{k+1}$, $y_{k+2}$, …, $y_n$ into account rather than just looking at the $n$th

$Lovász(R,\, 0.9999)$;
Solve $My_0 = x_0$ for $y_0$;
$w := \bar{R}y_0$;
**for** $i := n,\, n-1,\, \ldots,\, 1$
**do** $y_i := w_i - \left\lceil \sum_{j=i+1}^{n} \bar{R}_{i,j} y_j \right\rfloor$;
$x := My$;

FIGURE 4. How to find an integer vector $x$ that makes $\|R(x - x_0)\|_2$ small.

entry in $y_0$. Thus the Nearest Plane algorithm chooses $y_k$ according to the point $RMy$ nearest to $RMy_0$ on the plane determined by restricting $y_{k+1}, y_{k+2}, \ldots, y_n$ according to their chosen values.

Using (12), the norm of $RM(y - y_0)$ becomes

$$\|\sqrt{D}\,\bar{R}(y - y_0)\|_2 \tag{16}$$

The advantage of this is that $\bar{R}$ is unit upper triangular and Babai's Nearest plane algorithm reduces to doing back substitution with each element of $y$ rounded to the nearest integer as soon as it is computed. Writing $w = \bar{R}y_0$ for the right-hand side, $y$ can be computed as follows:

**for** $i := n,\, n-1,\, \ldots,\, 1$
**do** $y_i := w_i - \left\lceil \sum_{j=i+1}^{n} \bar{R}_{i,j} y_j \right\rfloor$

This produces a vector $y$ that makes (16) small, and via the relation $x = My$, an $x$ that makes (10) just as small.

Figure 4 gives the complete strategy for finding an integer vector $x$ that makes (10) small. It could be speeded up somewhat by starting with $w = Rx_0$ and modifying *Lovász* to maintain $w = \bar{R}M^{-1}x_0$. This eliminates the need to solve $My_0 = x_0$ and compute $\bar{R}y_0$. The maintenance of $w$ involves adding

$$w_{k-1:k} := \begin{pmatrix} 1 & \rho' \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\rho \end{pmatrix} w_{k-1:k}$$

at the end of *swap*, and changing the second line of *Lovász* from

$$\textbf{do } \{\; d_i := \bar{R}_{i,i}^2;\; \bar{R}_{i,\cdot} := \bar{R}_{i,\cdot}/\bar{R}_{i,i};\; \}$$

to

$$\textbf{do } \{\; d_i := \bar{R}_{i,i}^2;\; \bar{R}_{i,\cdot} := \bar{R}_{i,\cdot}/\bar{R}_{i,i};\; w_i := w_i/\bar{R}_{i,i} \}$$

Once the Nearest Plane algorithm has been implemented efficiently, we need to know how small it makes (16). Babai shows in [1] that it is within a factor $2^{n/2}$ of the best possible when the Lovász algorithm is run with $\alpha = \frac{3}{4}$. Using the recommended value $\alpha = 0.9999$ can only improve the bound since the effect of this change is to tighten the constraints on the final values of the entries of $D$ from $d_{k-i}/d_k \le 2^i$ to $d_{k-i}/d_k \le (\frac{4}{3})^i$. In fact, Babai's proof readily generalizes to give a bound of

$$\frac{\beta^{n/2}}{\sqrt{\beta - 1}}, \quad \text{where } \beta = \frac{1}{\alpha - \frac{1}{4}}.$$

Thus Nearest Plane is guaranteed to get within a factor of $1.732 \times 1.334^{n/2}$ of the optimum value of (16).

Experiments with the test problems given in Section 5 show that Nearest Plane combined with the Lovász algorithm usually reduces (16) to within 20% of the optimum value for $n \leq 20$. Thus there probably is not much to be gained by using more elaborate algorithms, although Schnorr[13] does suggest a family of lattice reduction algorithms that are theoretically superior to Lovász's. Lagarias, Lenstra and Schnorr[8] also discuss a number of issues relevant to the problem of finding near optimal solutions to the nearest lattice point problem. See also LaMacchia[9] for a comparison of Lovász and Seysen basis reduction.

Another option for small $n$ is brute force search. Experiments showed that with clever pruning, exhaustive search can find the optimum when, say, $n < 20$, but the running time can be huge and the payoff is usually small.

## 4. $l_2$ SPLINE WITH FREE-KNOTS

Imagine that you are on a deep space probe having just measured a function of some sort and fit it by a spline. You wish to transmit it over a communications channel with severely restricted bandwidth. Rather than sending double or even single precision floating point coefficients, you can afford to convert to integers in order to achieve better compression of the data.

We wish to approximate a general smooth $f$ by a spline $\sum a_j B_j$, which may be generically described by: $n$, the number of degrees of freedom; $k$, the order (k=4 for a piecewise cubic); a knot sequence $\{t_i\}_{1 \leq i \leq n+k}$; B-spline coefficients $\{a_j\}_{1 \leq j \leq n}$.

Let $\hat{a}_j$, $\hat{t}_j$ be the specific knots and coefficients that approximately minimize

$$\sum_{1 \leq i \leq m} \left( f(x_i) - \sum a_j B_j(x_i) \right)^2 \tag{17}$$

for some fixed sample points $\{x_i\}_{1 \leq i \leq m}$. Let $\hat{B}_j$ be the $n$-vector of B-splines on the knots $\hat{t}_j$. Denote by $B_a$ the $m$ by $n$ matrix $[\hat{B}_j(x_i)]$. Denote by $B_t$ the $m$ by $n-k$ matrix

$$\left[ \frac{\partial}{\partial t_j} \sum_l \hat{a}_l \hat{B}_l(x_i) \right], \tag{18}$$

where $j$ ranges over the free-knots, $k+1 \leq j \leq n$. Taylor expansion around $\hat{a}_j$, $\hat{t}_j$ gives a linearized residual

$$\left\| [B_a B_t] \begin{bmatrix} a \\ t \end{bmatrix} - [f(x) + B_t \hat{t}] \right\|_2^2 \tag{19}$$

as in equation (1).

