哈爾濱工業大學

运动控制

题　　目　运动控制作业二报告

专　　业　控制科学与工程

学　　号　20S053293

学　　生　张凌玮

指 导 教 师　李建刚

日　　期　2020 年 12 月 22 日

目　录

第 1 章 作业思路

1.1 生成 B 样条曲面

作业拟生成二次 B 样条曲面，生成步骤如下：

（1）生成由坐标点组成的控制点线段。

（2）重复生成控制点线段：在 u 和 v 方向分别生成若干组控制点线段，组合在一起变成控制点网格。

（3）通过 B 样条曲线的计算公式，根据单根控制线段生成单根 B 样条曲线。

（4）在 u 和 v 方向根据控制线段生成多根 B 样条曲线，多根线条组合在一起最后变成 B 样条曲面。

1.2 计算路径坐标点及相关参数

生成 B 样条曲面后，为了方便计算加工路径，取 B 样条曲面与垂直于 Z 轴的平面相交而成的路径。计算步骤如下：

（1）为了精确求出 B 样条曲面与垂直于 Z 轴的平面的交点，将 u 和 v 方向的节点向量细分，平均分成若干份。在实验多次后，计算数据时取平均分为 2000 份；绘制图形时取平均分为 100 份，因为这样计算速度较快并且生成图形比较稠密。

（2）筛选 z=40 交线附近的坐标点，作为加工路径的细分点。

（3）求路径切线方向：在加工点足够稠密的时候，切线方向可以取近似为：

$$k = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

其中(x_n, y_n, z_n)和$(x_{n+1}, y_{n+1}, z_{n+1})$为路径上相邻两点的坐标。

（4）计算平面法线方向：平面法线方向直接计算不好计算，本实验先计算平面的梯度，然后根据已知的切线方向求得平面法线方向。梯度的近似求法将在下文详细说明。

（5）根据路径切线方向和平面法向量求得$b(s)$向量。

（6）根据求得的结果，输出 txt 文档将结果保存。

本实验的实现工具为MATLAB2020。没有使用C++程序语言的主要原始是C++没有现成的图像绘制库，可能需要借助其他库实现，过程繁琐复杂。《Nurbs》书本提供的 nurbs 库无法编译安装，并且相关教程稀少。

第 2 章 生成 B 样条曲面

2.1 生成控制点网

2.1.1 生成控制线段

控制点线段就是根据一组空间坐标点生成的空间线段，例如本实验取得其中一组空间坐标点：

$$[X\ Y\ Z] = transpose\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 \\ 0 & 0 & 5 & 15 & 30 & 35 & 30 & 15 & 5 & 0 & 0 \end{bmatrix}$$

