Assignment #1: Free-form Surface Generation

- Make a program that
  - generates a bicubic Bezier surface from 4×4 defining polygon net points and
  - displays the surface graphically in a way that allows the three-dimensional features of the surface to appear comprehensively → You will need appropriate three-dimensional transformation and projection
- Suppose the 4×4 defining polygon net points are specified as $\boldsymbol{B}_{i,j} = [x_{i,j} \ \ y_{i,j} \ \ z_{i,j}]$ $(i, j = 1, 2, 3, 4)$; According to the definition of a Bezier surface, a point on the surface $\boldsymbol{P}(u, w)=[x_{u,w} \ \ y_{u,w} \ \ z_{u,w}]$ for the parameters u, w $(0 \leqq u \leqq 1, \ 0 \leqq w \leqq 1)$ is calculated
- By dividing the parameter range [0, 1] into 16 intervals, 17×17 points $[x_{u,w} \ \ y_{u,w} \ \ z_{u,w}]$ (u, w = 0, 1/16, 2/16, … 15/16 and 1) on the surface are calculated; By connecting the calculated points with lines and making a lattice, a mesh to approximate the Bezier surface is obtained
- The program you write will need to be able to edit the shape of the surface by changing the 4×4 defining polygon net points $\boldsymbol{B}_{i,j}$
- Because of the coordinate system of an NC milling machine and the work size subsequently used in this course, use the right-hand coordinate system, and the orientation and value range for the parameters x, y, and z should be as follows
  - x and y to control the horizontal range of the surface: $0 \leqq x \leqq 100$; y: $0 \leqq y \leqq 75$
  - z to control the convex/concave shape of the surface: $z \leqq 0$

- Submission
  - What is Expected
    - One printed copy of the graphics screen displaying the Bezier surface and defining polygon net
      - Do not submit the source code of your program at this time
      - The program is used for Assignment #2, so do not delete it after you finish Assignment #1
  - Deadline
    - June 30, 2004 (at the end of the lecture) in Room 8-222

- Bezier Curve
  - Definition
    - Defining polygon points: $\boldsymbol{B}_i = [x_i \;\; y_i \;\; z_i]$ (i = 0, 1 ,2, … , n)
    - Curve parameter: t ($0 \leqq t \leqq 1$)
    - Bezier/Bernstein basis function (blending function): $J_{n,i}(t)$
    - A point on the Bezier curve: $\boldsymbol{P}(t)=[x(t) \;\; y(t) \;\; z(t)]$

$$\boldsymbol{P}(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = \sum_{i=0}^{n} \boldsymbol{B}_i J_{n,i}(t) \qquad (0 \le t \le 1)$$

$$\boldsymbol{B}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix} \qquad J_{n,i}(t) = \binom{n}{i} t^i \,(1-t)^{n-i} \qquad \binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$

- Example: Cubic Bezier curve (n=3)

$$\boldsymbol{P}(t)=\sum_{i=0}^{3}\boldsymbol{B}_i J_{3,i}(t)=(1-t)^3\boldsymbol{B}_0+3t(1-t)^2\boldsymbol{B}_1+3t^2(1-t)\boldsymbol{B}_2+t^3\boldsymbol{B}_3$$

$$=\begin{bmatrix}(1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3\end{bmatrix}\begin{bmatrix}\boldsymbol{B}_0\\ \boldsymbol{B}_1\\ \boldsymbol{B}_2\\ \boldsymbol{B}_3\end{bmatrix}$$

$$\begin{cases}x(t)=(1-t)^3x_0+3t(1-t)^2x_1+3t^2(1-t)x_2+t^3x_3\\ y(t)=(1-t)^3y_0+3t(1-t)^2y_1+3t^2(1-t)y_2+t^3y_3\\ z(t)=(1-t)^3z_0+3t(1-t)^2z_1+3t^2(1-t)z_2+t^3z_3\end{cases}$$

  - $\boldsymbol{P}(t)$ is defined as a weighted sum of $\boldsymbol{B}_i$, where a weight for $\boldsymbol{B}_i$ is $J_{n,i}(t)$

