J. Korea Soc. Math. Educ. Ser. B: Pure Appl. Math.
Volume 10, Number 4 (November 2003), Pages 255–263
ISSN 1226-0657

A COMPARISON OF COMPUTING TIMES ON THE BICUBIC B-SPLINE

HOI SUB KIM

ABSTRACT. We compare the computing times on the bicubic B-spline dueing to the algorithms.

1. INTRODUCTION

The computing time is very important because it costs. The surface fitting requires the computation of components of matrix generated by least squares method whose components are composed of functions. In engineering, the fitting is very often repeated to test several problems so that the computing times can not be neglected. So we suggest the difference between two algorithms.

2. B-SPLINE BASIS

The application of B-spline basis appears in several fields (*cf.* Ashahi, Ichige & Ishii [1], de Boor [2], Höllig [3], Lee & Park [5], Pourazady & Xu [6], Stam [7] and Yong, Hu, Sun & Tan [8]). The following recurrence is preferred for its numerical stability and efficiency in computation (*cf.* Kim [4]).

$$M(t \mid t_{i-k}, \ldots, t_i) = \frac{t_i - t}{t_i - t_{i-k}} * M(t \mid t_{i-k+1}, \ldots, t_i) + \frac{t - t_{i-k}}{t_i - t_{i-k}} * M(t \mid t_{i-k}, \ldots, t_{i-1}) \tag{2.1}$$

Received by the editors March 28, 2003 and, in revised form, July 21, 2003.
2000 *Mathematics Subject Classification.* 65D07.
Key words and phrases. bicubic B-spline, computing times.

where the recurrence relation is started with

$$M(t \mid t_{i-1}, t_i) = \begin{cases} \dfrac{1}{t_i - t_{i-1}}, & t_{i-1} \le t < t_i \\ 0, & \text{otherwise} \end{cases}$$

The derivative of the equation (2.1) is given by

$$\frac{d}{dt} M(t \mid t_{i-k}, \ldots, t_i) = \frac{k-1}{t_i - t_{i-k}} \big(M(t \mid t_{i-k}, \ldots, t_{i-1}) - M(t \mid t_{i-k+1}, \ldots, t_i) \big). \tag{2.2}$$

But in programming, one may prefer to use the following piecewise cubic polynomial (2.6) and its derivative, which is more fast than the recursive relation (2.1) and its derivative (2.2) in the sense of implementation.

The piecewise constant function is given by

$$M(t \mid t_{i-1}, t_i) = \begin{cases} \dfrac{1}{t_i - t_{i-1}}, & t_{i-1} \le t < t_i \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

The piecewise linear function is given by

$$M(t \mid t_{i-2}, t_{i-1}, t_i) = \begin{cases} \dfrac{t - t_{i-2}}{t_i - t_{i-2}} * \dfrac{1}{t_{i-1} - t_{i-2}}, & t_{i-2} \le t < t_{i-1} \\ \dfrac{t_i - t}{t_i - t_{i-2}} * \dfrac{1}{t_i - t_{i-1}}, & t_{i-1} \le t < t_i \end{cases} \tag{2.4}$$

The piecewise quadratic function is given by

$$M(t \mid t_{i-3}, t_{i-2}, t_{i-1}, t_i) = \begin{cases} \dfrac{(t - t_{i-3})^2}{(t_i - t_{i-3})(t_{i-1} - t_{i-3})(t_{i-2} - t_{i-3})}, & t_{i-3} \le t < t_{i-2} \\ \dfrac{(t_i - t)(t - t_{i-2})}{(t_i - t_{i-3})(t_i - t_{i-2})(t_{i-1} - t_{i-2})} \\ \qquad + \dfrac{(t - t_{i-3})(t_{i-1} - t)}{(t_i - t_{i-3})(t_{i-1} - t_{i-3})(t_{i-1} - t_{i-2})}, & t_{i-2} \le t < t_{i-1} \\ \dfrac{(t_i - t)^2}{(t_i - t_{i-3})(t_i - t_{i-2})(t_i - t_{i-1})}, & t_{i-1} \le t < t_i \end{cases} \tag{2.5}$$