Jupp [7] has ably described the computational considerations in determining $\hat{a}_j$, $\hat{t}_j$. See also [4]. For the purpose of our experiments, we have used Schryer's **ssaf** [16] (which is based on de Boor's **newnot** [3]) to get a good initial guess to $\hat{t}_j$. **ssaf** only attempts to get in the general vicinity of the optimum; further effort would not lead to a dramatically smaller residual. In contrast, since we need the maximum flexibility in moving knots, it is important that $\hat{t}_j$ be close to the continuous optimum. Therefore we use Gay's [5] nonlinear least squares program **n2f** to refine the **ssaf** results. For the present purposes, just assume $\hat{t}_j$ given.

**4.1. Derivatives of B-splines.** It is apparent that a key building block for the quadratic model is $\partial B_l/\partial t_j$. We compute these by differentiating the recurrence relation for the B-splines,

$$B_{i,k}(u) = \frac{u - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i-1}} B_{i+1,k-1}(u). \tag{20}$$

(There is also a formula using divided differences of truncated powers, given in Lemma 3.1 of [7] and Theorem 4.27 of [17].) In practical terms, this just means editing `bsplvb` in de Boor's `pppack` collection of spline routines. For each line of code that contributes to the B-spline table, add a loop to accumulate derivatives. In other words, apply the chain rule of differentiation directly to the Fortran. This has been done and is available by e-mail:

```
mail netlib@research.att.com
send dbspvt from a
```

We have not studied stability of the algorithm, but it meets our needs in this application. On the half-open interval $[t_l, t_{l+1}[$, the nonzero B-splines are $B_{l-k+1}$, $\ldots$, $B_l$. These depend on $t_{l-k+1}$, $\ldots$, $t_{l+k-1}$. So in addition to the

$$\texttt{biatx(j)} = B_j(u), \qquad 1 \le j \le k \tag{21}$$

returned by `bsplvb`, our subroutine also returns

$$\texttt{dbiatx(j,i)} = \frac{\partial B_{l-k+j}(u)}{\partial t_{l-k+i}}, \qquad 1 \le j \le k, 1 \le i \le 2k. \tag{22}$$

It would be possible to proceed this way for the second derivatives needed to build the Taylor quadratic, but for convenience we use centered differences with step size $2\epsilon_{mach}^{1/3}$ and symmetrize by $(H_i + H_i^T)/2$.

**4.2. Adjusting the local model.** We want the quadratic model (5) to be valid over the region that the lattice algorithms search. In Figure 5, the solid curve indicates $\|r(x)\|^2/2 =: r^2$ in the vicinity of $x_0$, sliced along the eigenvector of $A$ corresponding to the smallest eigenvalue. The dashed line shows the quadratic model, adjusted as described in this section. Notice that $r^2$ is behaving more like a quartic than a quadratic, so a quadratic model based purely on derivative information at $x_0$ would be too flat. This means that the lattice algorithms will in effect be told that they can make radical changes in coefficients without changing $r^2$ much. For directions corresponding to large eigenvalues, the quadratic model is an excellent fit.

This figure is for function 2 (arcsin) described in Section 5 with $k = 4$, $n = 8$ (cubic spline with four interior knots). In this example, the condition number of A (the ratio of the largest to the smallest eigenvalue) is about $3 \cdot 10^5$; in other examples, the smallest eigenvalue is negative, even though the optimizer succeeded in finding an $x_0$ that is a minimum to graphical accuracy.

To repair these two defects in the model, we would like to build a quadratic based on information somewhat less local than just the derivatives at $x_0$, but making enough samples of $r(x)$ to fit all $(2n-k)^2/2$ coefficients for a better $\hat{A}$ is expensive and unnecessary. Use the eigenvectors of $A$, probing $r(x)$ some distance along those $2n-k$ orthogonal directions, and adjusting only the eigenvalues to get an $\hat{A}$ so the

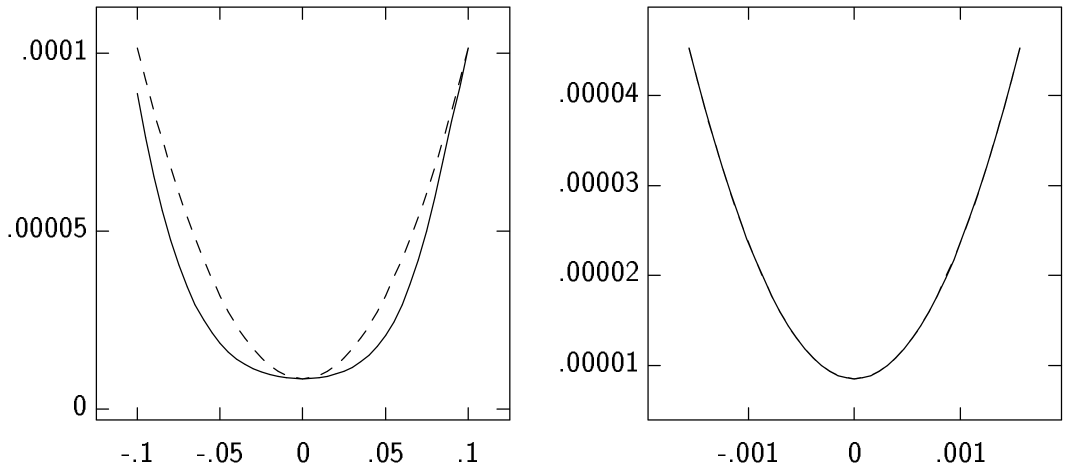


FIGURE 5. Slices of the $r^2$ surface and adjusted affine model along the first and last eigenvectors illustrate the need in certain directions for more than purely local derivative information.

model interpolates $r(x)$ at $x_0$ and the probes. This is easy to do and works well in our tests.