- Some properties of the Bezier curve
  - The curve generally follows the shape of the defining polygon
  - Only the first and last points of the defining polygon and the curve are coincident
  - The degree of the polynomial defining the curve is one less than the number of defining polygon points
  - The curve is invariant under an affine transformation

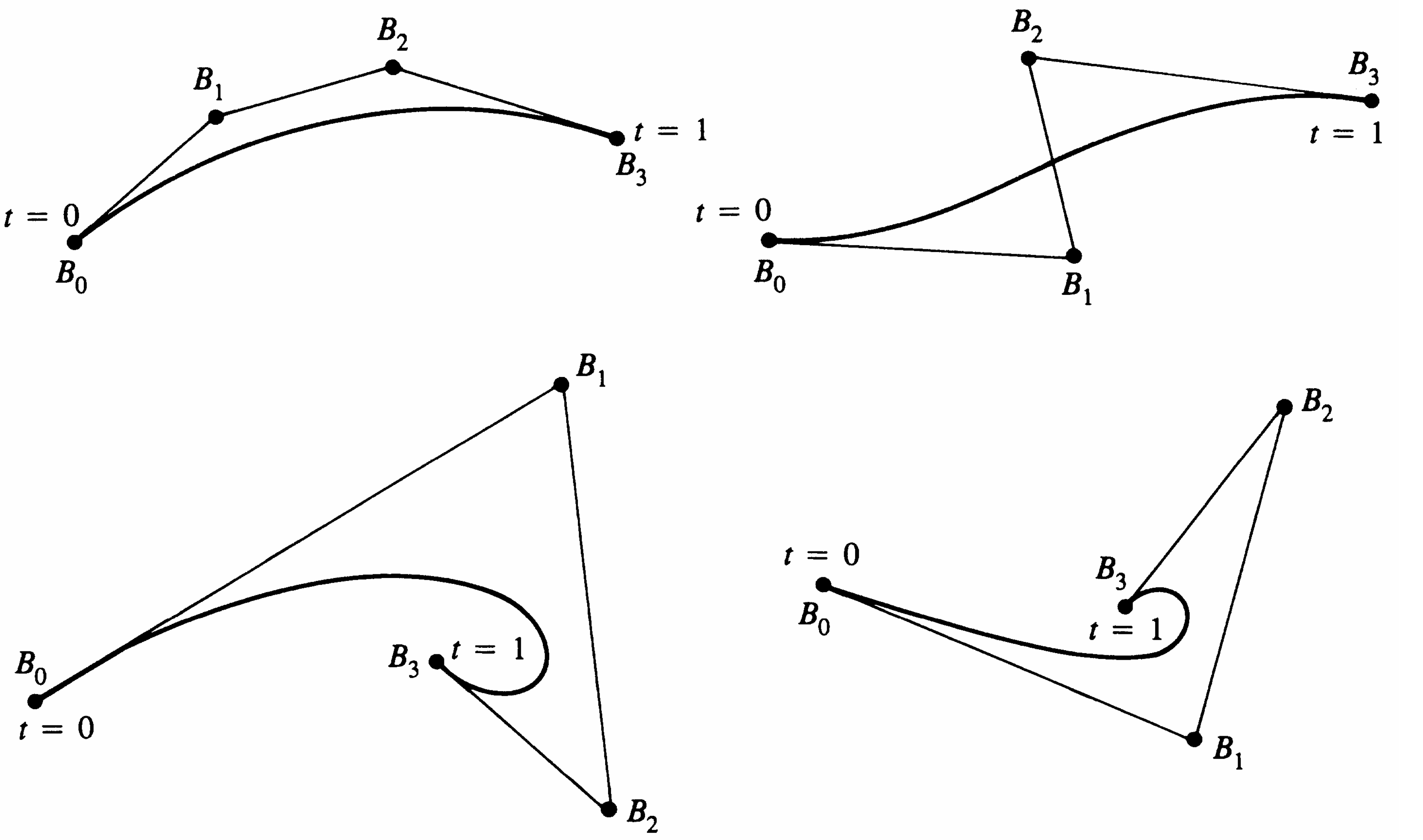

Figure 5–26 Bézier polygons for cubics. [Rogers and Adams 1990]

