# TP 3 Modélisation : Surfaces paramétriques

Mateusz Birembaut

September 29, 2024

## Contents

# 1 Exercice 1 : Tracer une surface cylindrique

## 1.1 Cours :

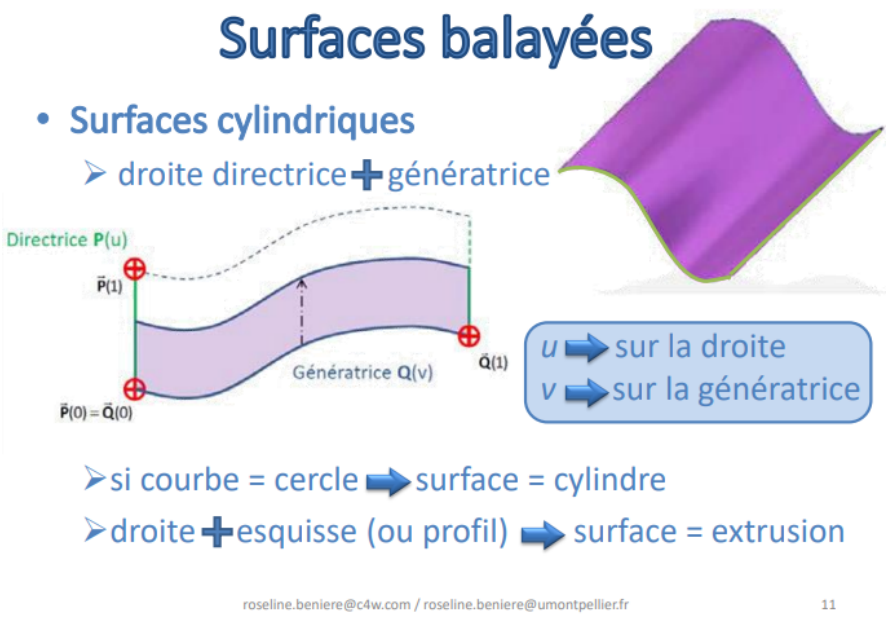


Figure 1: Rappel du cours sur les surfaces cylindriques

## 1.2 Fonction "surface cylindrique" :

En entrée : un tableau de points pour stocker les points de la surface, une courbe de Bézier (avec nbU points), une droite (définie par le premier point de contrôle et un vecteur), un nombre de points à calculer en u et un nombre de points à calculer en v.

En sortie : remplie le tableau "pts" représentant l'ensemble des points pour représenter la surface.

```
void surface_Cylindrique(Vec3* pts,Vec3* pts_bezier, Vec3 p0_droite, Vec3 v0_droite,int nbU, int nbV){
    for (int i = 0; i < nbU; i++) {
        Vec3 p_bezier = pts_bezier[i];

        for (int j = 0; j < nbV; j++) {
            float v = float(j) / (nbV - 1);

            Vec3 position = p_bezier + v * v0_droite;

            pts[i * nbV + j] = position;
        }
    }

}
```

Figure 2: Code fonction "surface cylindrique"