The piecewise cubic function is given by

$$
\begin{aligned}
(2.6)\quad & M(t \mid t_{t-4}, t_{i-3}, t_{i-2}, t_{i-1}, t_i) \\
&= \begin{cases}
\dfrac{(t-t_{i-4})^3}{(t_i-t_{i-4})(t_{i-1}-t_{i-4})(t_{i-2}-t_{i-4})(t_{i-3}-t_{i-4})}, & t_{i-4} \le t < t_{i-3} \\
\dfrac{(t_i-t)(t-t_{i-3})^2}{(t_i-t_{i-4})(t_i-t_{i-3})(t_{i-1}-t_{i-3})(t_{i-2}-t_{i-3})} & \\
\quad + \dfrac{(t-t_{i-4})(t_{i-1}-t)(t-t_{i-3})}{(t_i-t_{i-4})(t_{i-1}-t_{i-4})(t_{i-1}-t_{i-3})(t_{i-2}-t_{i-3})} & \\
\quad + \dfrac{(t-t_{i-4})^2(t_{i-2}-t)}{(t_i-t_{i-4})(t_{i-1}-t_{i-4})(t_{i-2}-t_{i-4})(t_{i-2}-t_{i-3})}, & t_{i-3} \le t < t_{i-2} \\
\dfrac{(t_i-t)^2(t-t_{i-2})}{(t_i-t_{i-4})(t_i-t_{i-3})(t_i-t_{i-2})(t_{i-1}-t_{i-2})} & \\
\quad + \dfrac{(t_i-t)(t-t_{i-3})(t_{i-1}-t)}{(t_i-t_{i-4})(t_i-t_{i-3})(t_{i-1}-t_{i-3})(t_{i-1}-t_{i-2})} & \\
\quad + \dfrac{(t-t_{i-4})(t_{i-1}-t)^2}{(t_i-t_{i-4})(t_{i-1}-t_{i-4})(t_{i-1}-t_{i-3})(t_{i-1}-t_{i-2})}, & t_{i-2} \le t < t_{i-1} \\
\dfrac{(t_i-t)^3}{(t_i-t_{i-4})(t_i-t_{i-3})(t_i-t_{i-2})(t_i-t_{i-1})}, & t_{i-1} \le t < t_i
\end{cases}
\end{aligned}
$$

3. The least squares method

Let

$$B_i(t) = M(t \mid t_{i-4}, \ldots, t_i).$$

Then we find the bicubic spline surface

$$S(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} B_i(x) B_j(y)$$

of degree 3 which best fits the derivative data (x_k, y_k, f_x^k, f_y^k), $k = 1, \ldots, l$, where l is the number of data points and $f_x^k = \frac{\partial f}{\partial x}(x_k, y_k)$, $f_y^k = \frac{\partial f}{\partial y}(x_k, y_k)$ are the derivative data obtained by Snell's law in the correction lens of Color Display Tube, for example.

We must assign one height value so that the surface is uniquely determined. In our case, we give zero at the center of the correction lens.

Then we minimize the residual sum of the squares of the errors.

(3.1)

$$E = E(a_{11}, a_{12}, \ldots, a_{mn})$$
$$= \sum_{k=1}^{l} \Big\{ (\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} B_i'(x_k) B_j(y_k) - f_x^k)^2 + (\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} B_i(x_k) B_j'(y_k) - f_y^k)^2 \Big\}$$

The necessary and sufficient condition for minimizing the equation (3.1) is as follows.

$$\frac{\partial E}{\partial a_{pq}} = 2\sum_{k=1}^{l} \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} B_i'(x_k) B_j(y_k) - f_x^k \Big) B_p'(x_k) B_q(y_k)$$
$$+ 2\sum_{k=1}^{l} \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} B_i(x_k) B_j'(y_k) - f_y^k \Big) B_p(x_k) B_q'(y_k) \tag{3.2}$$
$$= 0$$

where $p = 1, \ldots, m,\ q = 1, \ldots, n$.

Thus, the coefficients satisfy the normal equations

$$\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} \sum_{k=1}^{l} \Big\{ B_i'(x_k) B_j(y_k) B_p'(x_k) B_q(y_k) + B_i(x_k) B_j'(y_k) B_p(x_k) B_q'(y_k) \Big\}$$
$$= \sum_{k=1}^{l} \Big\{ f_x^k B_p'(x_k) B_q(y_k) + f_y^k B_p(x_k) B_q'(y_k) \Big\}, \tag{3.3}$$

where $p = 1, \ldots, m,\ q = 1, \ldots, n$.

This equation can be written in matrix form $AX = B$ where A is the $mn \times mn$ matrix, X is the mn coefficients a_{ij} and B is the mn vector. In this case, we compute the components of the matrix A and B.

In Appendix, Visual basic Pseudocode is given. `Public Sub fitting spline derivative()` is the computation of the components of the matrix A and B of the matrix equation $AX = B$.

`Public Function basis()` is the basis computation of the piecewise cubic function (2.6). `Public Function dbasis()` is the computation of the derivative of the piecewise cubic function (2.6). `Public Function basisr()` is the computation of the old recursive function (2.1).