生成空间坐标线段后可以得到如图 2-1 的图像：

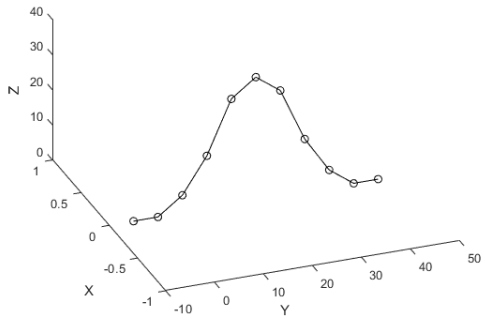

图 2-1 控制线段图像

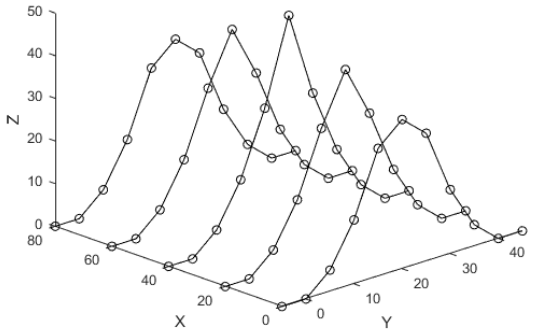

图 2-2 重复生成控制点线段

在获得单根控制线段图像后，重复多次，取不同的控制空间坐标，可以得到 u 方向上的多根控制线段，如图 2-2 所示。

2.1.2 生成控制点网

得到 u 方向上的多跟控制点线段后，在 v 方向重复步骤，可以得到如图 2-3 的控制点网格：

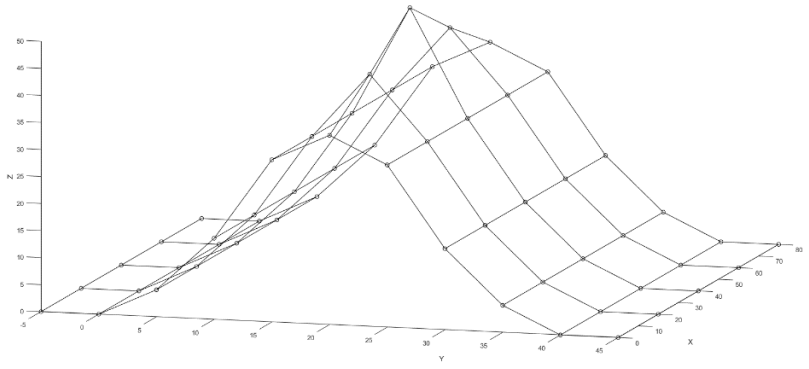

图 2-3 控制点网

2.2 生成 B 样条曲面

2.2.1 生成 B 样条曲线

B 样条曲线的定义公式（2-1）为：

$$C(u)=\sum_{i=0}^{n}N_{i,p}(u)P_i$$

其中

$u\in[a,b]$ —— u 为计算线段上的点

$$N_{i,0}(u)=\begin{cases}1, if\ u_i\le u\le u_{i+1}\\ 0, otherwise\end{cases}$$

$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u)$$

$U=\{a,\dots,a,u_{p+1},\dots,u_{m-p-1},b,\dots,b\}$ —— 节点向量

根据上述定义，在 MATLAB 中先编写节点向量$N_{i,p}(u)$计算函数（如图 2-4 所示），然后输入控制坐标点，计算绘画出单根 B 样条曲线。

```
function Nik_u = BaseFunction(i, k , u, NodeVector)
% 计算基函数Ni,k(u),NodeVector为节点向量

if k == 0       % 0次B样条
    if (u >= NodeVector(i+1)) && (u < NodeVector(i+2))
        Nik_u = 1.0;
    else
        Nik_u = 0.0;
    end
else
    Length1 = NodeVector(i+k+1) - NodeVector(i+1);
    Length2 = NodeVector(i+k+2) - NodeVector(i+2);      % 支撑区间的长度
    if Length1 == 0.0       % 规定0/0 = 0 防止报错
        Length1 = 1.0;
    end
    if Length2 == 0.0
        Length2 = 1.0;
    end
    Nik_u = (u - NodeVector(i+1)) / Length1 * BaseFunction(i, k-1, u, NodeVector) + (NodeVector(i+k+2) - u) / Length2 * BaseFunction(i+1, k-1, u, NodeVector);
end
```

图 2-4 节点向量计算函数

2.2.2 生成 B 样条曲面

在生成单根 B 样条曲线后，重复上述步骤，得到稀疏的 B 样条曲面网格。然后通过均匀化函数（如图 2-5）将 u 和 v 方向的节点向量均分成若干份，可以得到稠密的 B 样条曲面，如图 2-6.最后生成的 B 样条曲面极其控制网格如图 2-7 所示。

```
% u方向的细分
NodeVector_u = linspace(0, 1, n+k+2);      % 均匀B样条的u向节点矢量
u = linspace(k/(n+k+1), (n+1)/(n+k+1), piece_u);     % u向节点分成若干份

X_M_piece = zeros(M, piece_u);     % 沿着u方向的网格，M * piece
Y_M_piece = zeros(M, piece_u);
Z_M_piece = zeros(M, piece_u);
for i = 1 : M
    for j = 1 : piece_u
        for ii = 0 : 1: n
            Nik_u(ii+1, 1) = BaseFunction(ii, k , u(j), NodeVector_u);
        end
        X_M_piece(i, j) = X(i, :) * Nik_u;
        Y_M_piece(i, j) = Y(i, :) * Nik_u;
        Z_M_piece(i, j) = Z(i, :) * Nik_u;
    end
end
```