### 1.3 Résultat

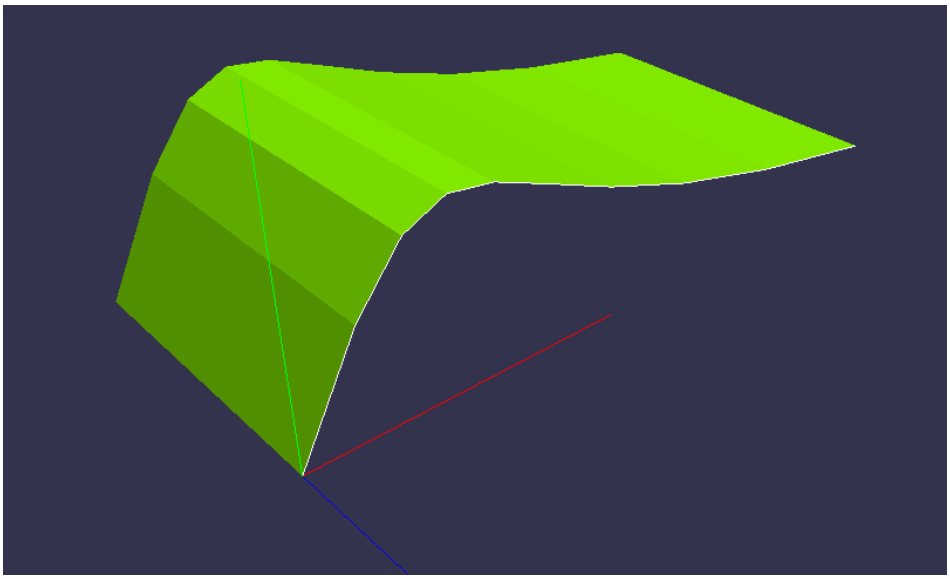

Figure 3: Surface cylindrique avec vecteur directeur (0,0,-1) et une courbe de Bézier en blanc