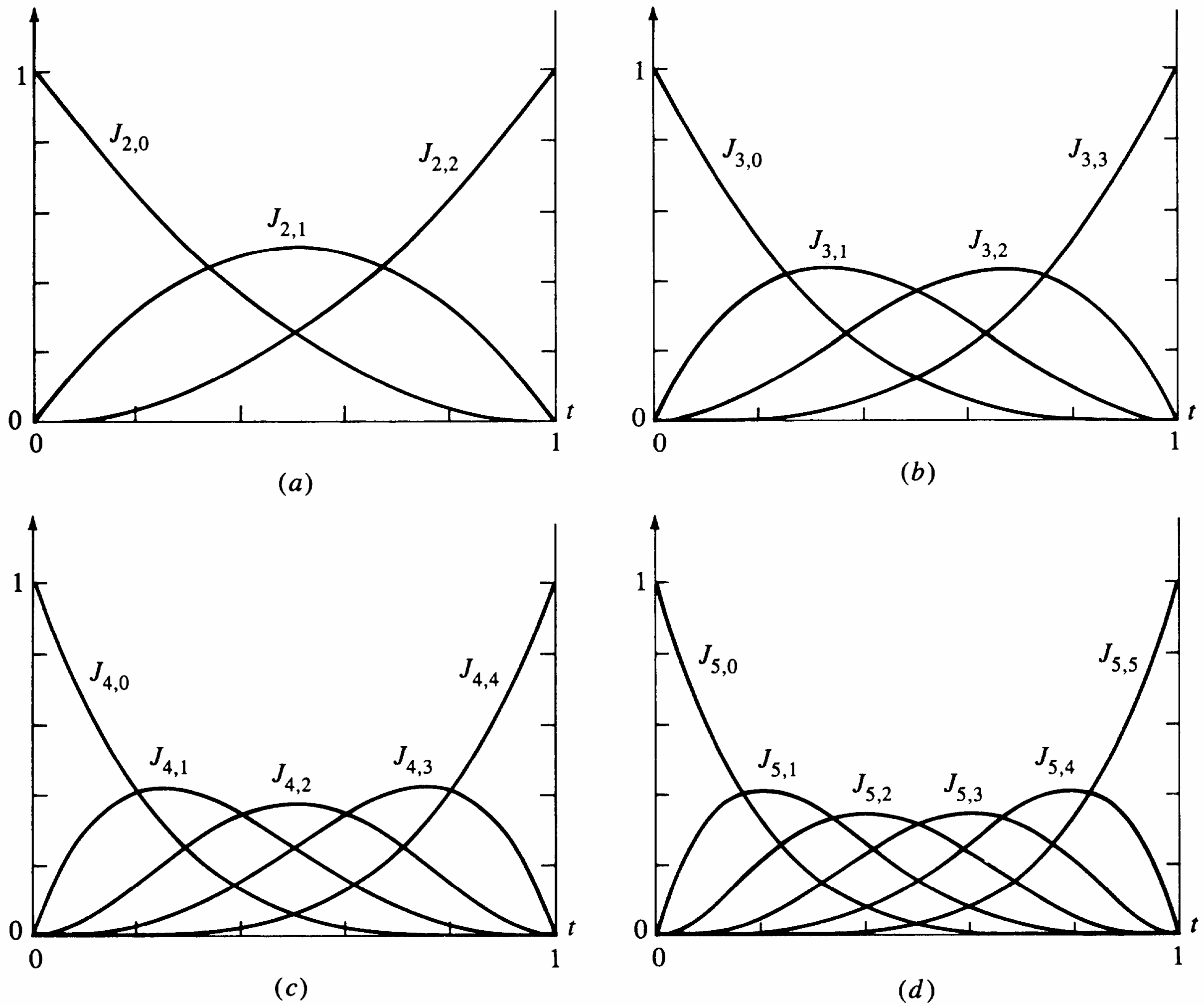

Figure 5–27 Bézier/Bernstein blending functions. (a) Three polygon points, $n = 2$; (b) four polygon points, $n = 3$; (c) five polygon points, $n = 4$; (d) six polygon points, $n = 5$.