图 2-5 细分节点向量

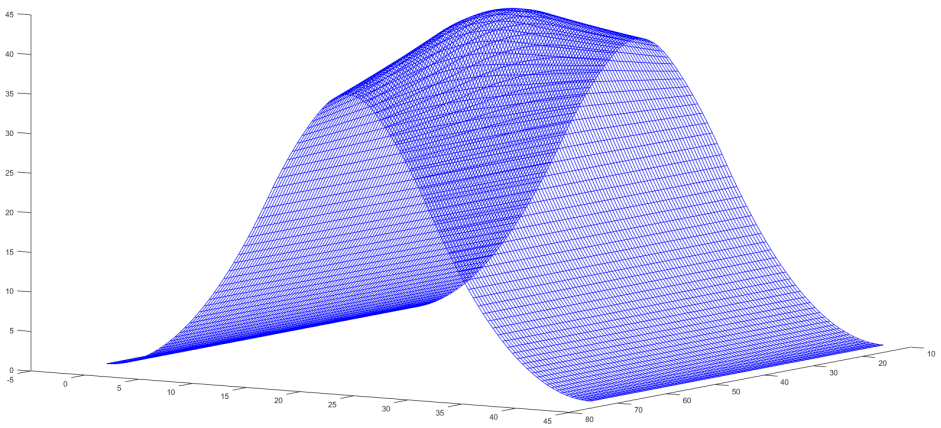

图 2-6 B 样条曲面

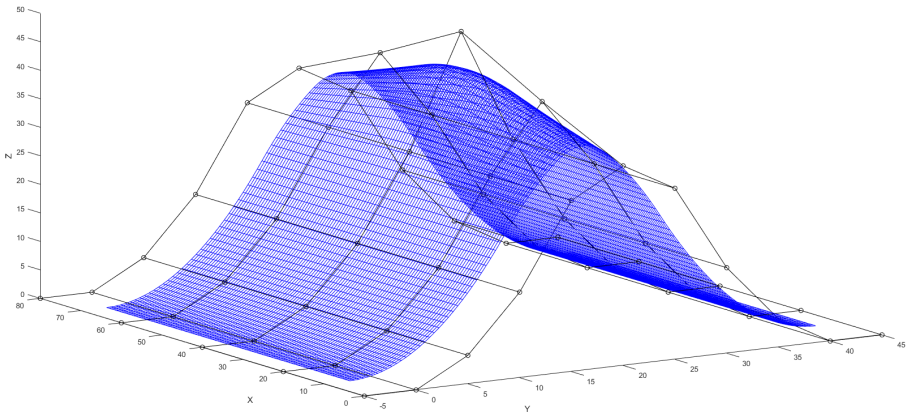

图 2-7 B 样条曲面极其控制点网

第 3 章 求解路径坐标及相关参数

3.1 求解路径坐标

为了方便计算，根据控制点坐标参数，选取加工路径为 B 样条曲面与 Z=40 的平面的相交线，在 MATLAB 绘图如图 3-1 所示。

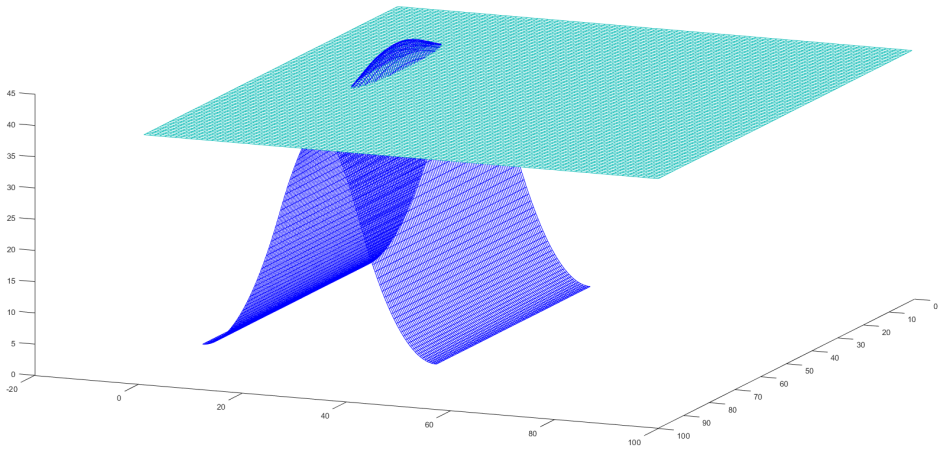

图 3-1 平面与曲面相交

在得到平面曲面相交示意图后，进行路径求解。因为 MATLAB 求解是一个个数据点，为了保证数据点足够精确，计算时选取将 u 和 v 方向的节点向量均分为 2000 份（画图时取的 100）。然后计算每个小节点的空间坐标，图 3-2 为每一个小节点 z 坐标。

