Degree Elevation of B–spline Curves and Its Matrix Form

Byung-Gook Lee

Department of Applied Mathematics, Dongseo University, Pusan, 617–716, Republic of Korea

Yunbeom Park

Department of Mathematics Education, Seowon University, Chongju, 361–742, Republic of Korea

Abstract

An algorithmic approach to degree elevation of B–spline curves is presented. The new algorithms are based on the blossoming process and its matrix representation. The elevation method is introduced that consists of the following steps: (a) decompose the B–spline curve into piecewise Bézier curves, (b) degree elevate each Bézier piece, and (c) compose the piecewise Bézier curves into B–spline curve.

Key words: Degree Elevation, B–spline, Blossoming

1 Introduction

Degree elevation of B–spline curves are well understood and several algorithms are published([10], [11], [12], [13], [14], [15]). From a software engineering point of view, it is desirable to implement a simple and easy-to-understand algorithm. This approach was taken by Piegl and Tiller([11], [12], [13]), who implemented the simplest algorithm; they decomposed the B–spline curve into piecewise Bézier curves, elevated the degree of each Bézier piece, and then composed the piecewise Bézier curves into B–spline curves. We describe here the modified form of Piegl and Tiller's degree elevation algorithm. The new algorithms are based on the blossoming analysis ([2], [8], [16], [17]) and matrix representation of the process.

2 Degree Elevation

Since a B–spline curve is a piecewise polynomial curve, it must be possible to elevate its degree from p to $p+r$. That is, there must exist control points $\tilde{P}$ and a knot vector $\tilde{U}$ such that

$$C_P^n(u) = C_{\tilde{P}}^{\tilde{n}}(u) = \sum_{i=0}^{\tilde{n}} \tilde{P}_i N_{i,p+r}(u).$$

The curve $C_P^n(u)$ and $C_{\tilde{P}}^{\tilde{n}}(u)$ are the same geometrically and parametrically. The computing of $\tilde{n}$, $\tilde{P}$ and $\tilde{U}$ is referred to as degree elevation. The knot vector $\tilde{U}$ and number of points $\tilde{n}$ can easily be computed as follows.

Assume that U has the form

$$U = \{\underbrace{a, a, \ldots, a}_{p+1}, \underbrace{u_1, \ldots, u_1}_{m_1}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s}, \underbrace{b, b, \ldots, b}_{p+1}\}$$

where the end knots a and b are repeated with multiplicity $p+1$, the interior knots u_i are repeated with multiplicity m_i and s is the number of distinct interior knots. Since the curve $C_P^n(u)$ is $C^{p-m_i}-$continuous at the knot of multiplicity m_i, $C_{\tilde{P}}^{\tilde{n}}(u)$ must have the same continuity. Consequently, the new vector must take the form

$$\tilde{U} = \{\underbrace{a, a, \ldots, a}_{p+r+1}, \underbrace{u_1, \ldots, u_1}_{m_1+r}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s+r}, \underbrace{b, b \ldots, b}_{p+r+1}\}$$

which gives $\tilde{n} = n + (s+2)r$.

The computation of $\tilde{P}$ can be done in a number of different ways, including solving a system of linear equations([3]), combining some techniques of box splines with knot insertion([14]), using blossoming analysis([10]), or approaching aspect of software engineering ([11]). We provide a procedural method that combine the blossoming analysis and the software engineering aspect with the generalized least square method. The procedure can be summarized as follows:

(1) Decompose the B–spline curve into piecewise Bézier curves.
(2) Degree elevate each Bézier piece.
(3) Make the B–spline curve from the piecewise Bézier segment.

Curve Decomposition: Curve decomposition is normally done via knot insertion. This is very convenient in curve design when it is necessary to have local control. In the case of a nonuniform B–spline curve a new knot can be inserted to increase the number of control points and the number of curve segments. Consider the cubic B–spline curve in Fig. 1 with blossoming notation. Here, the Bézier segments are extracted by inserting knot 1 two times.