[Rogers and Adams 1990]

Bezier surface

- Extension of a Bezier curve to the surface
- Definition
  - Defining polygon net points: $\boldsymbol{B}_{i,j} = [x_{i,j}\ \ y_{i,j}\ \ z_{i,j}]$ (i, j = 0, 1 ,2, … , n)
  - Surface parameters: u, w ($0 \leqq u \leqq 1$, $0 \leqq w \leqq 1$)
  - Bezier/Bernstein basis functions (blending functions): $J_{n,i}(u)$, $K_{m,j}(w)$
  - A point on the Bezier surface: $\boldsymbol{P}(u, w)=[x(u, w)\ \ y(u, w)\ \ z(u, w)]$

$$\boldsymbol{P}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} \boldsymbol{B}_{i,j} J_{n,i}(u) K_{m,j}(w) \qquad 0 \le u \le 1 \quad 0 \le w \le 1$$

$$\boldsymbol{B}_{i,j} = \begin{bmatrix} x_{i,j} & y_{i,j} & z_{i,j} \end{bmatrix} \qquad J_{n,i}(u) = \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i} \quad K_{m,j}(w) = \begin{pmatrix} m \\ j \end{pmatrix} w^j (1-w)^{m-j}$$

- Example: Bicubic Bezier surface (n=m=3)

$$\boldsymbol{P}(u,w) = \begin{bmatrix} J_{3,0}(u) & J_{3,1}(u) & J_{3,2}(u) & J_{3,3}(u) \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_{0,0} & \boldsymbol{B}_{0,1} & \boldsymbol{B}_{0,2} & \boldsymbol{B}_{0,3} \\ \boldsymbol{B}_{1,0} & \boldsymbol{B}_{1,1} & \boldsymbol{B}_{1,2} & \boldsymbol{B}_{1,3} \\ \boldsymbol{B}_{2,0} & \boldsymbol{B}_{2,1} & \boldsymbol{B}_{2,2} & \boldsymbol{B}_{2,3} \\ \boldsymbol{B}_{3,0} & \boldsymbol{B}_{3,1} & \boldsymbol{B}_{3,2} & \boldsymbol{B}_{3,3} \end{bmatrix} \begin{bmatrix} K_{3,0}(w) \\ K_{3,1}(w) \\ K_{3,2}(w) \\ K_{3,3}(w) \end{bmatrix}$$

$$= \begin{bmatrix} (1-u)^3 & 3(1-u)^2 u & 3(1-u)u^2 & u^3 \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_{i,j} \end{bmatrix} \begin{bmatrix} (1-w)^3 \\ 3(1-w)^2 w \\ 3(1-w)w^2 \\ w^3 \end{bmatrix}$$

- Some properties of the Bezier surface
  - The surface generally follows the shape of the defining polygon net
  - Only the corner points of the defining polygon net and the surface are coincident
  - The degree of the surface in each parametric direction is one less than the number of defining polygon net points in that direction
  - The surface is invariant under an affine transformation