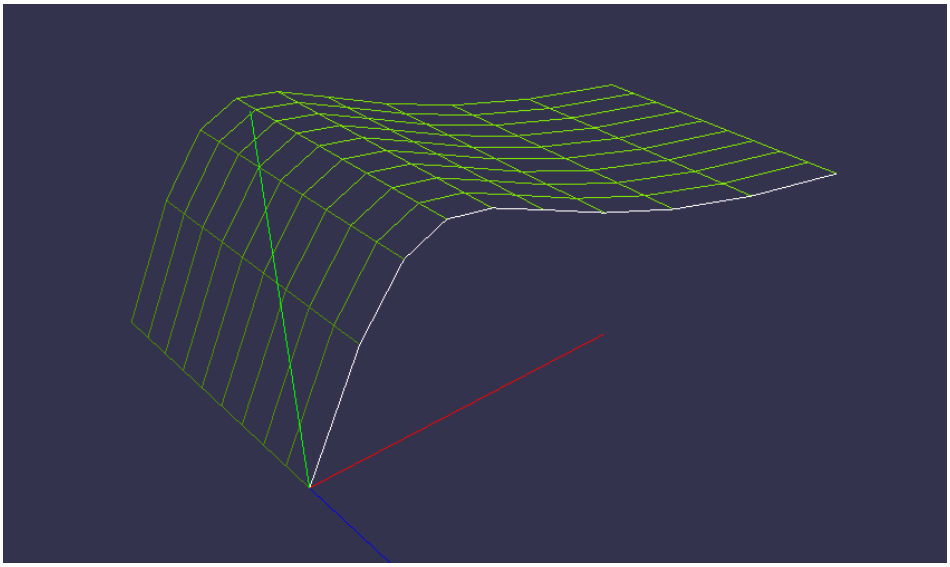

Figure 4: Surface cylindrique en affichant les quadrangles

# 2 Exercice 2 : Tracer une surface réglée

## 2.1 Cours :

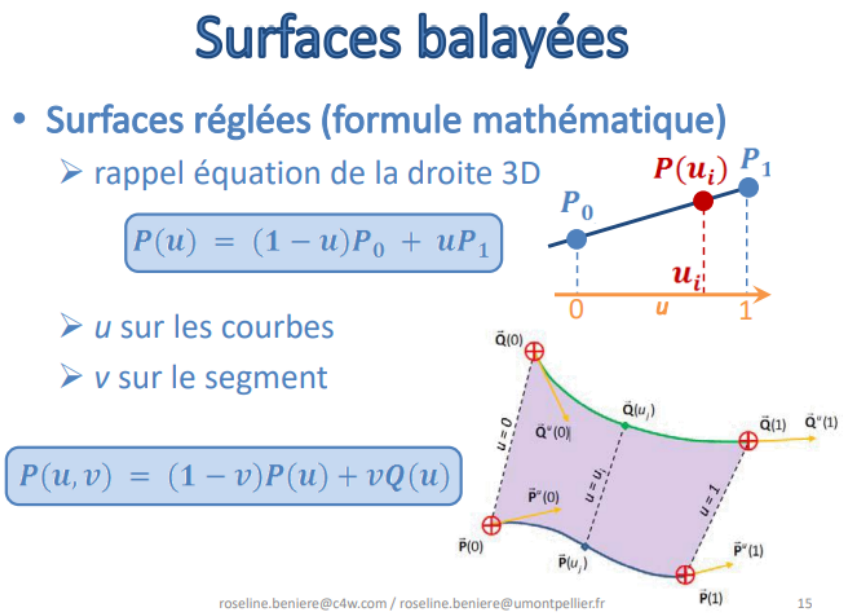


Figure 5: Rappel du cours sur les surfaces réglées

## 2.2 Fonction "surface Réglée" :

En entrée : un tableau de points pour stocker les points de la surface, deux courbes de Bézier (avec nbU points chacunes), un nombre de points à calculer en u et un nombre de points à calculer en v.

En sortie : remplie le tableau "pts" représentant l'ensemble des points pour représenter la surface.

```
void surface_Reglee(Vec3* pts,Vec3* pts_bezier_1 , Vec3* pts_bezier_2,int nbU, int nbV){
    for (int i = 0; i < nbU; i++) {
        Vec3 p_bezier_1 = pts_bezier_1[i];
        Vec3 p_bezier_2 = pts_bezier_2[i];

        for (int j = 0; j < nbV; j++) {
            float v = float(j) / (nbV - 1);

            Vec3 position = (1-v) * p_bezier_1 + v * p_bezier_2;

            pts[i * nbV + j] = position;
        }
    }

}
```

Figure 6: Code fonction "surface réglée"