From these we can get first Bézier piece $B_G(0,0,0)$, $B_G(0,0,1)$, $B_G(0,1,1)$, and $B_G(1,1,1)$, and second Bézier piece $B_G(1,1,1)$, $B_G(1,1,3)$, $B_G(1,3,3)$, and $B_G(3,3,3)$.

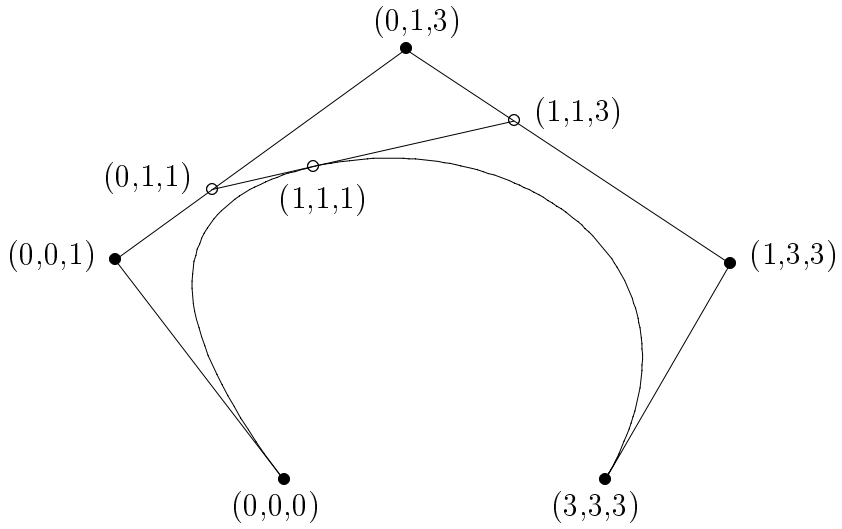

Fig. 1. Curve decomposition of a cubic B–spline curve. The control polygon(circle) after inserting knot $u = 1$ two times into the knot vector $(0,0,0,0,1,3,3,3,3)$.

The multiaffine and symmetry property can be used to compute new blossom values from old ones. For example, consider finding the value $B_G(0,1,1)$, $B_G(1,1,1)$, and $B_G(1,1,3)$. Note,

$$
\begin{aligned}
B_G(0,1,1) &= 2/3 B_G(0,1,0) + 1/3 B_G(0,1,3) = 2/3 B_G(0,0,1) + 1/3 B_G(0,1,3) \\
B_G(1,1,1) &= 2/3 B_G(1,0,1) + 1/3 B_G(1,3,1) = 2/3 B_G(0,1,1) + 1/3 B_G(1,1,3) \\
&= 4/9 B_G(0,0,1) + 4/9 B_G(0,1,3) + 1/9 B_G(1,3,3) \\
B_G(1,1,3) &= 2/3 B_G(0,1,3) + 1/3 B_G(3,1,3) = 2/3 B_G(0,1,3) + 1/3 B_G(1,3,3).
\end{aligned}
$$

The Decomposition algorithm in Fig. 1 is expressed in a matrix form as

$$Q = M_d P$$

where

$$
Q = \begin{pmatrix} B_G(0,0,0) \\ B_G(0,0,1) \\ B_G(0,1,1) \\ B_G(1,1,1) \\ B_G(1,1,3) \\ B_G(1,3,3) \\ B_G(3,3,3) \end{pmatrix}, M_d = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 4/9 & 4/9 & 1/9 & 0 \\ 0 & 0 & 2/3 & 1/3 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } P = \begin{pmatrix} B_G(0,0,0) \\ B_G(0,0,1) \\ B_G(0,1,3) \\ B_G(1,3,3) \\ B_G(3,3,3) \end{pmatrix}.
$$

Next we give detailed pseudocode to compute the decomposition matrix M_d. It uses a local array V of size s to store the ith distinct interior knot values and another one M of size s to store the multiplicity.