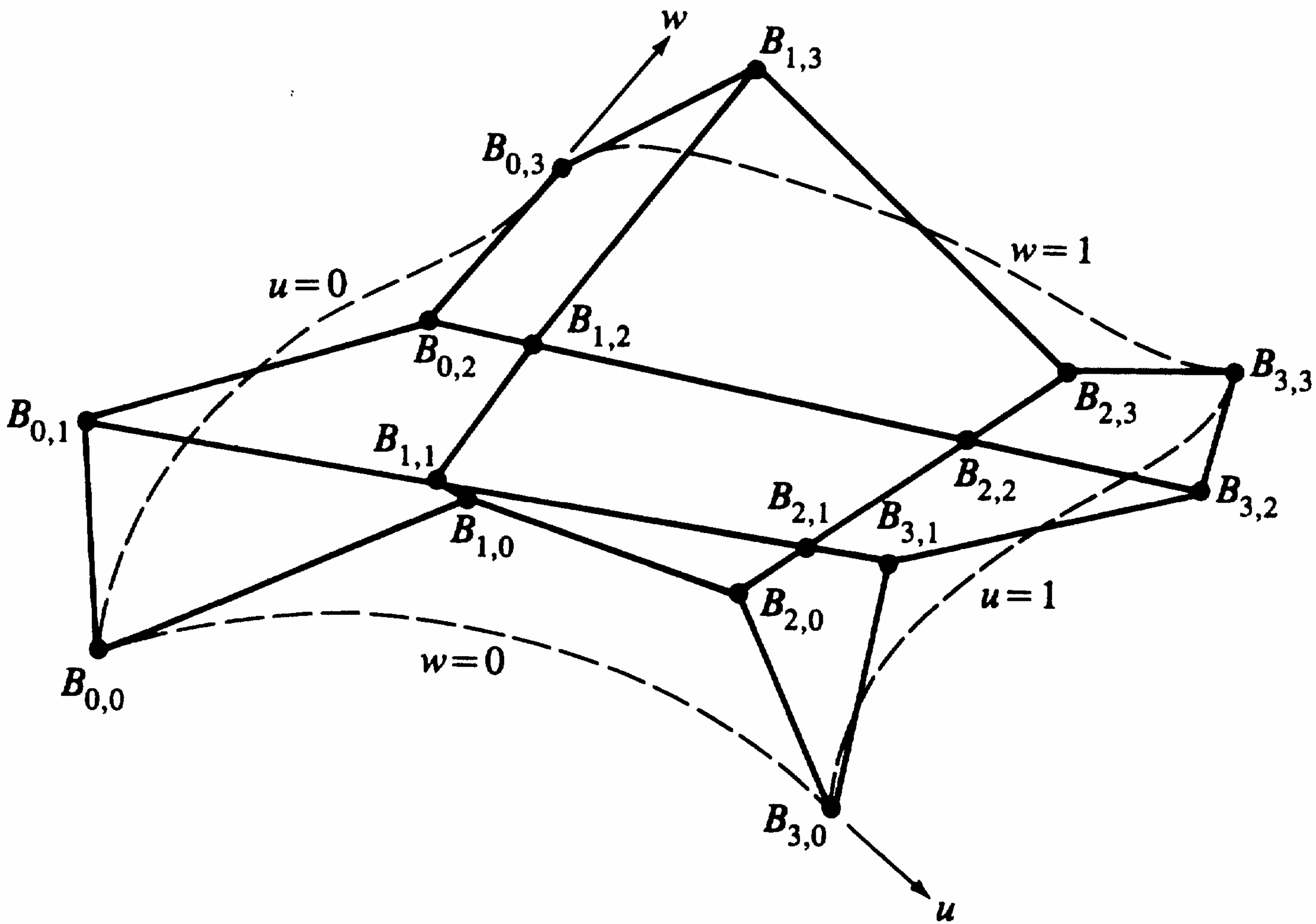

Figure 6–37 Bézier surface nomenclature. [Rogers and Adams 1990]