In Color Picture Tube (for TV usage) and Color Display Tube (for Monitor usage), the ray-tracing occurs in Exposure Process. The mechanism consists of Lamphouse, Correction Lens, Filter, Shadow Mask, Panel as in Figure 1. The

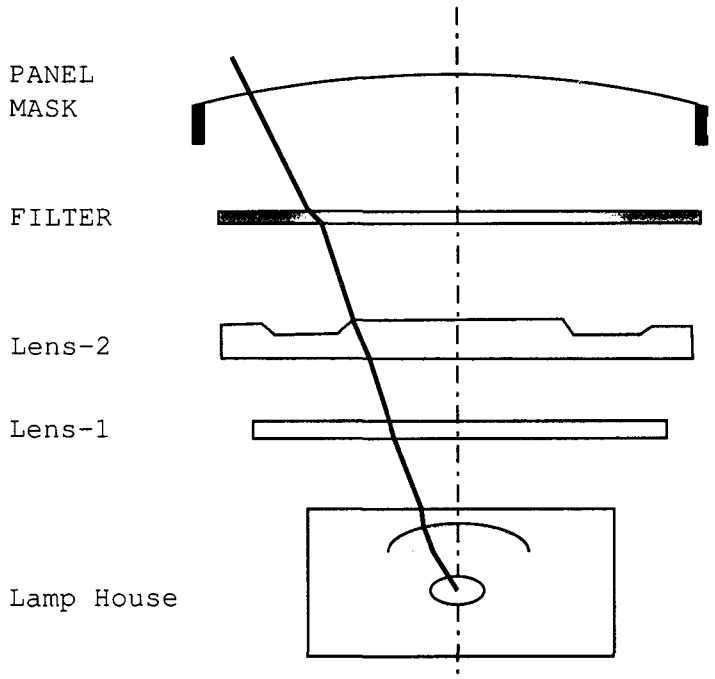

Figure 1. Exposure Process

Lamphouse is the source of light. The surface type for correction lens is polynomial or B-spline surface, that of Filter is flat plane, that of Shadow mask is a polynomial and that of the panel is a polynomial.

In ray-tracing of landing error corrections, the surface of the correction lens is reconstructed by least squares method of derivative data generated by Snell's law. In that case, the calculation of the B-spline basis function and its derivative function is repeated several times for which measurement points of landing errors are 169 points, for example. When we call `fitting spline derivative()`, the piecewise cubic algorithm is more fast than the old recursive algorithm approximately 50 as in Table 1.

Table 1. Computing times according to algorithms

Number of points	The old recursive algorithm (2.1)	The piecewise cubic function (2.6)
100	5 seconds	3 seconds
1000	50 seconds	30 seconds
3000	150 seconds	85 seconds

REFERENCES

1. T. Ashahi, K. Ichige & R. Ishii: A Computationally Efficient Algorithm for Exponential B-splines Based on Difference/IIR Filter Approach. *IEICE transactions on fundamentals of electronics communcations and computer science* **85** (2002), no. 6, 1265–1273.
2. C. de Boor: Package for calculating with B-splines. *SIAM J. Numer. Anal.* **14** (1977), no. 3, 441–472. MR **55**#1711
3. K. Höllig: Stability of the B-spline basis via knot insertion. *Comput. Aided Geom. Design* **17** (2000), no. 5, 447–450. MR **2000m:**41009
4. H. S. Kim: On the construction of a surface from discrete derivative data and its extended surface using the least squares method. *Korean J. Comput. Appl. Math.* **4** (1997), no. 2, 327–336. MR **98e:**65008
5. B.-G. Lee & Y. Park: Degree elevation of B-spline curve and its matrix representation. *Korea Soc. Indust. Appl. Math.* **4** (2000), no. 2, 1–10.
6. M. Pourazady & X. Xu: Direct manipulations of B-spline and NURBS curves. *Adv. Eng. Softw.* **31** (2000), no. 2, 107–118.
7. J. Stam: On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. *Comput. Aided Geom. Des.* **18** (2001), no. 5, 383–396.
8. J.-H. Yong, S.-H. Hu, J.-G. Sun & X.-Y. Tan: Degree reduction of B-spline curves. *Comput. Aided Geom. Des.* **18** (2001), no. 2, 117–127.