Make_DecomposeBsplineMatrix(U, m, n, p)
// Input: Knots vector $U = \{a, a, \ldots, a, u_1, \ldots, u_1, \ldots, u_s, \ldots, u_s, b, b, \ldots, b\}$,
// number of knots $m + 1$,
// number of control points $n + 1$ and degree p
// Output: $p(s + 1) + 1 * n + 1$ matrix M_d

(* In case of Bezier curve *)
if $(n = p)$ **exit**;

(* initialize some variables *)
$s = 1$; $t = 1$; $l = p$;

(* initialize M_d matrix *)
for $(i = 0$ to n by $1)$
  for $(j = 0$ to n by $1)$
    if $(i = j)$ $M_d[i][j] = 1$; **else** $M_d[i][j] = 0$;

(* compute knot multiplicity *)
$V[s] = U[p + 1]$; $M[s] = 1$;
for $(i = p + 2$ to n by $1)$
  if $(V[s] = U[i])$ $M[s] = M[s] + 1$;
  else $s = s + 1$; $V[s] = U[i]$; $M[s] = 1$;
  endif
endfor

(* make M_d matrix *)
for $(i = 1$ to s by $1)$
  $l = l + M[i]$;
  for $(j = M[i]$ to $p - 1$ by $1)$
    KnotsInsertion $(U, l, V[i], n, n + t, p)$;
    for $(k = m + t$ to $l + 1$ by $-1)$ $U[k] = U[k - 1]$;
    $U[l] = V[i]$; $l = l + 1$; $t = t + 1$;
  endfor
endfor

KnotsInsertion(U, l, v, n, k, p)
// Input: Knots vector U, new knot $v \in [u_l, u_{l+1})$, and degree p
// Output: $n + 1 * k + 1$ matrix M_d

for $(i = l - p + 1$ to $l - 1$ by $1)$

$\alpha = (v - U[i])/(U[i+p] - U[i])$;
for $(j = 0$ to n by $1)$
$T[i][j] = (1-\alpha)M_d[i-1][j] + \alpha M_d[i][j]$;
endfor
endfor
for $(i = k$ to l by $-1)$
for $(j = 0$ to n by $1)$ $M_d[i][j] = M_d[i-1][j]$;
for $(i = l - p + 1$ to $l - 1$ by $1)$
for $(j = 0$ to n by $1)$ $M_d[i][j] = T[i][j]$;

Degree Elevation of Bézier Curves: The degree elevation of Bézier curves is well understood and well documented([7]). As a first step, consider raising the degree of the Bézier curve by one. We can show that new control points are obtained from the old polygon by piecewise linear interpolation at the parameter values $j/(n+1)$. We may repeat this process and obtain a sequence of control points. After r degree elevations, we have a linear system $R = T_{n,r}Q$ where the $(n+r+1) \times (n+1)$ matrix $T_{n,r} = \{t_{i,j}\}$ has elements([9])

$$t_{i+j,i} = \frac{\binom{n}{i}\binom{r}{j}}{\binom{n+r}{i+j}}, \quad \begin{cases} i = 0, 1, \ldots, n \\ j = 0, 1, \ldots, r. \end{cases} \tag{1}$$

Bézier curves are known to be a special polynomial type of B–spline curve with the knot vector given by n knots at a and n knots at b. By the dual functional property the control vertices Q_i for the Bézier representation of the curve are given in terms of blossom by $Q_i = B_G(a, a, \ldots, a, b, b, \ldots, b)$ where a appears as an argument $(n-i)$ times and b appears i times. For the sake of simplicity, consider the cubic case. We begin with the control vertices from the blossom representations $B_G(a,a,a)$, $B_G(a,a,b)$, $B_G(a,b,b)$, and $B_G(b,b,b)$. From these we wish to raise the degree of the Bézier curve by two. We can represent the new Bézier curve as having a knot vector with $n+2$ knots at a and another $(n+2)$ at b as shown in Fig. 2.