The hardest issue is deciding how far to probe. We aim to have a quadratic model valid over the region searched in the discrete optimization phase. For each eigenvector we initially take a step $\delta_0$ which is the smaller of: 1/10 of interval on which the spline is defined, and 1/2 of distance that would lead to knots coalescing. These are arbitrary parameters designed merely to give a sensible starting model. Because the eigenvectors are orthogonal, it is valid to probe independently in each direction. Let $\bar{r} = r(\lceil x_0 \rfloor)^2$, where $\lceil x_0 \rfloor$ is the result of rounding the components of $x_0$. We then examine $r^2$ at $\delta := \delta_0$ to ensure that $r^2 < \bar{r}$. If not, we cut the step, $\delta := \delta/2$, and iterate. We then adjust eigenvalues so the model is at least $r^2$ for $\delta$ and at least $\bar{r}$ for $\delta_0$. (The latter restriction is necessary to prevent later rounding from coalescing the knots.) After updating the eigenvalues $S$, we form $R = SQ^T$ and then use the rounding process of Section 3.

Section 5 shows that this improved rounding strategy is much better than just rounding the components of $x_0$. This suggests that a better $\bar{r}$ can then be obtained by using the result of improved rounding in place of $\lceil x_0 \rfloor$. Repeating the eigenvalue adjustment and updating $R = SQ^T$ produces a less pessimistic model with which we then repeat the rounding process of Section 3. Section 5 shows that this iterated improvement strategy is somewhat better than improved rounding without the extra iteration.

## 5. NUMERICAL RESULTS FOR FREE-KNOT SPLINES

The complete process of generating rounded spline coefficients and knots was implemented and tested. For lack of any standard collection of test functions in the approximation community, we arbitrarily took the six nonsingular test functions that came along with Schryer's `ssaf` program:

**function 2::** $\frac{1}{2} + \frac{1}{2}\arcsin(-.99 + 1.98x)/\arcsin(.99)$
**function 6::** $(\cosh(2x-1)-1)/(\cosh(1)-1)$

**function 7::** $x^{1.6}$
**function 8::** $(e^{2x-1} - e^{-1})/(e - e^{-1})$
**function 15::** $\frac{1}{2} + \frac{1}{2}\sin 2\pi x$
**function 21::** $-ex \log x$

The functions have been scaled so that $x$ and $f(x)$ are both confined to the interval $[0, 1]$, hence the spline coefficients $a_1, a_2, \ldots, a_n$ and the knots $t_{k+1}, t_{k+2}, \ldots, t_n$ are also $O(1)$. These quantities are to be represented in fixed point as integers scaled by $2^{-b}$ where $b$ is a parameter chosen in advance to control the number of bits of precision to use.

For example, consider the case of function 15 with $b = 10$, $n = 8$, and $k = 3$. Using `n2f` to find the optimum for the continuous problem results in setting the eight spline coefficients

$$a_1, a_2, \ldots, a_8 = 510.18u, 721.17u, 1179.29u, 714.34u, 309.66u, -155.29u, 302.83u, 513.82u$$

and placing the five interior knots

$$t_4, t_5, \ldots, t_8 = 126.24u, 388.875u, 512u, 635.125u, 897.756u.$$

Here $u = 2^{-b}$ is one unit in the last place. Sampling the difference between the resulting spline and function 15 at 115 $x$ values produces a root mean square (RMS) error of $1.045 \times 10^{-3}$.

One way to get fixed-point versions of the knots and spline coefficients is to naïvely round the thirteen numerators to integers, thereby increasing the RMS error to $1.076 \times 10^{-3}$. Creating a quadratic model for this error value as explained in Section 2 and using this to produce improved integer values for the numerators as in Section 3, we find

$$a_1, a_2, \ldots, a_8 = 510u, 723u, 1177u, 720u, 318u, -158u, 303u, 514u$$
$$t_4, t_5, \ldots, t_8 = 127u, 388u, 509u, 633u, 898u$$

for an RMS error of $1.057 \times 10^{-3}$.

Rounding the knots and spline coefficients to fixed point values has little effect on the error in this example because the error in the continuous problem is relatively high. Figures 6 and 7 show what happens to the RMS error when $k$ and $n$ are increased. As $n$ is increased, the error for the continuous optimum spline parameters decreases steadily while the error for the various rounding strategies tends to level off at some point.

Like all error graphs given in this section, these graphs are log-log plots of the RMS error for the various rounding strategies as a function of $n$. The upper line always shows the error for simple rounding, the middle two are for improved rounding and iterated improvement, and the lowest line is always for the continuous optimum.

The error for simple rounding tends to stay near $10^{-3.4}$ whenever $n$ is large enough to make the error for the continuous optimum less than $10^{-4}$. The error at the continuous optimum has to be reduced to about $10^{-5}$ in order to make the error for the more sophisticated rounding strategies flatten out, but then they tend to jump around in the vicinity of $10^{-4.3}$. Thus for data compression applications, it makes sense to choose $n$ near this point where the error graphs first start to flatten out. Larger $n$ would mean having unnecessary knots and spline coefficients, while smaller $n$ would mean sacrificing accuracy.

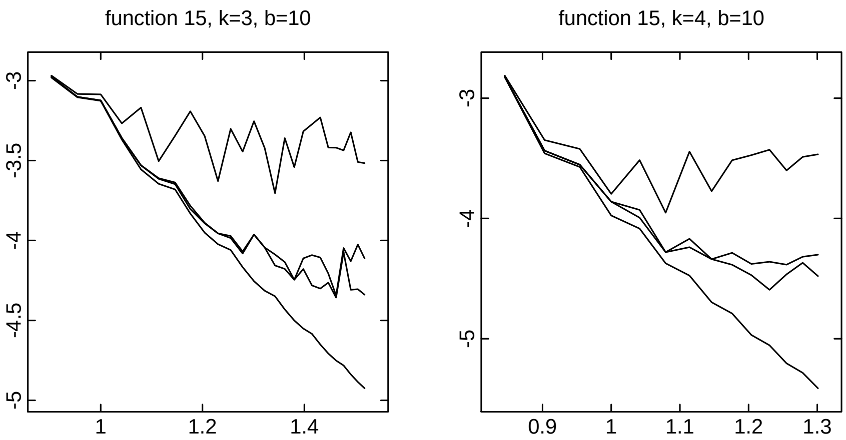


FIGURE 6. The $\log_{10}$(RMS error) in the spline approximation to function 15 as a function of $\log_{10}(n)$ for for $k = 3, 4$. Reading from top to bottom, the four error curves in each graph are for simple rounding, improved rounding, iterated improvement, and the continuous optimum.

