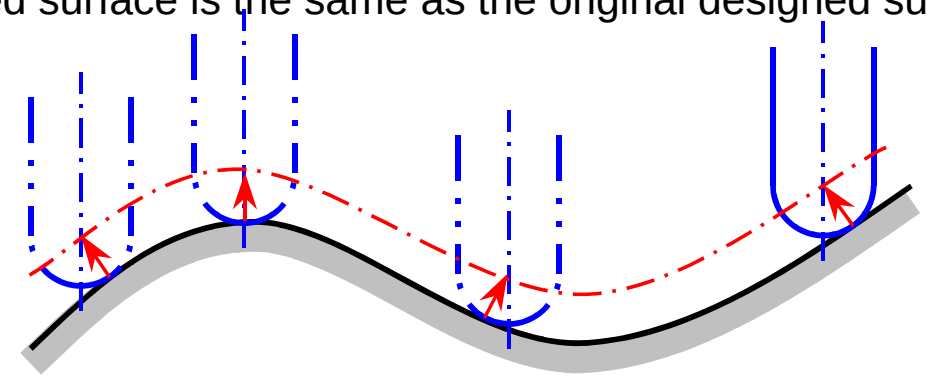
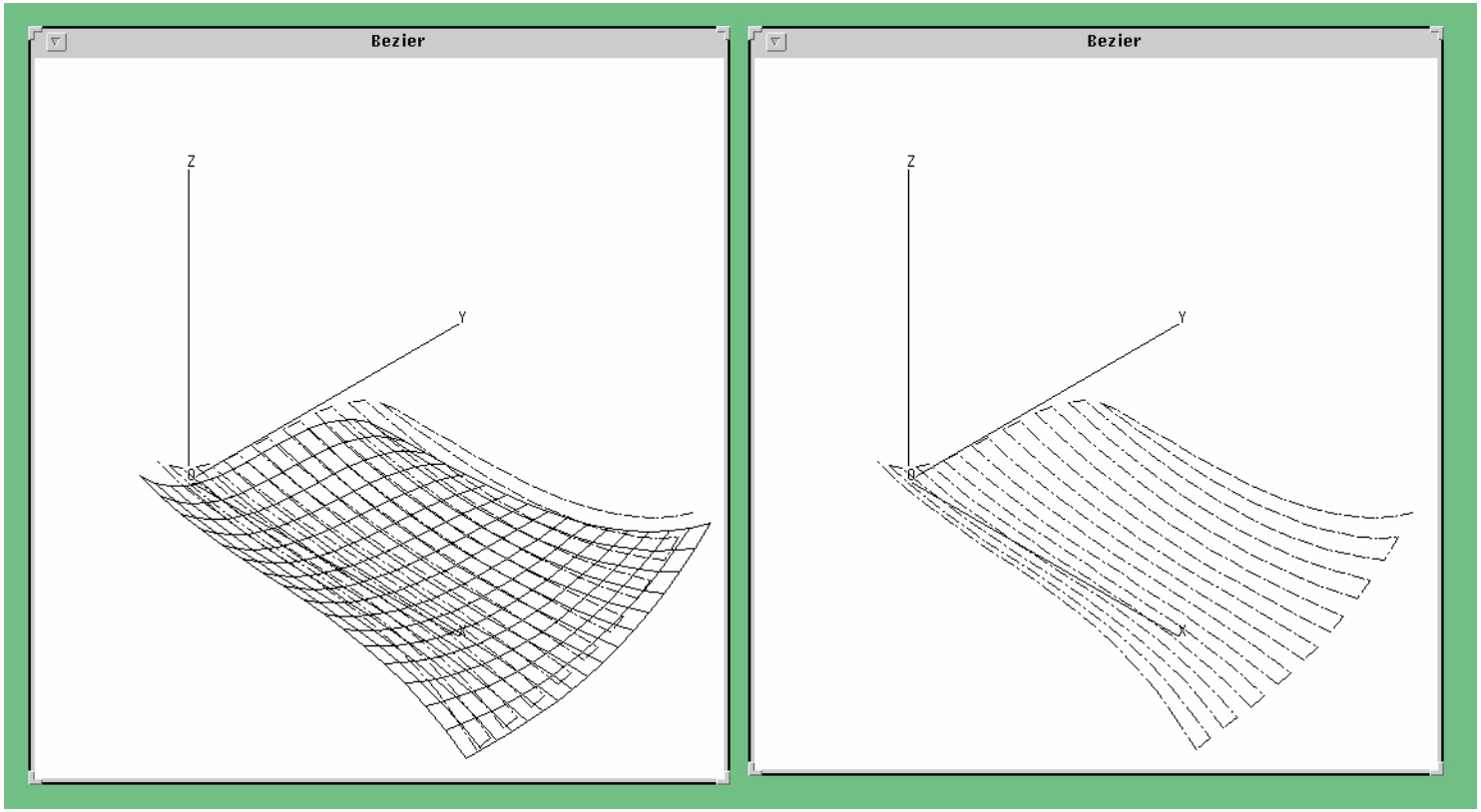
- Assignment #2: “Tool Path and NC Code Generation”
  - Make a program that
    - generates a tool path for milling a Bezier surface using a ball end mill,
    - shows the tool path graphically on a computer display, and
    - generates an NC program for the tool path as a text file

