Adaptive Scattered Data Interpolation with Multilevel Nonuniform B-Splines

J. Lin† and Z. Huang‡
National University of Singapore

Abstract
We present an adaptive method for scattered data interpolation. The method is based on multilevel nonuniform B-splines. It makes use of a coarse-to-fine hierarchy of control lattices to generate a sequence of bicubic nonuniform B-spline functions whose sum approaches the desired interpolation function. Experimental results demonstrate that the method performs better than the method using uniform B-splines.

1. Introduction
Scattered data interpolation is the practice of fitting a smooth surface through a scattered distribution of data samples. This is often applied in science and engineering where data are measured or generated at sparse and irregular positions. Interpolation is to estimate an underlying function that can be evaluated at any position. The use of bicubic B-spline surfaces to represent functions is very popular due to the advantages, such as $C^2$ continuity, of such surfaces.

There has been much work in this area. However, scattered data interpolation is still a difficult and computationally expensive problem. Much of the work suffers from limitations in smoothness, time complexity or allowable data distributions. Lee et al. have significantly improved performance by applying multilevel B-spline interpolation.

Based on the idea of multilevel interpolation, we present a different way to adaptively interpolate the scattered data samples. It is an extension of the method by Lee et al., where uniform B-splines are used. They also propose an adaptive representation of the control lattice hierarchy but a linear array is used. We will show the use of nonuniform B-splines is a better representation in our method.

2. Other Relevant Work
There is much work devoted to scattered data interpolation. An excellent review can be found in. The well-known methods include approaches based on Shepard's method, radial basis functions, thin plate splines, finite element method, a hierarchical B-Spline approach, and optimization based techniques.

3. The Algorithm
3.1. Overview
The algorithm is outlined as follows:

1. Initial approximation: use a coarse lattice $\Phi_0$ to calculate a uniform B-spline surface $f_0$ based on point set $P$. We use index $i$ to represent the current layer of subdivision. So $i = 0$ at this step. More details can be found in Lee et al.

2. Derivation of the deviation: increase index $i$ by 1, i.e., $i = i + 1$. Then, derive the deviation function $\Delta_i$ of the last approximation in layer $i - 1$.

3. Refinement of the approximation: check the error and gradient of the surface $f_{i-1}$ on each subdividing region of $\Phi_{i-1}$:
   a. If the test results for all current regions are below the specified threshold, sum $f_0, f_1, \ldots, f_{i-1}$ to get the final approximation function $f$. Stop.
   b. Otherwise, go to next step.

4. Adaptive subdivision: for each region in which the test fails, further subdivide it. A new lattice $\Phi_i$ is derived.

† Email: scip6151@mango.math.nus.edu.sg. Department of Mathematics, Faculty of Science, NUS, Singapore 119260.
‡ Email: huangzy@comp.nus.edu.sg. Department of Computer Science, School of Computing, NUS, Singapore 119260.
§ This work was supported by the NUS under Research Scholarship and grant RP3982704.
5. Derive an approximation to the deviation: derive a nonuniform B-spline function $f_i$ to approximate $\Delta'$, go to step 2 for deriving the deviation.

### 3.2. Adaptive Subdivision

Consider a hierarchy of control lattices, $\Phi_0, \Phi_1, \cdots, \Phi_k$, overlaid on domain $\Omega$. We derive $\Phi_0$ from the initial approximation. As the lattice $\Phi_0$ is very coarse, the initial approximation $f_0$ leaves large errors at the data points in $P$. Assume $f_0$ leaves a deviation $\Delta x_i z_i = z_i - f_0(x_i, y_i)$ for each point $(x_i, y_i, z_i)$ in $P$. The next finer control lattice $\Phi_1$ is then used to obtain function $f_1$ to approximate the difference $P_1 = \{(x_i, y_i, \Delta x_i z_i)\}$. From this step, all successively finer lattices serve to approximate and remove the residual error. In general, for level $k$ in the hierarchy, the function $f_k$ is derived by using control lattice $\Phi_k$ to approximate $P_k = \{(x_i, y_i, \Delta x_i z_i)\}$, where $\Delta x_i z_i = z_i - \sum_{i=0}^{k-1} f_i(x_i, y_i) = \Delta x_i z_0 - f_{k-1}(x_i, y_i)$, and $\Delta x_0 z_0 = z_0$. This process continues incrementally until some conditions are met, e.g., $\Delta x_i z_i < \varepsilon$. Then, we get $f_k$ to approximate $\Delta x_i z_i$ over lattice $\Phi_k$. The final approximation function $f$ is derived by summing the $f_k$:

$$f = \sum_{k=0}^{\infty} f_k.$$  \hspace{1cm} (1)

In this way, the scattered data $P = \{(x_i, y_i, z_i)\}$ is approximated by $f$, the sum of multilevel B-spline functions.

Lee et al. have proposed a sufficient condition for the function $f$ to become an interpolant from an approximant: if no two data points share a control point in their 4 x 4 neighborhoods, i.e., each control point in $\Phi_k$ contains at most one data point in its proximity data set.

The sufficient condition means that a multilevel interpolation requires the control point spacing in the finest lattice $\Phi_k$ becomes sufficiently small. However, it is not necessary to use a fine uniform lattice to overlay the whole domain $\Omega$ in which not all regions have data points. Lee et al. noticed the problem and proposed an adaptive control lattice hierarchy to tackle the problem. Their purpose is only to save memory. Thus a linear array is used to represent a hierarchy of two dimensional lattices.