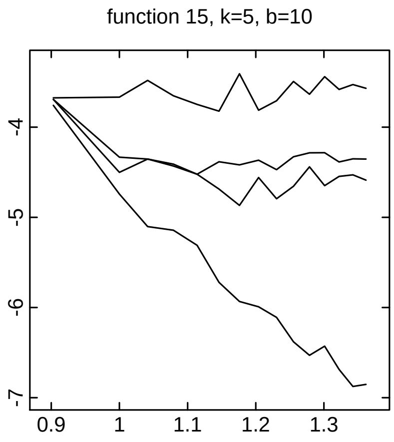


FIGURE 7. The logarithm of RMS error versus $\log_{10}(n)$ for a spline approximation problem as in Figure 6, but with $k = 5$. As in that figure, there is one curve for each rounding strategy and one at the bottom for the continuous optimum.

Figure 7 shows that the primary effect of increasing $k$ is to make the continuous optimum error fall to $10^{-5}$ before $n$ gets very large. Thus a sufficiently smooth test function is represented most compactly when $k$ be as large as possible. Of course there is a time penalty for large $k$ and some applications may impose a priori restrictions on $k$. We shall restrict out attention to the case $k = 4$; Figures 6 and 7 give a general idea of the effect of changing $k$.

We can get a better idea of the error behavior for the various rounding strategies by considering the other test functions with other values of $b$. The error for simple rounding is generally somewhat less than half of $u = 2^{-b}$ once $n$ is large enough but this depends somewhat on the test function. Since $u/2$ is an upper bound on the effect of rounding each spline coefficient, one would normally expect a significantly smaller effect on the value of the spline function obtained by taking a weighted average of four spline coefficients. However, this is offset by the effect of rounding the knot positions $t_{k+1}, t_{k+2}, \ldots, t_n$.

Figure 10 in the supplement gives the RMS error for the various rounding strategies for all six test functions when rounding to eight bits ($b = 8$). All the graphs show a significant amount of noise due to the unpredictable nature of rounding error, but improved rounding is generally about five times better that simple rounding and iterated improvement improves this to eight times better.

The more sophisticated rounding strategies really prove their usefulness when $b$ is increased. As can be seen from Figure 11 the gap between simple rounding and iterated improvement increases to about a factor of 20 when rounding to twelve bits. Since the error for simple rounding does appear to be roughly proportional to $2^{-b}$, this implies that iterated improvement with 12-bit rounding is likely to be better than simple rounding to 16-bits.

What about the running time? It is mainly a function of the number of spline coefficients $n$ and the spline order $k$. Figure 8 gives log-log graphs of the observed running time for the main subtasks for $k = 3, 4, 5$. The predominant subtask labeled "`c`" in the graphs is finding the continuous optimum and constructing the quadratic model. The continuous optimizer `n2f` takes dozens of steps each involving $O(n^3)$ work, and computing eigenvectors for building the quadratic model is also $O(n^3)$. The other three graphs labeled "`u`", "`L`" and "`n`" refer to the time to update the quadratic model and run the Lovász and Nearest Plane algorithms. Since iterative improvement causes each of these tasks to be done twice, their actual contributions to the total running time are roughly twice what is shown but this would not shift the curves very much on the log-log graphs. (All timings were done with double precision floating point on a VAX 8550).

## 6. Parametric splines

Section 4 mentioned the motivation of limited bandwidth; an equivalent situation is limited storage. Returning from the deep space probe to earth, imagine you wish to compress a large database of road coordinates. Subject to the restriction that the map curves remain accurate enough to guide a driver, you wish to fit as much of the country as possible into limited storage space.

We wish to approximate a curve $f : [0,1] \to \mathbb{R}^p$ by a parametric spline $s = \sum a_j B_j$, described by: $n$, the number of degrees of freedom; $k$, the order (k=4 for a piecewise cubic); B-spline control points $\{a_j\}_{1 \le j \le n} \in \mathbb{R}^p$.