## 2.3 Résultat

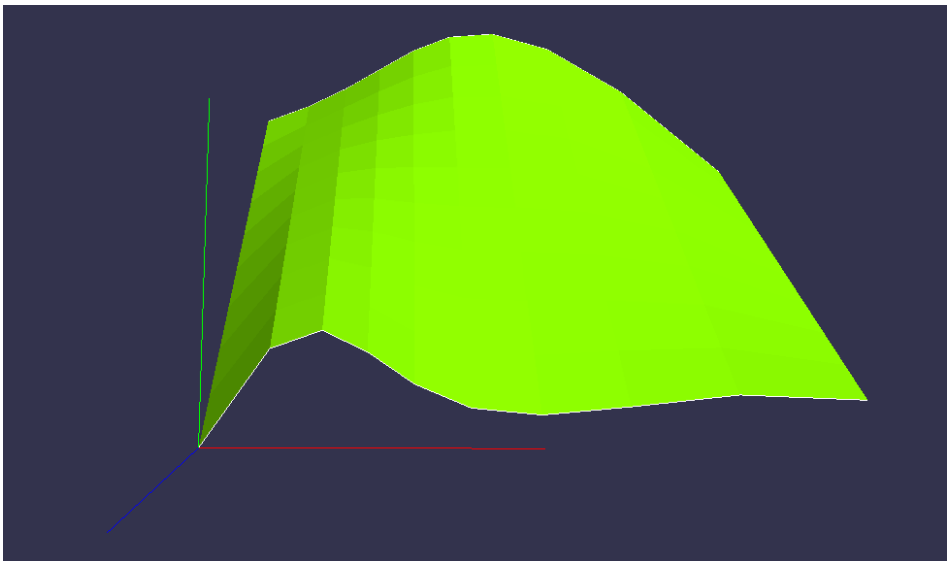

Figure 7: Surface réglée avec deux courbes de Bézier en blanc

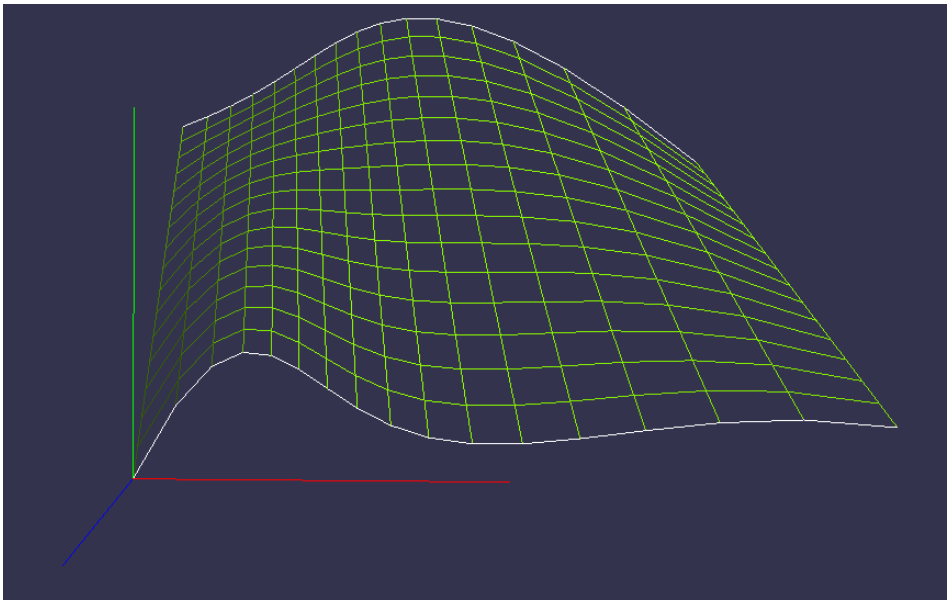

Figure 8: Surface réglée en affichant les quadrangles

# 3 Exercice 3 : Tracer une surface de Bézier par les polynômes de Bernstein

## 3.1 Cours :

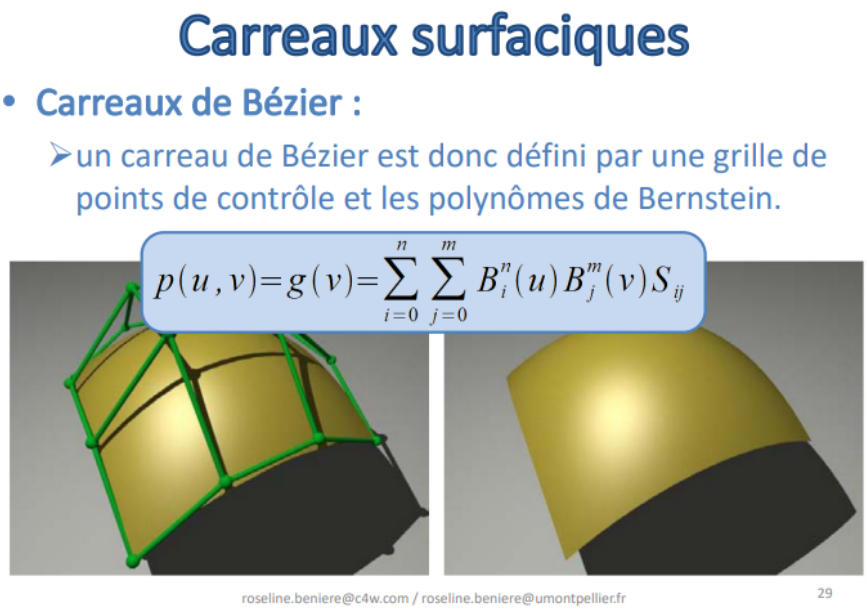


Figure 9: Rappel du cours sur les surfaces de Bézier

## 3.2 Fonction "surface Bézier" :

En entrée : un tableau de points pour stocker les points de la surface (de taille nbU * nbV), grille de points de contrôle de la surface (tableau à deux dimensions), le nombre de points de contrôle en u, le nombre de points de contrôle en v, un nombre de points à calculer en u et un nombre de points à calculer en v.