$R_0 = B_G(a,a,a,a,a) = B_G(a,a,a)$
$R_1 = B_G(a,a,a,a,b) = 2/5B_G(a,a,a) + 3/5B_G(a,a,b)$
$R_2 = B_G(a,a,a,b,b) = 1/10B_G(a,a,a) + 3/5B_G(a,a,b) + 3/10B_G(a,b,b)$
$R_3 = B_G(a,a,b,b,b) = 3/10B_G(a,a,b) + 3/5B_G(a,b,b) + 1/10B_G(b,b,b)$
$R_4 = B_G(a,b,b,b,b) = 3/5B_G(a,b,b) + 2/5B_G(b,b,b)$
$R_5 = B_G(b,b,b,b,b) = B_G(b,b,b)$

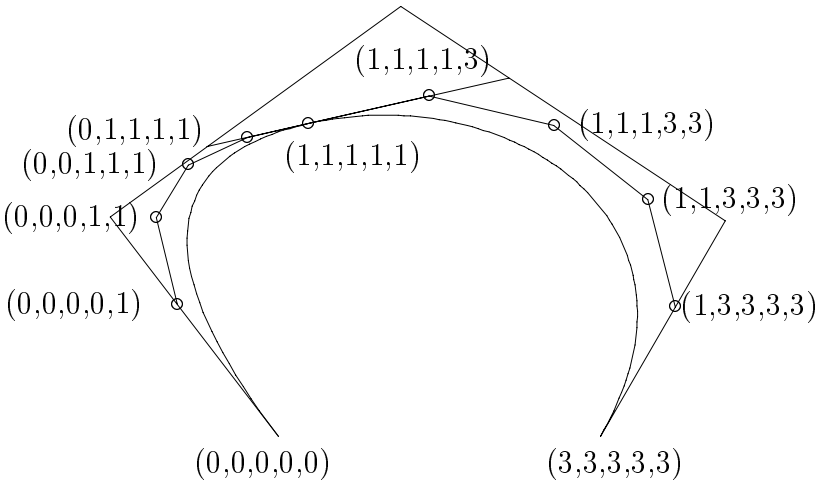

Fig. 2. Degree elevation each Bézier piece(circle).

The degree elevation algorithm in Fig. 2 is expressed in a matrix form as

$$R = M_e Q$$

where

$$R = \begin{pmatrix} B_G(0,0,0,0,0) \\ B_G(0,0,0,0,1) \\ B_G(0,0,0,1,1) \\ B_G(0,0,1,1,1) \\ B_G(0,1,1,1,1) \\ B_G(1,1,1,1,1) \\ B_G(1,1,1,1,3) \\ B_G(1,1,1,3,3) \\ B_G(1,1,3,3,3) \\ B_G(1,3,3,3,3) \\ B_G(3,3,3,3,3) \end{pmatrix}, M_e = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/5 & 3/5 & 0 & 0 & 0 & 0 & 0 \\ 1/10 & 3/5 & 3/10 & 0 & 0 & 0 & 0 \\ 0 & 3/10 & 3/5 & 1/10 & 0 & 0 & 0 \\ 0 & 0 & 3/5 & 2/5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/5 & 3/5 & 0 & 0 \\ 0 & 0 & 0 & 1/10 & 3/5 & 3/10 & 0 \\ 0 & 0 & 0 & 0 & 3/10 & 3/5 & 1/10 \\ 0 & 0 & 0 & 0 & 0 & 3/5 & 2/5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } Q = \begin{pmatrix} B_G(0,0,0) \\ B_G(0,0,1) \\ B_G(0,1,1) \\ B_G(1,1,1) \\ B_G(1,1,3) \\ B_G(1,3,3) \\ B_G(3,3,3) \end{pmatrix}.$$

The following algorithm compute degree elevation M_e matrix.