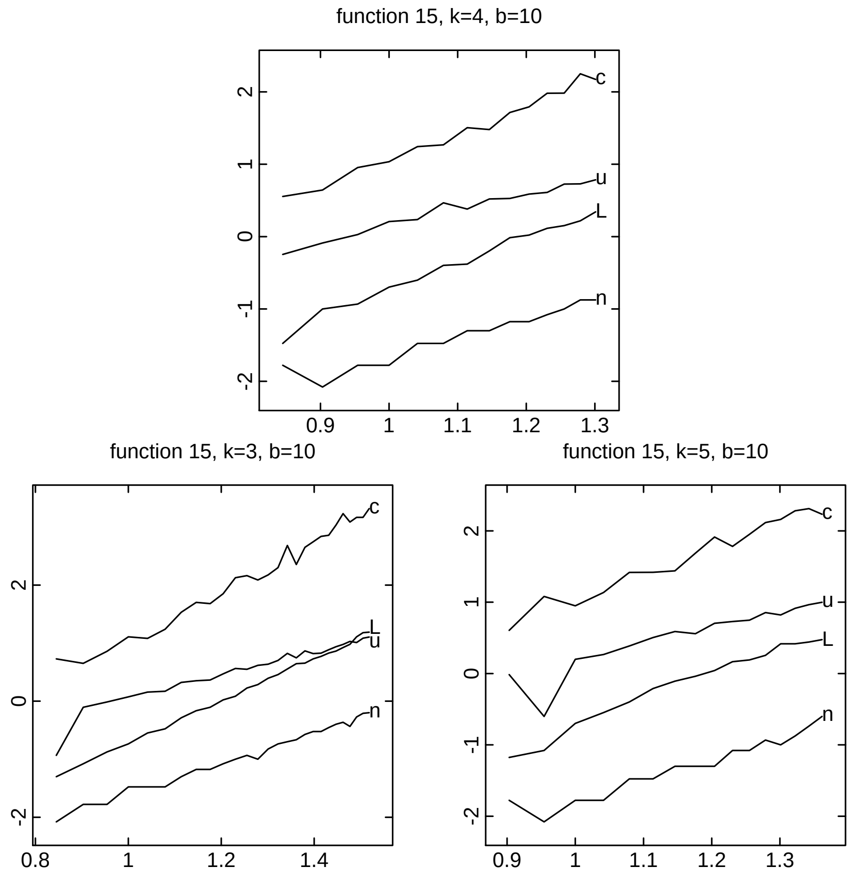


FIGURE 8. The base-ten logarithm of observed running time in seconds versus $\log_{10}(n)$ for the indicated $k$ values. The line labeled "c" is the time for finding the continuous optimum and quadratic model; the "u" line gives the time for updating the model with a new error estimate; and the "L" and "n" lines are for the Lovász and Nearest Plane algorithms.

The relevant measure of error in this approximation is Hausdorff distance between the set of points $f([0,1])$ and $s([0,1])$. This permits considerable freedom in the choice of spline parameterization. For convenience and to avoid having to store any coefficients other than the control points, we use as knot sequence, $\{t_i\}_{1\le i\le n+k}$, the chordal distance along the control point polygon.

To start the continuous optimization, we use the Wall and Danielsson method for fast polygonal approximation[20]. This method assumes $f$ is piecewise linear and produces a coarser piecewise linear curve $g$ meeting a constraint on the area between $f$ and $g$. The vertices of $g$ provide a crude initial guess for $\{a_j\}$. From these and the Hausdorff sampling described next, we can solve a linear least squares problem to get a better initial guess for $\{a_j\}$.

Computing the true Hausdorff distance between $f$ and $s$ is no simple matter [6]. For the experiments reported below, we first took a sample of points along $f$, getting higher density in areas of high curvature by using a fixed number of samples per segment in the piecewise linear $f$. Each time a new $s$ was constructed, we evaluated it at uniformly spaced $t$, made a monotone matching with the samples from $f$, and took one step of Newton's method to refine the "nearest neighbor" estimate. Using spline derivatives up to second order and taking $p = 2$, the Newton step in $t$ that attempts to make the tangent $s'(t)$ be orthogonal to the residual $s(t) - f(x_i)$ is given by

$$t_{\text{new}} = t_{\text{old}} - \frac{(f_1 - s_1)s_1' + (f_2 - s_2)s_2'}{-s_1's_1' + (f_1 - s_1)s_1'' - s_2's_2' + (f_2 - s_2)s_2''}, \tag{23}$$

where $f_1$ means the first component of $f(x_i)$, $s_2'$ is the second component of the derivative of the spline at $t_{\text{old}}$, and so on. This Newton step is safeguarded by checking that it does not increase $s(t) - f(x_i)$ and that it does not step outside the interval on which the spline is defined. Instead of our discrete search followed by one Newton, one could choose to sample at many more points or use more sophisticated univariate optimization.

Let there be $m$ such sample points and denote the neighbor distances by $r_i = \|s(t_i) - f(x_i)\|$. Then $\sum_{1\le i\le m} r_i^2$ is an objective which we minimize by letting Gay's **n2f** adjust the spline control points. To avoid adding a stochastic component to the objective, we started the matching afresh each time, accepting $O(m^2)$ runtime. Since $s$ is changing slowly, it is clear that with care the cost of the Hausdorff estimate could be reduced to $O(m)$.

The final step is to build an affine model, $\|R(x - x_0)\|_2 \approx \|r(x) - r(x_0)\|_2$. Unlike the univariate free-knot case, an analytic Taylor series for $e$ is complicated to obtain, involving the chain rule and derivatives of the spline, the chordal parameterization, and the Hausdorff estimation. Instead, we directly solve for the $mn$ parameters in $R$ by sampling $r \in \mathbb{R}^m$ at $q$ points and formulating a $q$ by $n$ linear least squares problem with $m$ right hand sides. Motivated by analogy with estimating the quadratic Taylor polynomial by centered first and second differences of step size $\sigma$, we choose $q = 2(n + (n-1)n)$ and sample $\|r(x) - r(x_0)\|_2$ at $x = x_0 + \sigma_1 u_i + \sigma_2 u_j$ for $1 \le i < j \le n$ and for $\sigma_1$, $\sigma_2$ taking values from $\{0, \sigma, -\sigma\}$.

As in Section 4.2, it is important to correct this model by probing along nearly null vectors of $R$. Let $R = USV^T$ be a singular value decomposition of the $R$ obtained by least squares as described in the preceding paragraph. Probe along directions corresponding to columns of $V$, update $S$,and finally let the new model be $R = SV^T$. In this way, the parametric spline rounding problem has been reduced to the form needed by the algorithm of Section 3.

## 7. Numerical results for parametric curves

Parametric spline fitting with rounded B-spline control points was tested on a United States government map database and on outlines for a Cyrillic font. The map database[18] describes roads, streams, railroads, and other features as sequences of (latitude,longitude) pairs. Many of the resulting polygonal lines are promising candidates for curve approximation because actual roads, streams and railroads are often curved.