En sortie : remplie le tableau "pts" représentant l'ensemble des points pour représenter la surface.

```
void surface_Bezier(Vec3* pts,Vec3** pts_controle , int nb_pts_controles_u , int nb_pts_controles_v,  int nbU, int nbV){
    int n = nb_pts_controles_u-1; // degrès des courbes en u
    int m = nb_pts_controles_v-1; // degrès des courbes en v
    for (int k = 0; k < nbU; k++) {
        float u = float(k)/(nbU-1);
        for (int l = 0; l < nbV; l++) {
            float v = float(l)/(nbV-1);
            Vec3 p_u_v = Vec3(0.,0.,0.);
            for (int i = 0; i < nb_pts_controles_u; i++){
                for (int j = 0; j < nb_pts_controles_v; j++){
                    float Bu = ((factorial(n) / (factorial(i)*factorial(n - i))) * pow(u,i) * pow((1-u), n-i));
                    float Bv = ((factorial(m) / (factorial(j)*factorial(m - j))) * pow(v,j) * pow((1-v), m-j));
                    p_u_v = p_u_v + (  Bu * Bv * pts_controle[i][j] );
                }
            }
            pts[k * nbV + l] = p_u_v;
        }
    }
}
```

Figure 10: Code fonction "surface Bézie"