- Offset Surface and Tool Path
  - If the center of a spherical face at the end of a ball end mill is moved along the [*x y z*] points of a designed surface as they are, the manufactured surface is different from the designed surface (namely, the surface is cut too deeply)
  - A designed surface needs to be offset by the radius *r* of a spherical face of a ball end mill
  - If the center of a ball end mill is moved along the [*x y z*] points of the offset surface, the manufactured surface is the same as the original designed surface



- In Assignment #1, we already have mesh points on a Bezier surface
- In this, Assignment #2, please calculate an offset point for every mesh point on the Bezier surface; Then connect the offset points as a sequence to form a tool path



Offset Point of a Surface

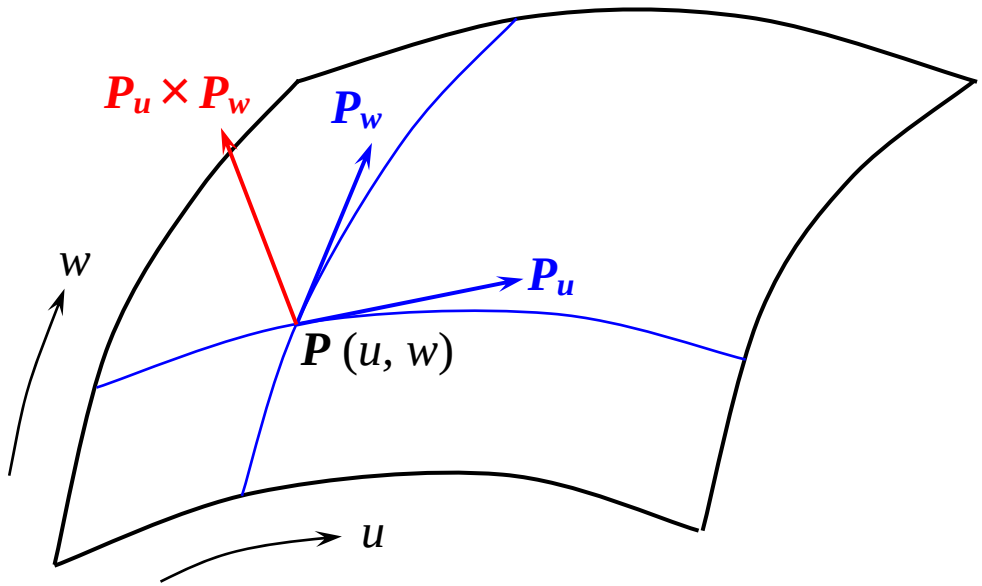
$$\boldsymbol{P}(u,w) = [\, x(u,w) \;\; y(u,w) \;\; z(u,w) \,]$$

$$\boldsymbol{P}_u = \frac{\partial \boldsymbol{P}(u,w)}{\partial u} = \left[\frac{\partial x(u,w)}{\partial u} \;\; \frac{\partial y(u,w)}{\partial u} \;\; \frac{\partial z(u,w)}{\partial u} \right]$$

$$\boldsymbol{P}_w = \frac{\partial \boldsymbol{P}(u,w)}{\partial w} = \left[\frac{\partial x(u,w)}{\partial w} \;\; \frac{\partial y(u,w)}{\partial w} \;\; \frac{\partial z(u,w)}{\partial w} \right]$$

$$\boldsymbol{e}(u,w) = \frac{\boldsymbol{P}_u \times \boldsymbol{P}_w}{\left| \boldsymbol{P}_u \times \boldsymbol{P}_w \right|}$$