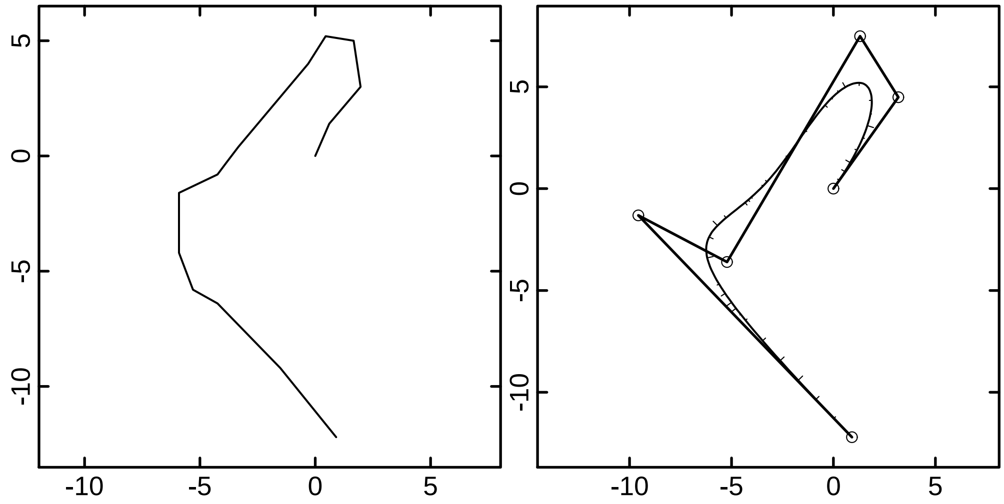


Figure 9. A road from the map database and the corresponding parametric B-spline each plotted in units of .0005° of latitude (about 56 meters). Open circles in the right-hand plot mark B-spline control points and the heavy line is the control polygon. Short segments perpendicular to the spline give the error.

The object is to choose suitably accurate parametric B-spline approximations and represent them as concisely as possible. Then the control points are rounded so that they are integer multiples of some basic unit $u$. This could be done by simple componentwise rounding, but it is better to use the quadratic error model from Section 6 with the improved rounding algorithm in Section 3.

For data compression, the basic unit $u$ should be as large as possible so that the control points can be represented with small integers. An overall error target of 22.2 meters was set based on the apparent intrinsic local error in the data for examples like the one in Figure 9. Next, the map features were broken at sharp corners and the ones that had less than 8 vertices or were straight to within the desired tolerance were omitted from further consideration. Spline approximations to the resulting 175 map features produced RMS errors ranging from 0.51 to 21.3 meters.

A basic unit of $u = 56$ meters sufficed to keep all the RMS errors under 22.2 meters. A simple delta encoding of the resulting control points produced 1498 numbers. The UNIX `pack` command provides a reasonably efficient way to encode these numbers. Its Huffman encoding requires 8552 bits including a dictionary that describes the encoding. Hence the actual numbers require 7808 bits or 5.12 bits per number encoded. This is very close to the entropy bound of 7595 bits given by the formula

$$e = -\sum_i c_i \log_2 \left( \frac{c_i}{c_{tot}} \right),$$

where $c_i$ is the number of times $i$ occurs and $c_{tot} = \sum_i c_i$.

How much improvement do the techniques of Sections 6 and 3 provide over simple componentwise rounding? With simple rounding, the basic unit has to be reduced to 16 meters in order to keep the RMS errors under 22.2 meters. This makes some of the numbers produced by delta encoding the control points too big for `pack` to handle since they do not to fit in one byte. However the entropy bound of 10078 bits or 6.73 bits per number encoded is adequate comparison. Based on the entropy, improved rounding provides a 29% improvement in data compression.

The actual improvement in data compression will be lower because parts of the map database were eliminated from consideration as not suitable for spline approximation. This is to be expected since compressing the map database would be a large project and spline fitting is only one part of it.

The font application involved 59 alphabetic characters from a Cyrillic font. The character shapes were described as polygonal outlines containing a total of 3404 vertices. They were preprocessed by breaking at sharp corners and removing straight segments that were not suitable for spline approximation. The continuous spline approximation yielded errors up to 0.33% of the font's em size; i.e., if the 10 point font were scaled so that 10 printer's points is 100 "font units", the RMS errors would range from 0.032 to 0.16 font units.

The target for RMS errors after the rounded B-spline control points was set to 0.35 font units. With improved rounding from Sections 6 and 3, the control point coordinates could be expressed as multiples of 2.19 font units. The resulting 1510 delta-encoded coordinates had an entropy bound of 6665 bits or 4.41 bits per coordinate.

With simple rounding, keeping the RMS errors under 0.35 font units required the control pont coordinates to be multiples of 0.92 font units. The delta encoded coordinates had an entropy bound of 8511 bits or 5.64 bits per coordinate. This implies improved rounding should give a roughly 28% improvement in data compression. This does not count the relatively straight segments that were filtered out as not suitable for spline approximation. About 50% of the segments in the original polygonal outlines fall into this category.

For comparison, we note that the original data in the map database, rounded to units of 22.2 meters, has a theoretical entropy of 33283 bits (versus 7595 for splines). The font outlines, rounded to .466% of em size, have a theoretical entropy of 27328 bits (versus 6665 for splines). Thus we are achieving about a factor of 4 improvement with this (lossy) compression.

## 8. Future directions

There are many aspects of the problem still to explore.

In the univariate free-knot spline, it would be interesting to try Wall-Danielsson as an alternative to `ssaf`. The optimization should use the log transform recommended by Jupp[7], which we have had good success with in conformal mapping[2]. Other approaches that we would like to try, suitably adapted, are [12] and [11]. A thorough comparison of the various approaches would be desirable, but is tangential to our main topic of how to round once the continuous optimum has been found.

Although we have demonstrated excellent theoretical compression of the map database, practical completion of the project calls for further tuning of tolerances, a more robust spline optimization procedure, maintenance of intersection topology, and other work. Since this is a database that at present does not quite fit on one CD-ROM but could potentially be of value in millions of automobiles, we hope that someone will be motivated to follow up on this.

There are applications besides splines, in areas like image processing, where current algorithms compute approximations in floating point but the final desired output is represented in small integers. If suitable error metrics can be found, improved rounding might be found valuable for these problems as well.