### 3.3 Résultat

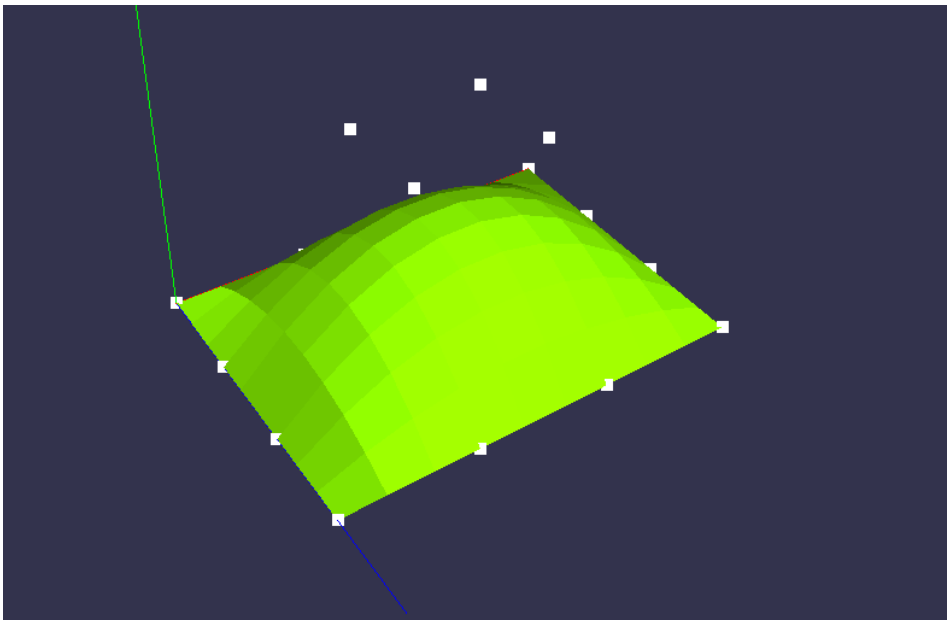

Figure 11: Surface de Bézier avec les points de contrôles en blanc

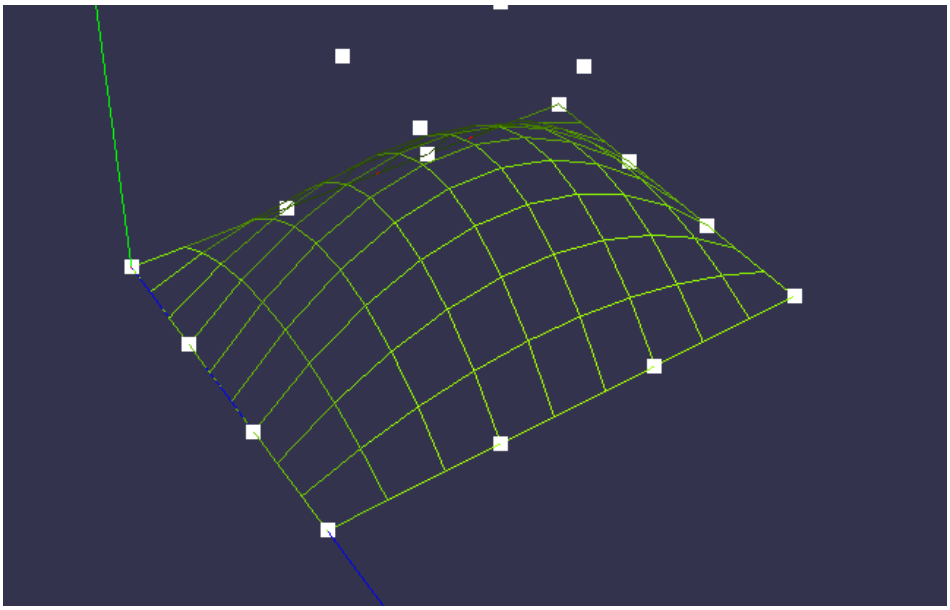

Figure 12: Surface de Bézier en affichant les quadrangles avec les points de contrôles en blanc

# 4 Fonctions Annexes

```
void DrawSurface(Vec3* pts, int nbU, int nbV, Vec3 color){
    glDisable(GL_CULL_FACE);
    glBegin(GL_QUADS);
    glColor3f(color[0], color[1], color[2]);
    for (int i = 0; i < nbU-1; i++){
        for (int j = 0; j < nbV-1; j++){

            Vec3 normal = calculateQuadNormal(
                pts[i * nbV + j],
                pts[(i + 1) * nbV + j],
                pts[(i + 1) * nbV + j + 1],
                pts[i * nbV + j + 1]
            );

            glNormal3f(normal[0], normal[1], normal[2]);

            glVertex3f(pts[i*nbV + j][0], pts[i*nbV + j][1], pts[i*nbV + j][2]);   // pos de base

            glVertex3f(pts[(i+1)*nbV + j][0], pts[(i+1)*nbV + j][1], pts[(i+1)*nbV +j][2]); // pos du haut

            glVertex3f(pts[(i+1)*nbV + j+1][0], pts[(i+1)*nbV + j+1][1], pts[(i+1)*nbV + j+1][2]);  // pos de droite haut

            glVertex3f(pts[i*nbV + j+1][0], pts[i*nbV + j+1][1], pts[i*nbV + j+1][2]);       // pos de droite

        }

    }
    glEnd();
}
```

Figure 13: Dessin de la surface et calcul de normales

```
if(display_surface_Reglee){

    Vec3 pts_ctrls_1[5] = { // pts de controle de la courbe 1
        Vec3(0., 0, 0.),
        Vec3(0.5, 1, 0.),
        Vec3(0.5, -0.5, 0.),
        Vec3(1, 0.3, 0.),
        Vec3(2, 0.15, 0.)
    };

    Vec3 pts_ctrls_2[5] = { // pts de controle de la courbe 2
        Vec3(0, 1.15, -1.5),
        Vec3(0.4, 1.3, -1.5),
        Vec3(0.7, 1.5, -1.5),
        Vec3(0.8, 2, -1.5),
        Vec3(2., 1, -1.5)
    };

    Vec3 pts[nbU * nbV]; // stocker les points de la surface
    Vec3 pointsBezier_1[nbU]; // stocker les points de la courbe 1
    Vec3 pointsBezier_2[nbU]; // stocker les points de la courbe 2

    BezierCurveByCasteljau(pointsBezier_1, pts_ctrls_1, 5, nbU); //récupérer les points de la courbe 1
    BezierCurveByCasteljau(pointsBezier_2, pts_ctrls_2, 5, nbU); //récupérer les points de la courbe 2

    surface_Reglee(pts, pointsBezier_1, pointsBezier_2, nbU, nbV); // remplir le tableau de points de la surface

    DrawCurve(pointsBezier_1, nbU, Vec3(1.0, 1.0, 1.0)); // dessiner la courbe 1
    DrawCurve(pointsBezier_2, nbU, Vec3(1.0, 1.0, 1.0)); // dessiner la courbe 1

    DrawSurface(pts, nbU, nbV, Vec3(0.5, 0.9, 0.0)); // dessiner la surface
    DrawAxes(); // dessiner les axes x y z
}
```

Figure 14: Exemple d'utilisation des fonctions (Ex 2)