Make_ElevateBezierMatrix(p, r, s)
// Input: Degree p, elevation degree r and number of distinct interior knots s
// Output: $(p+r)(s+1)+1*p(s+1)+1$ matrix M_e

$div = 1$;
for ($i = 1$ to r by 1) $div = div \times (p+i)$;

$M[0] = 1$;
for ($i = 0$ to p by 1)
  for ($j = 1$ to r by 1) $M[j] = M[j] + M[j-1]$;
  for ($j = 0$ to r by 1)

```
      left = 1;
      for (k = 1 to j by 1) left = left × (r − k + 1);
      right = 1;
      for (k = 1 to r − j by 1) right = right × (p − i + k);
      T[i + j][i] = left × M[j] × right/div;
    endfor
endfor

M_e[0][0] = T[0][0];
for (i = 0 to s by 1)
  for (j = 1 to p + r by 1)
    for (k = 0 to p by 1) M_e[j + (p + r)i][(k + p)i] = T[j][k];
```

Curve Composition: Now, we consider the knot removal algorithm. It is the reverse process of inserting a knot. While knot insertion is a precise procedure, i.e. the knot-inserted curve is precisely the same as the original one, knot removal, in general, produces an approximation of the original curve. Clearly, after a knot is inserted, it can be removed precisely.

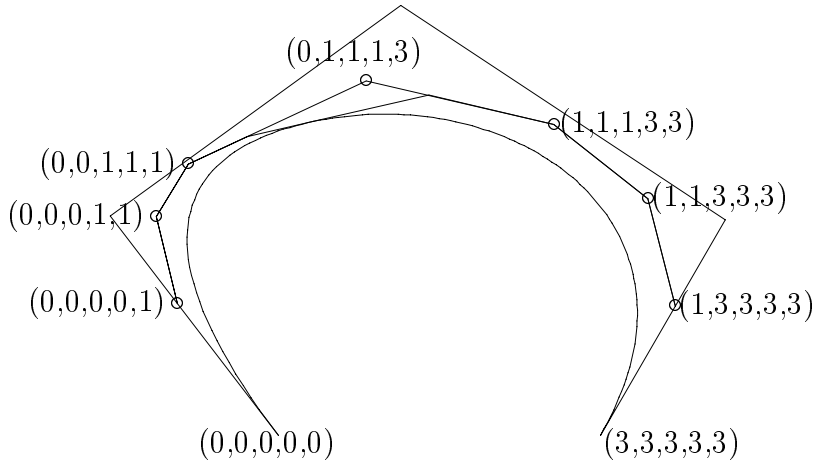

Fig. 3. Composition of the piecewise Bézier curves with the knot vector $(0, 0, 0, 0, 0, 0, 1, 1, 1, 3, 3, 3, 3, 3, 3)$.

Consider the curve shown in Fig. 3. The knot $u = 1$ is removable two times since the curve is C^2-continuous at $u = 1$. Curve composition is the inverse of decomposition. In Fig. 3, it can be expressed in a linear equation as

$$R = M_D \tilde{P}$$

where

$$R = \begin{pmatrix} B_G(0,0,0,0,0) \\ B_G(0,0,0,0,1) \\ B_G(0,0,0,1,1) \\ B_G(0,0,1,1,1) \\ B_G(0,1,1,1,1) \\ B_G(1,1,1,1,1) \\ B_G(1,1,1,1,3) \\ B_G(1,1,1,3,3) \\ B_G(1,1,3,3,3) \\ B_G(1,3,3,3,3) \\ B_G(3,3,3,3,3) \end{pmatrix}, M_D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4/9 & 4/9 & 1/9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } \tilde{P} = \begin{pmatrix} B_G(0,0,0,0,0) \\ B_G(0,0,0,0,1) \\ B_G(0,0,0,1,1) \\ B_G(0,0,1,1,1) \\ B_G(0,1,1,1,3) \\ B_G(1,1,1,3,3) \\ B_G(1,1,3,3,3) \\ B_G(1,3,3,3,3) \\ B_G(3,3,3,3,3) \end{pmatrix}.$$

Because M_D has full column rank and $M_D^t M_D$ is non-singular, $(M_D^t M_D)^{-1} M_D^t$ is a left inverse of the matrix M_D([18]). Therefore the consistent equation $R = M_D \tilde{P}$ have the unique solution

$$\tilde{P} = M_c R$$

where $M_c = (M_D^t M_D)^{-1} M_D^t$. This shows that

$$R = M_D (M_D^t M_D)^{-1} M_D^t R. \tag{2}$$

We write down the algorithm for the degree elevation of a B–spline curve in a shorthand notation.