## Acknowledgements

We would like thank David Gay for his `n2f` nonlinear least squares program, Norm Schryer for his `ssaf` program, and Henry Baird and Ruby Jane Elliott for their implementation of Wall-Danielsson. We thank Jeff Lagarias and Andrew Odlyzko for pointing us to [1, 8, 9, 15, 19]. Ken Thompson's extensive processing of the TIGER database[18] made it possible to extract sample curves with moderate effort. Norm Schryer and Margaret Wright and an anonymous referee made helpful comments on the manuscript. PostScript is a registered trademark of Adobe Systems Incorporated. UNIX is a registered trademark of UNIX System Laboratories, Inc. VAX is a trademark of Digital Equipment Corporation.

## References

1. L. Babai, *On Lovász' lattice reduction and the nearest lattice point problem*, Combinatorica, 6 (1986), pp. 1–13.
2. P. E. Bjørstad and E. H. Grosse, *Conformal mapping of circular arc polygons*, SISSC, 8 (1987), pp. 19–32.
3. C. de Boor, *Good approximation by splines with variable knots, II*, in Numerical Solution of Differential Equations, G. A. Watson, ed., Springer-Verlag, New York, 1974, pp. 12–20.
4. C. de Boor and J. R. Rice, *Least squares cubic spline approximation. ii: variable knots*, tech. rep., CSD TR 21, Purdue Univ., Lafayette, IN, 1968.
5. J. E. Dennis, Jr., D. M. Gay, and R. E. Welch, *An adaptive nonlinear least squares algorithm*, Trans. on Mathematical Software, 7 (1981), pp. 348–368,369–383.
6. F. N. Fritsch and G. M. Nielson, *On the problem of determining the distance between parametric curves*, tech. rep., Lawrence Livermore National Laboratory UCRL-JC-105408, 1990.
7. D. L. B. Jupp, *Approximation to data by splines with free knots*, SIAM J. Numerical Analysis, 15 (1978), pp. 328–343.
8. J. C. Lagarias, H. W. Lenstra Jr., and C. P. Schnorr, *Korkin-Zolotarev bases and successive minima of a lattice and its reciprical*, Combinatorica, 10 (1990), pp. 333–348.
9. B. A. LaMacchia, *Basis reduction algorithms and subset sum problems*, Master's thesis, EECS Dept., Massachusetts Institute of Technology, 1991.
10. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen, 261 (1982), pp. 515–534.

11. P. D. Loach and A. J. Wathen, *On the best least squares approximation of continuous functions using linear splines with free knots*, IMA J. Numerical Analysis, 11 (1991), pp. 393–409.
12. B. Meinardus, G. Nürnberger, M. Sommer, and H. Strauss, *Algorithms for piecewise polynomials and splines with free knots*, Math. Comp., 53 (1989), pp. 235–247.
13. C. P. Schnorr, *A heirarchy of polynomial time lattice basis reduction algorithms*, Theoretical Comput. Sci., 53 (1987), pp. 201–224.
14. ———, *A more efficient algorithm for lattice basis reduction*, Journal of Algorithms, 9 (1988), pp. 47–62.
15. C. P. Schnorr and M. Euchner, *Lattice basis reduction: Improved practical algorithms and solving subset sum problems*, in Proceedings of Fundamentals of Computation Theory '91, Springer-Verlag, New York, 1991. Lecture Notes in Computer Science.
16. N. L. Schryer, *SSAF – smooth spline approximations to functions*, tech. rep., AT&T Bell Laboratories CSTR 131, 1986.
17. L. L. Schumaker, *Spline Functions: Basic Theory*, Wiley, 1981.
18. *TIGER/LINE census files, 1990, technical documentation*, tech. rep., Bureau of the Census, U. S. Dept. of Commerce, Washington, D.C., 1991.
19. P. van Emde Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*, Report 81-04, Math. Institute, Univ. of Amsterdam, 1981.
20. K. Wall and P.-E. Danielsson, *A fast sequential method for polygonal approximation of digitized curves*, Computer Vision, Graphics, and Image Processing, 28 (1984), pp. 220–227.

## Supplement

AT&T Bell Laboratories, Murray Hill NJ 07974 USA.
*E-mail address*: ehg@research.att.com