39.8889	39.8970	39.9045	39.9115	39.9180	39.9239	39.9293
39.8975	39.9055	39.9131	39.9201	39.9265	39.9325	39.9379
39.9060	39.9141	39.9217	39.9287	39.9351	39.9411	39.9465
39.9146	39.9227	39.9302	39.9372	39.9437	39.9497	39.9550
39.9232	39.9312	39.9388	39.9458	39.9523	39.9582	39.9636
39.9317	39.9398	39.9474	39.9544	39.9609	39.9668	39.9722
39.9403	39.9484	39.9560	39.9630	39.9695	39.9754	39.9808
39.9488	39.9570	39.9645	39.9716	39.9781	39.9840	39.9894
39.9574	39.9655	39.9731	39.9802	39.9867	39.9926	39.9980
39.9660	39.9741	39.9817	39.9888	39.9953	40.0012	40.0067
39.9746	39.9827	39.9903	39.9974	40.0039	40.0098	40.0153
39.9831	39.9913	39.9989	40.0059	40.0125	40.0184	40.0239
39.9917	39.9999	40.0075	40.0145	40.0211	40.0270	40.0325
40.0003	40.0084	40.0161	40.0231	40.0297	40.0357	40.0411
40.0089	40.0170	40.0247	40.0317	40.0383	40.0443	40.0497
40.0174	40.0256	40.0332	40.0403	40.0469	40.0529	40.0583
40.0260	40.0342	40.0418	40.0489	40.0555	40.0615	40.0669

图 3-2 节点 Z 坐标

39.9443	39.9674	39.9899
39.9526	39.9757	39.9983
39.9609	39.9840	40.0066
39.9692	39.9924	40.0150
39.9775	40.0007	40.0233
39.9858	40.0090	40.0317
39.9941	40.0173	40.0400
40.0024	40.0257	40.0484

图 3-3 筛选示意图

计算得到每一个点的 z 坐标后，筛选 z 坐标为 40 附近的点。为了找出两个面的相交临界点，采用如下的筛选规则：

（1）将 Z 坐标数据表分为四部分：左上、右上、左下、右下部分对应 B 样条曲面的左上、右上、左下、右下半曲面。

（2）分别对左上、右上、左下、右下部分采取类似的筛选方式。例如左上半曲

面：输入 Z 坐标值，其对应矩阵行数和列数分别为 *i* 和 *j*，然后计算其*(i+1,j)*数据是否大于 40；若大于 40，再计算其*(i,j-1)(i-1,j)(i-1,j-1)*值是否小于 40，若都小于 40，则保留*(i,j)*对应的 Z 值，并记录对应的 X 值和 Y 值，如图 3-3 所示。算法如图 3-4 所示。

（3）筛选出来符合要求的数据点后，根据路径顺序排序出路径点。其算法如图 3-5 所示。

```
%%
%筛选边界点
for i = 2:piece_u-1
    for j = 2:piece_v-1
        if((Z_MN_piece(i, j)>=40 && Z_MN_piece(i-1, j-1)<40 && Z_MN_piece(i-1, j)<40 && Z_MN_piece(i, j-1)<40) ...
                || (Z_MN_piece(i, j)>=40 && Z_MN_piece(i-1, j+1)<40 && Z_MN_piece(i, j+1)<40  && Z_MN_piece(i-1, j)<40) ...
                ||(Z_MN_piece(i, j)>=40 && Z_MN_piece(i+1, j)<40 && Z_MN_piece(i+1, j-1)<40 && Z_MN_piece(i, j-1)<40) ...
            ||(Z_MN_piece(i, j)>=40 && Z_MN_piece(i+1, j+1)<40 && Z_MN_piece(i+1, j)<40 && Z_MN_piece(i, j+1)<40))
        a(t)= X_MN_piece(i, j);
        b(t)= Y_MN_piece(i, j);
        c(t)= Z_MN_piece(i, j);
        if(X_MN_piece(i, j)<=40 && Y_MN_piece(i, j)>20)
            grade_x(t) = X_MN_piece(i-1, j+1) - X_MN_piece(i, j);
            grade_y(t) = Y_MN_piece(i-1, j+1) - Y_MN_piece(i, j);
            grade_z(t) = Z_MN_piece(i-1,j+1) - Z_MN_piece(i,j);
```

图 3-4 筛选算法

```
for i = 1:360
    if(point(i,1) <= 40 && point(i,2) > 20)
        point_4(order,:) = [point(i,1) point(i,2) point(i,3) point(i,4) point(i,5) point(i,6)];
        order = order + 1;
    end
end

[point_3_size,~]=size(point_3);
[point_4_size,~]=size(point_4);

for i =1: point_3_size
    point_right_3(i,:) = point_3(end-i+1,:);
end

for i =1: point_4_size
    point_right_4(i,:) = point_4(end-i+1,:);
end
```

图 3-5 路径排序算法

经过上述步骤后，可以得到平面与曲面的相交路径图，如图 3-6 所示。可以看出，筛选出来的路径点比较稠密，并且在相同 X，Y 值得情况下没有重复得 Z 值——路径点可以组成一条均匀的曲线。得到路径曲线后，下一步进行计算相关参数。