Noticing the same problem, we believe the use of nonuniform B-spline functions is a better choice:

1. Less control points are needed than the multilevel approximation using uniform B-splines.
2. More accurate and efficient:
   a. the highly varying regions of the data points are approximated by finer control lattices.
   b. the low varying regions by coarser lattices.
3. Easier to maintain the data structure than the use of a linear array for a hierarchy of two dimensional lattices.

We generalize the subdivision condition of Lee et al. who proposed that a region needs further division if it has data points. It is true that a region without data points does not need to be divided. However, a region with data points does not need subdivision either if the approximation is already accurate enough.

We measure the accuracy by checking the error at data points and gradient of the approximant for each region. Thus, a more general condition for adaptive subdivision is given: a subdivision is necessary for a region if its error or the magnitude of gradient is larger than threshold. The error of a region $R$ is derived by:

$$err_k(R) = \max_{(x,y) \in R} |\Delta^k z_i|,$$  \hspace{1cm} (2)

where $k$ is the layer of subdivision.

Using error as the subdivision condition is intuitive. If a surface interpolates all data points in a region, it approximates the underlying function well in this region. Another important factor is gradient. The gradient of a function $g$ is defined as:

$$\nabla g = \frac{\partial g}{\partial x} \hat{x} + \frac{\partial g}{\partial y} \hat{y}.$$  \hspace{1cm} (3)

In a region $R$, a high magnitude of gradient means the function changes rapidly. Thus $R$ needs a finer subdivision. In our estimation, we use the intermediate $B$-spline function $f_k$ to approximate $\Delta^k z_i$. We calculate the maximal magnitude of partial derivatives $\max_{(x,y) \in R} |\frac{\partial g}{\partial x}|$ and $\max_{(x,y) \in R} |\frac{\partial g}{\partial y}|$.

The results are compared with specified threshold to determine whether subdivisions for $R$ are necessary in $x$ and $y$ directions respectively.

Before estimating the maximal magnitude of partial derivatives, we check the length $t$ and width $w$ of $R$. If $t$ ($w$) is smaller than a specified threshold, no further subdivision is necessary in $x$ ($y$) direction in $R$. Therefore, there is no need to estimate partial derivative with respect to $x$ ($y$).

We derive the knot vectors $s$ and $t$ of nonuniform B-spline functions directly from the subdivision of lattice. The parametric $st$-space is same as the object $xy$-space in our implementation. The method to derive the knot vectors is consistent with the approximation principle that a highly variable region needs a finer lattice. The knot vectors are derived by

\begin{align*}
s &= \{x_{-3}, x_{-2}, x_{-1}, x_0, \cdots, \\
  &\quad x_m, x_{m+1}, x_{m+2}, x_{m+3}\}, \\hspace{1cm} (4)

t &= \{y_{-3}, y_{-2}, y_{-1}, y_0, \cdots, \\
  &\quad y_n, y_{n+1}, y_{n+2}, y_{n+3}\}.
\end{align*}

If $\{x_i\} = 0, \cdots, m$ and $\{y_j\} = 0, \cdots, n$ are the subdivisions of the current layer in $x$ and $y$ directions respectively. The additional knots $x_{-3}, x_{-2}, x_{-1}$, and $x_0$ are chosen so that $x_0 = x_{-1} = x_{-2} = x_{-3} = (x_m - x_0)/m$. Similarly, $y_{-3}, y_{-2}, y_{-1}, y_{n+1}$, and $y_{n+3}$ are added after $y_m$, $y_{-3}, y_{-2}, y_{-1}, y_{n+1}$, and $y_{n+3}$ are derived in a similar way.

4. Results

In the first test we used a hat function shown in Figure 1 (left). We took 200 randomly distributed data points in its domain (Figure 1 (right)) and reconstructed the function. We compared the results between the AMBA (Adaptive Multilevel B-spline Approximation) method and uniform B-spline based method. The reconstructed results are shown in Figure 2 with the final subdivisions shown in Figure 3. We observe the AMBA method performs a little better than uniform method in accuracy but with a much less memory cost (Table 1). More subdivisions are dedicated to the highly variable regions. Notice that some regions with data points are not further subdivided such as the top of the hat.

In the second test we used another function shown in Figure 4 (left). We took 120 data points, 40 in each of the two nonzero regions and 40 randomly distributed in the entire domain. The reconstructed result is shown in Figure 4 (right).

Table 1: Comparison results of Test 1.

<table>
<thead>
<tr>
<th>layer</th>
<th>nonuniform memory</th>
<th>NRMS (%)</th>
<th>uniform memory</th>
<th>NRMS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7 x 7</td>
<td>20.564</td>
<td>7 x 7</td>
<td>20.564</td>
</tr>
<tr>
<td>2</td>
<td>11 x 11</td>
<td>10.250</td>
<td>11 x 11</td>
<td>10.250</td>
</tr>
<tr>
<td>3</td>
<td>15 x 15</td>
<td>5.034</td>
<td>19 x 19</td>
<td>5.334</td>
</tr>
<tr>
<td>4</td>
<td>19 x 19</td>
<td>4.058</td>
<td>35 x 35</td>
<td>4.928</td>
</tr>
</tbody>
</table>

Note: After four subdivisions, the memory cost of nonuniform AMBA method is less than half of that of uniform method. NRMS is Normalized Root Mean Square error.

Figure 1: Original hat function (left). Random points used in Test 1 (right).

References