(1) Make_DecomposeBsplineMatrix(knots, m, n, p) : M_d
with knots= $\{\underbrace{a, a, \ldots, a}_{p+1}, \underbrace{u_1, \ldots, u_1}_{m_1}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s}, \underbrace{b, b, \ldots, b}_{p+1}\}$

(2) Make_ElevateBezierMatrix(p, r, s) : M_e

(3) Make_DecomposeBsplineMatrix(knots, m+(s+2)r, n+(s+2)r, p+r) : M_D
with knots= $\{\underbrace{a, a, \ldots, a}_{p+r+1}, \underbrace{u_1, \ldots, u_1}_{m_1+r}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s+r}, \underbrace{b, b, \ldots, b}_{p+r+1}\}$

Then,

$$\tilde{P} = MP \tag{3}$$

where $M = M_c M_e M_d$.

References

[1] T.S. Arthanari, Y. Dodge, *Mathematical programming in statistics* (John Wiley & Sons, New York, 1981).

[2] P.J. Barry, An Introduction to Blossoming, in: R.N. Goldman, T. Lyche, Ed., *Knot Insertion and Deletion Algorithms for B–spline Curves and Surface*, (SIAM, 1993), 1-10.

[3] E. Cohen, T. Lyche, L.L. Schumaker, Algorithms for degree-rasing of splines, *ACM Trans. Graph.* **4**(1985), 171-181.

[4] W.L.F. Degen, Best approximations of parametric curve by spline, in: T. Lyche, L.L. Schumaker, Ed., *Mathematical Method in Computer Aided Design II* (Academic Press, New York, 1992).

[5] J. Dugundji, *Topology*, (Allyn and Bacon, Boston, 1966).

[6] J.D. Emery, The definition and computation of a metric on plane curves, *Comput. Aided Des.* **18**(1986), 25-28.

[7] G. Farin, *Curves and surfaces for computer aided geometric design : A practical guide*, (Academic Press, San Diego, 1993).

[8] G. Farin, *NURBS Curves and Surfaces : from projective geometry to practical use*, (AK Peters, Ltd., Wellesley, 1995).

[9] B.G. Lee, Y. Park, The Distance for Bézier Curves and Degree reduction, *Bull. Australian Math. Soc.* **56**(1997), 507-515.

[10] W. Liu, A simple, efficient degree rasing algorithm for B–spline curves, *Comput. Aided Geom. Des.* **14**(1997), 693-698.

[11] L. Piegl, W. Tiller, Software-engineering approach to degree elevation of B–spline curves, *Comput. Aided Des.* **26**(1994), 17-28.

[12] L. Piegl, W. Tiller, Algorithm for degree reduction of B–spline curves, *Comput. Aided Des.* **27**(1995), 101-110.

[13] L. Piegl, W. Tiller, *The NURBS Book*, (Springer-Verlag, 1995).

[14] H. Prautzsch, Degree elevation of B–spline curves, *Comput. Aided Geom. Des.* **2**(1984), 193-198.

[15] H. Prautzsch, B. Piper, A fast algorithm to raise the degree of B–spline curves, *Comput. Aided Geom. Des.* **8**(1991), 253-266.

[16] L. Ramshaw, *Blossoming : a connect-the-dots approach to splines*, (Technical report, Digital Systems Research Center, Palo Alto, 1987).

[17] L. Ramshaw, Blossoms are polar forms, *Comput. Aided Geom. Des.* **6**(1989), 323-359.

[18] S.R. Searle, *Linear Regression Analysis*, (John Wiley & Sons, New York, 1977).