$$\boldsymbol{P}_o(u,w) = \boldsymbol{P}(u,w) + \boldsymbol{e}(u,w) \cdot r$$



- $\boldsymbol{P}(u, w)$: a point on a surface
- $\boldsymbol{P}_u$: a tangent vector (in u direction) of the surface at $\boldsymbol{P}(u, w)$
- $\boldsymbol{P}_w$: a tangent vector (in w direction) of the surface at $\boldsymbol{P}(u, w)$
- $\boldsymbol{P}_u \times \boldsymbol{P}_w$: a normal vector of the surface at $\boldsymbol{P}(u, w)$
- $\boldsymbol{e}(u, w)$: a unit normal vector of the surface at $\boldsymbol{P}(u, w)$
- $\boldsymbol{P}_o$(u, w): an offset point of $\boldsymbol{P}(u, w)$
- Vector product: $\boldsymbol{v}_1 \times \boldsymbol{v}_2 = [x_1 \;\; y_1 \;\; z_1] \times [x_2 \;\; y_2 \;\; z_2] = [\, y_1 z_2 - z_1 y_2 \;\;\; z_1 x_2 - x_1 z_2 \;\;\; x_1 y_2 - y_1 x_2 \,]$

Partial Derivative of the Bezier Surface

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

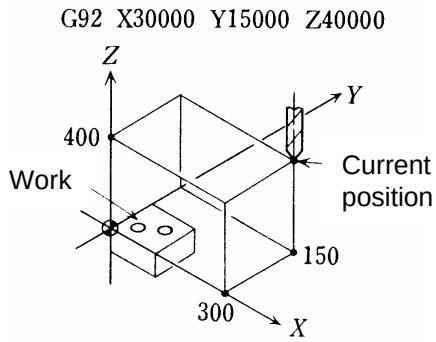
$$= \begin{bmatrix} (1-u)^3 & 3(1-u)^2 u & 3(1-u)u^2 & u^3 \end{bmatrix} \begin{bmatrix} \boldsymbol{B}_{i,j} \end{bmatrix} \begin{bmatrix} (1-w)^3 \\ 3(1-w)^2 w \\ 3(1-w)w^2 \\ w^3 \end{bmatrix}$$

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

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

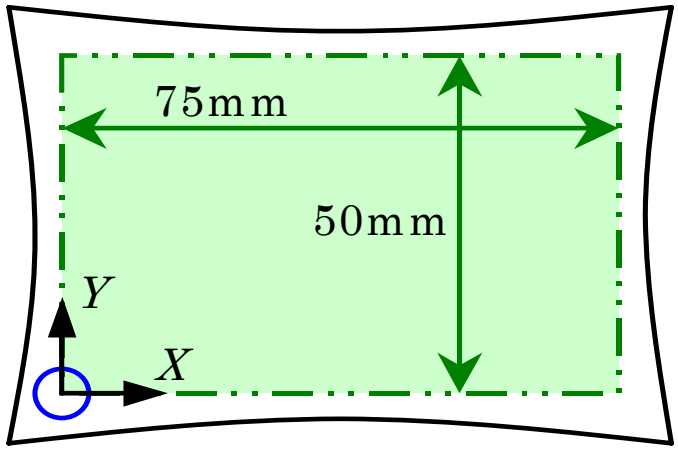
- NC Program
  - A program to control NC (Numerical Control) machine tools
  - The examples ("// Remarks") below are presented to help this handout; Your NC code must not include them