APPENDIX. VISUAL BASIC PSEUDOCODE

```
Public Sub fitting_spline_derivative()
For k = 1 To ld
For ip = 0 To im
    sxp = basis(ip, dx(k), knotx)
    dsxp = dbasis(ip, dx(k), knotx)
   For jq = 0 To jn
       syp = basis(jq, dy(k), knoty)
       dsyp = dbasis(jq, dy(k), knoty)
   For i = 0 To im
       sx = basis(i, dx(k), knotx)
       dsx = dbasis(i, dx(k), knotx)
   For j = 0 To jn
       sy = basis(j, dy(k), knoty)
```

```
      dsy = dbasis(j, dy(k), knoty)
         a(i * (jn + 1) + j + 1, ip * (jn + 1) + jq + 1) =
         a(i * (jn + 1) + j + 1, ip * (jn + 1) + jq + 1)_
         + weightx(k) * dsx * sy * dsxp * syp_
         + weighty(k) * sx * dsy * sxp * dsyp_
         + basis(i, 0, knotx) * basis(j, 0, knoty)_
         * basis(ip, 0, knotx) * basis(jq, 0, knoty)
      Next j
   Next i
         b(ip * (jn + 1) + jq + 1) = b(ip * (jn + 1) + jq + 1)_
         + weightx(k) * dsxp * syp * dzdx(k)_
         + weighty(k) * sxp * dsyp * dzdy(k)_
         + basis(ip, 0, knotx) * basis(jq, 0, knoty) * zcentervalue
   Next jq
Next ip
Next k
End Sub

Public Function basis(ByVal i As Integer,_
                      ByVal x As Double, knot) As Double
  If (x >= knot(i - 4) And x < knot(i - 3)) Then
     basis = (x - knot(i - 4)) * (x - knot(i - 4))_
             * (x - knot(i - 4))
  ElseIf (x >= knot(i - 3) And x < knot(i - 2)) Then
     basis = (knot(i) - x) * (x - knot(i - 3))_
            * (x - knot(i - 3)) _
            + (x - knot(i - 4)) * (knot(i - 1) - x)_
            * (x - knot(i - 3)) + (x - knot(i - 4))_
            * (x - knot(i - 4)) * (knot(i - 2) - x)
  ElseIf (x >= knot(i - 2) And x < knot(i - 1)) Then
     basis = (knot(i) - x) * (knot(i) - x) * (x - knot(i - 2))_
            + (knot(i) - x) * (x - knot(i - 3)) * (knot(i - 1) - x)_
            + (x - knot(i - 4)) * (knot(i - 1) - x)_
            * (knot(i - 1) - x)
  ElseIf (x >= knot(i - 1) And x < knot(i)) Then
```

```
    basis = (knot(i) - x) * (knot(i) - x) * (knot(i) - x)
  Else
    basis = 0
  End If
End Function

Public Function dbasis(ByVal i As Integer,_
                       ByVal x As Double, knot) As Double
  If (x >= knot(i - 4) And x < knot(i - 3)) Then
    dbasis = 3 * (x - knot(i - 4)) * (x - knot(i - 4))
  ElseIf (x >= knot(i - 3) And x < knot(i - 2)) Then
    dbasis = (-(x - knot(i - 3)) * (x - knot(i - 3))_
               + 2 * (knot(i) - x) * (x - knot(i - 3)))_
           + ((knot(i - 1) - x) * (x - knot(i - 3))_
           - (x - knot(i - 4)) * (x - knot(i - 3))_
           + (x - knot(i - 4)) * (knot(i - 1) - x))_
           + (2 * (x - knot(i - 4)) * (knot(i - 2) - x)_
           - (x - knot(i - 4)) * (x - knot(i - 4)))
  ElseIf (x >= knot(i - 2) And x < knot(i - 1)) Then
    dbasis = (-2 * (knot(i) - x) * (x - knot(i - 2))_
              + (knot(i) - x) * (knot(i) - x))_
           + (-(x - knot(i - 3)) * (knot(i - 1) - x)_
           + (knot(i) - x) * (knot(i - 1) - x)_
           - (knot(i) - x) * (x - knot(i - 3)))_
           + ((knot(i - 1) - x) * (knot(i - 1) - x)_
           - 2 * (x - knot(i - 4)) * (knot(i - 1) - x))
  ElseIf (x >= knot(i - 1) And x < knot(i)) Then
    dbasis = -3 * (knot(i) - x) * (knot(i) - x)
  Else
    dbasis = 0
  End If
End Function

Public Function basisr(ByVal i As Integer,_
                       ByVal x As Double, ByVal k As Integer)_
```

```
                        As Double
Dim d As Double
  d = knotx(i) - knotx(i - k)
  If (k = 1) Then
    If (x < knotx(i) And x >= knotx(i - 1)) Then
      basisx = 1# / (knotx(i) - knotx(i - 1))
    Else
      basisx = 0#
    End If
  Else
      basisx =((knotx(i)-x)/d)*basisx(i,x,k-1)+((x-knotx(i-k))/d)_
               *basisx(i-1,x,k-1)
  End If
End Function
```

DEPARTMENT OF MATHEMATICS, KYUNGWON UNIVERSITY, SAN 65, BOKJEONG-DONG, SUJEONG-GU, SEONGNAM, GYEONGGI 461-701, KOREA
Email address: hskimm@kyungwon.ac.kr