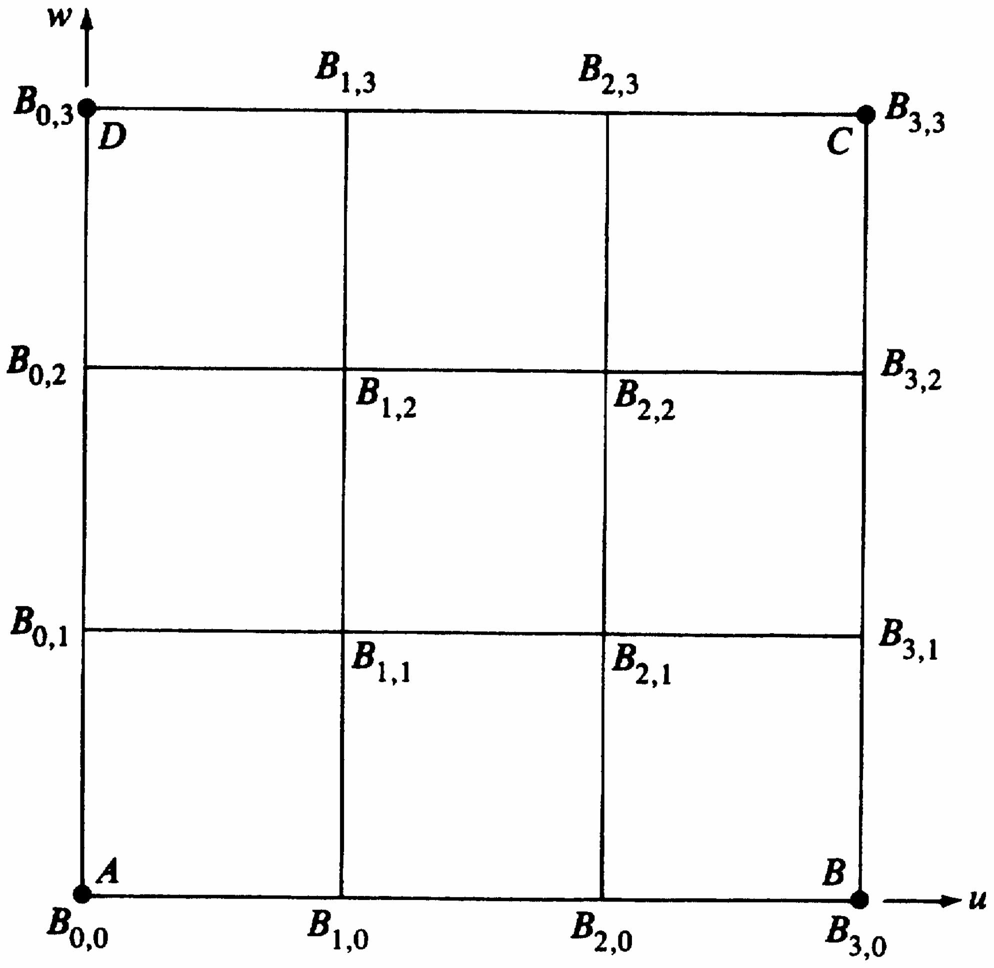

Figure 6–38 Schematic of the defining polygon net for a 4 × 4 Bézier surface.

[Rogers and Adams 1990]