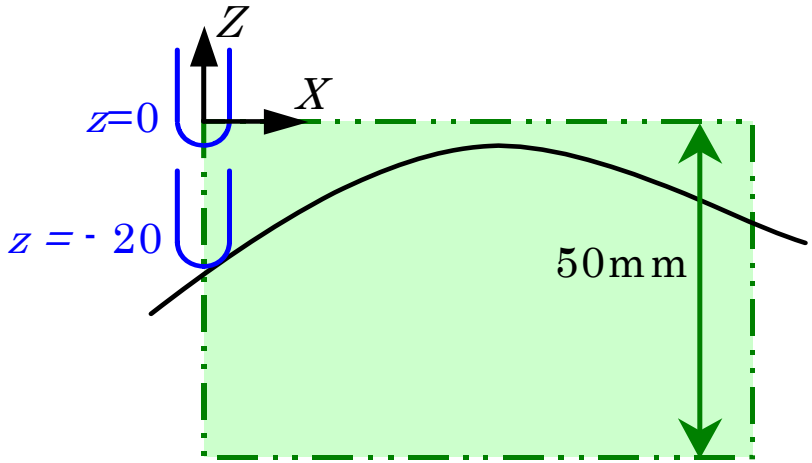
```
%;                          // Program start
G90;                        // G90: Absolute command, G91: Increment command
G92X0.0Y0.0Z50.0;           // G92: Absolute preset (Set current tool position as [0  0  50])
S300M03;                    // S: Spindle speed (rpm), M03: Spindle on clockwise
G00Z20.0;                   // G00: Positioning (Rapid feed to the position)
G01Z10.0F80;                // G01: Linear interpolation, F: Tool feed speed (mm/min)
                            // Your [x y z] sequence begins here
X-7.402Y-4.036Z-0.844;      // Tool position of the [x y z] sequence
X-0.297Y-4.34Z-4.523;
X6.762Y-4.709Z-7.214;
X13.808Y-5.134Z-9.09;
      :                     // Omitted to make the handout short
X86.192Y80.135Z-9.09;
X93.238Y79.71Z-7.214;
X100.298Y79.341Z-4.523;
X107.403Y79.037Z-0.844;
                            // Your [x y z] sequence ends here
G00Z50.0;                   // G00: Positioning (Rapid feed to the position)
X0.0Y0.0;                   // Tool position [x y z] (Return to the origin)
M05;                        // M05: Spindle off
M02;                        // M02: End of program
```



G92

- Detailed Requirement for Assignment #2
  - Java Program Implementation
    - Calculate 17×17 offset points and connect them as a sequence
    - Diameter d of a spherical face at the end of a ball end mill can be specified as a parameter
    - Three display modes can be selectively switched
      - Bezier surface
      - Tool path
      - Bezier surface and tool path
  - NC Program Generation
    - Use diameter d = 15mm
    - For x and y, the tool path ([x y z] sequence) should range a little wider than the work size $0\text{mm} \leqq x \leqq 75\text{mm}$, $0\text{mm} \leqq y \leqq 50\text{mm}$ so that no portion is left uncut
    - For z, the tool path should be within $-20\text{mm} \leqq z \leqq 0\text{mm}$ to avoid too deep (heavy) cutting
    - The tool path should start from the point closest to the origin [0 0 0] because the ball end mill is initially positioned at [0 0 10] (in mm)
    - Generate an NC program as a text file consisting of
      - The first six lines of the NC program example as they are given
      - Lines describing your [x y z] sequence
      - The last four lines of the NC program example as they are given





- Submission
  - Deadline: June 30
  - Printed copy (copies) of the two screens
    - (1) tool path, (2) Bezier surface and tool path
    - Hand this to me in Room #8-222, or send it by inter-department mail (“Gakunai-bin”)
  - NC program
    - Send a text file by e-mail to murakami@mech.t.u-tokyo.ac.jp; The program can be an e-mail text or an attachment file; Either way, be sure to write your name and student ID number.

- Schedule Change
  - No lecture (Q&A is OK) on June 30
  - Assignment #3 and instruction about an actual machining on July 7 will be sent you by e-mail

- Implementation Hint

```
public class vector3d
{
    /* Method to calculate multiplication of vector V and scalar A */
    /* Usage: Vm = V.multiply(A) */
    public vector3d multiply(double a)
    {
        double new_x, new_y, new_z;

        new_x = get_x() * a;
        new_y = get_y() * a;
        new_z = get_z() * a;
        return  new  vector3d(new_x, new_y, new_z);
    }
}

public  class  bicubic_bezier_surface
{
    private  vector3d  unit_normal(point_lattice  polygon_net,  double  u,  double  w)
    {
    }
```

- Implementation Hint (continued)

```
    private  vector3d  offset_point(point_lattice  polygon_net, double  u,  double  w, double r)
    {
          vector3d  p, un;
          p = surface_point(polygon_net,  u,  w);
          un = unit_normal(polygon_net,  u,  w);
          return  p.add( un.multiply( r ) );
    }

    public  void  draw_cutter_path(point_lattice  polygon_net, double u, double w, double r,
                                   double[][]  matrix)
    {
          offset_point(polygon_net,  u,  w,  r);
    }

    public  void  generate_nc_code(point_lattice  polygon_net, double u, double w, double r)
    {
          offset_point(polygon_net,  u,  w,  r);
    }

}
```