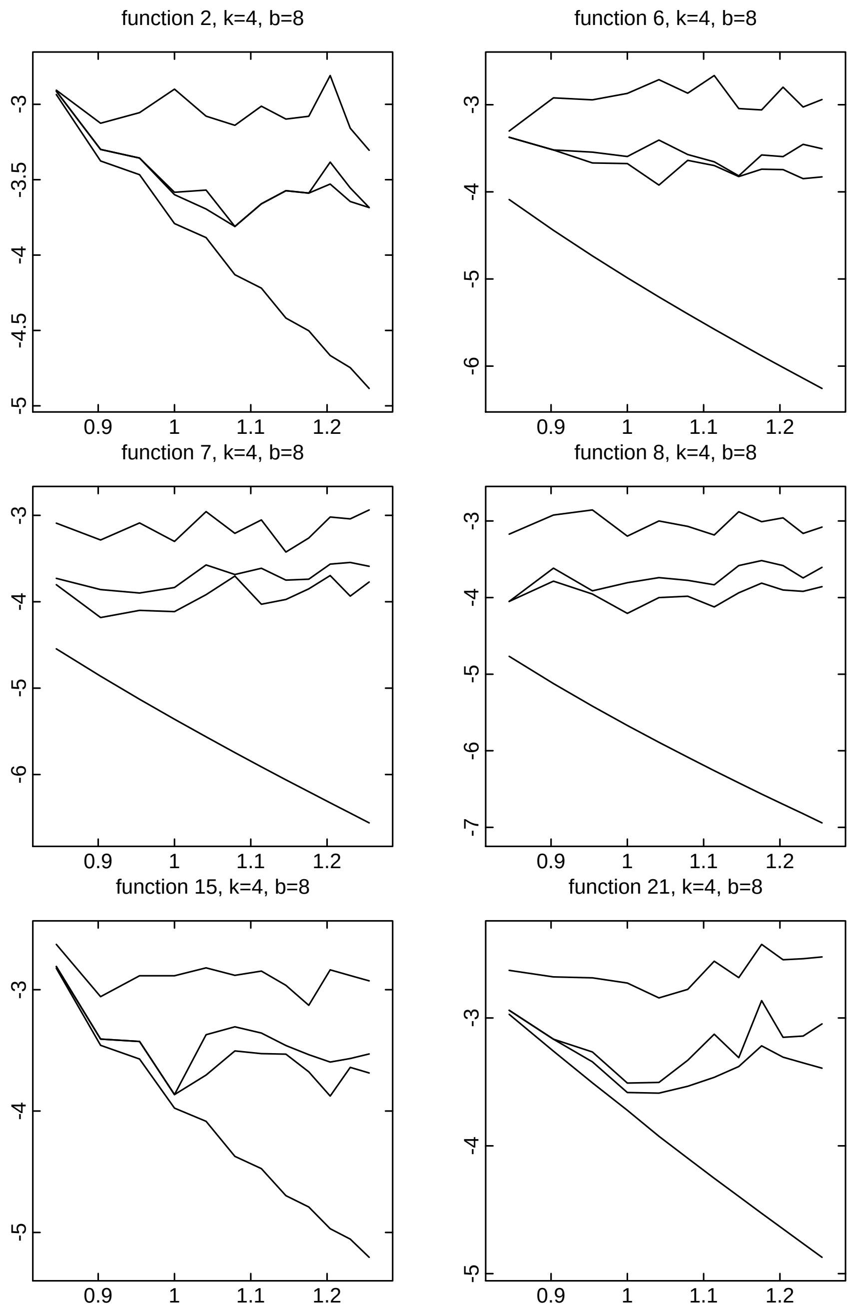


FIGURE 10. The logarithm of RMS error versus $\log_{10}(n)$ for each test function when $b = 8$.

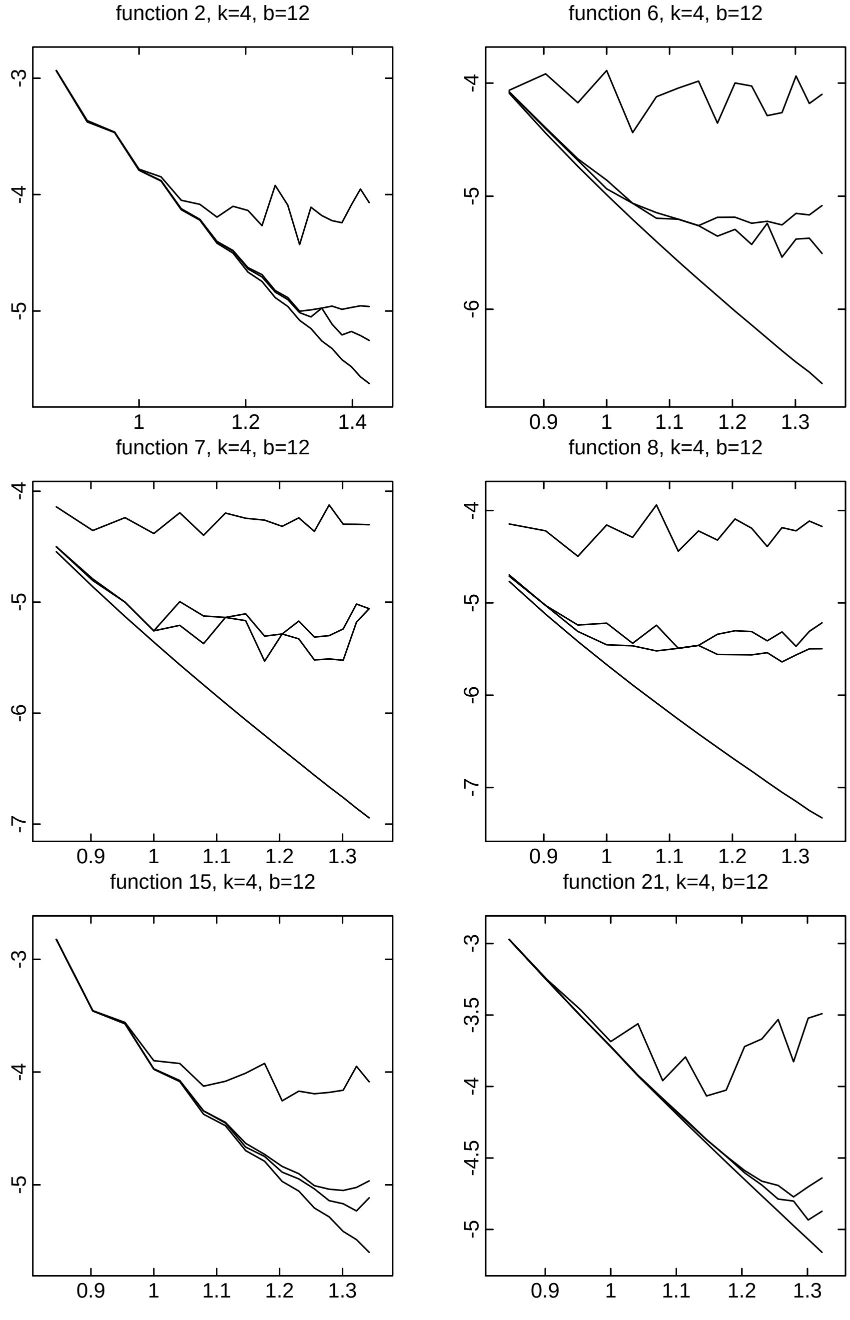


FIGURE 11. The logarithm of RMS error versus $\log_{10}(n)$ for each test function when $b = 12$.