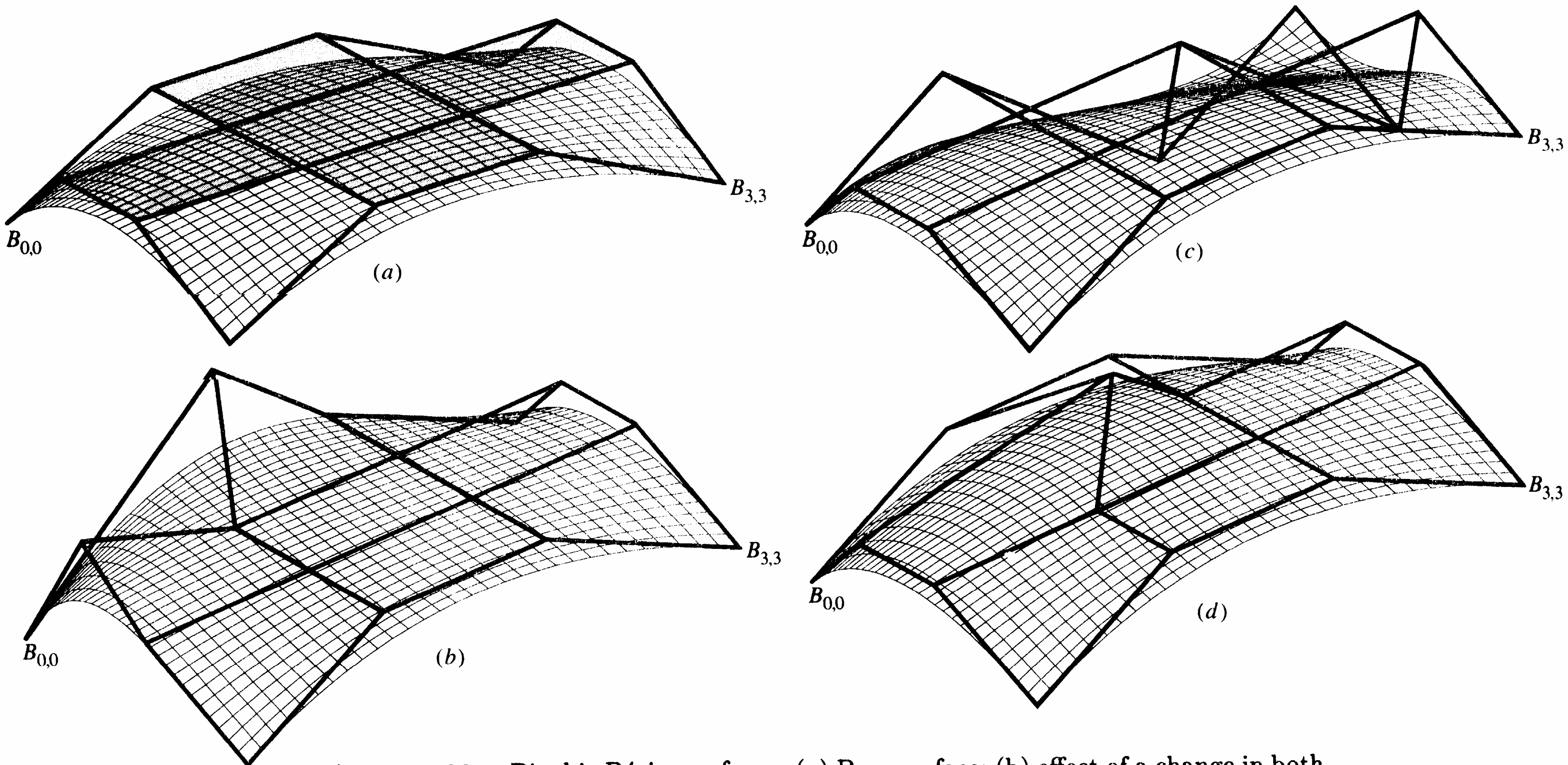

Figure 6–39 Bicubic Bézier surfaces. (a) Base surface; (b) effect of a change in both tangent vector magnitudes at $B_{0,0}$; (c) effect of a change in tangent vector direction at $B_{0,3}$; (d) effect of a change in twist vector magnitude at $B_{0,0}$.

[Rogers and Adams 1990]

- Three dimensional transformation
  - Ordinary coordinates: [*x* *y* *z*]
  - Homogeneous coordinates: [*x* *y* *z* 1]
  - 4×4 Transformation matrix: [***T***]

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1][\boldsymbol{T}]$$

  - Composition of transformations
    - Matrix multiplication

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1][\boldsymbol{T}_1][\boldsymbol{T}_2]\Lambda\,[\boldsymbol{T}_n]$$

  - Scaling

$$[\boldsymbol{T}] = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotation (about *x* axis)

$$[\boldsymbol{T}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotation (about *y* axis)

$$[\boldsymbol{T}] = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotation (about *z* axis)

$$[\boldsymbol{T}] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Translation

$$[\boldsymbol{T}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

- Reference

  - Rogers, D.F. and Adams, J.A., Mathematical Elements for Computer Graphics, 2nd ed., McGraw-Hill, 1990.

Implementation Hint

```
public class bicubic_bezier_surface
{
    private double basis_function(int i, double t)
    {
    }

    private vector3d surface_point(point_lattice polygon_net, double u, double w)
    {
        /* Loop for i and j */
        polygon_net.get_point(i, j);
        basis_function(i, u);
        basis_function(j, w);
    }

    public void draw(Graphics2D g2, point_lattice polygon_net, double[][] matrix)
    {
        /* Loop for u and w */
        surface_point(polygon_net, u, w).transform(matrix);
    }

}
```