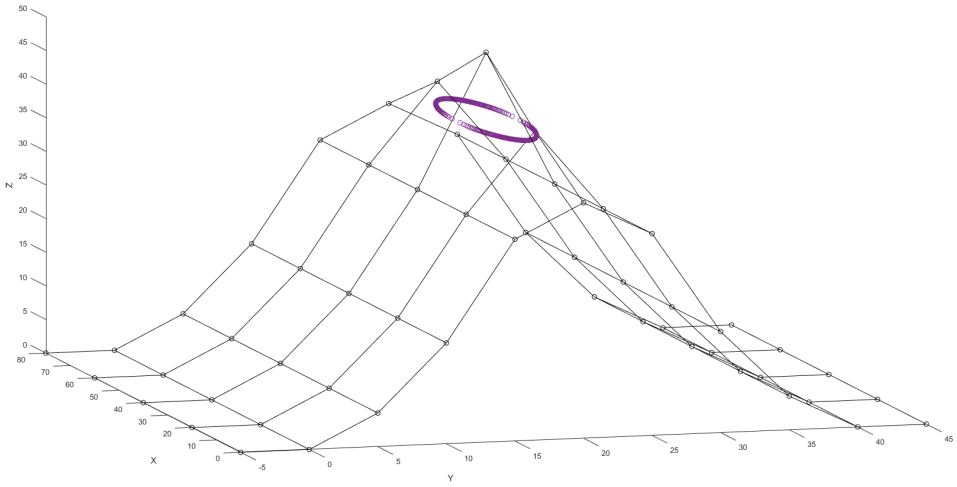

图 3-6 路径点图

3.2 求解相关参数

3.2.1 速度方向（路径切线方向）

上一小节已经求得路径点得空间坐标，其比较稠密，所以求速度方向的时候使用斜率的定义来近似：

$$k = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

其中(x_n, y_n, z_n)和$(x_{n+1}, y_{n+1}, z_{n+1})$为路径上相邻两点的坐标。算法如图 3-7 所示。

```
%速度
speed(1,:) = [points(1,1)-points(360,1) points(1,2)-points(360,2) 0];
for i = 2:360
    speed(i,:)=[points(i,1)-points(i-1,1) points(i,2)-points(i-1,2) 0];
end
```

图 3-7 速度算法

3.2.2 曲面法线

B 样条曲面的导数很难求出，$k(s)$也相当难求。本实验过程先求出路径点在 B 样条曲面的最速下降梯度，然后根据式子 3-1 求出路径点对应的法向量。

$$n(s) = gradient(s) \times speed(s) \qquad (3-1)$$

直接通过 B 样条曲面的解析式求出每个路径点的对应最速下降梯度也很困难，所以本实验采用了一种近似求解的方法：与筛选路径点的方法类似，将 Z 坐标值分为左上、右上、左下、右下四部分，对应 B 样条曲面的左上、右上、左下、右下四半曲面。位于左上部分的路径坐标点与其左上角的数据点相减，通过式子 3-2 计算出来梯度；位于右上部分的路径坐标点与其右上角的数据点相减，计算出梯度；位于左下部分的路径坐标点与其左下角的数据点相减，计算出梯度；位于右下部分的路径坐标点与其右下角的数据点相减，得到梯度。以左上部分为例，如图 3-8 所示。算法如图 3-9 所示。

$$gradient = \left(x_{i,j} - x_{i-1,j-1}, y_{i,j} - y_{i-1,j-1}, z_{i,j} - z_{i-1,j-1}\right) \qquad (3-2)$$

39.9341	39.9384	39.9422	39.9454	39.9481	39.9502	39.9519
39.9427	39.9470	39.9508	39.9540	39.9567	39.9589	39.9605
39.9513	39.9556	39.9594	39.9626	39.9653	39.9675	39.9691
39.9599	39.9642	39.9680	39.9712	39.9739	39.9761	39.9777
39.9685	39.9728	39.9766	39.9798	39.9825	39.9847	39.9863
39.9771	39.9814	39.9852	39.9885	39.9912	39.9933	39.9949
39.9857	39.9900	39.9938	39.9971	39.9998	40.0019	40.0036
39.9943	39.9986	40.0024	40.0057	40.0084	40.0106	40.0122
40.0029	40.0073	40.0110	40.0143	40.0170	40.0192	40.0208
40.0115	40.0159	40.0197	40.0229	40.0256	40.0278	40.0294
40.0201	40.0245	40.0283	40.0315	40.0343	40.0364	40.0381
40.0288	40.0331	40.0369	40.0402	40.0429	40.0451	40.0467
40.0374	40.0417	40.0455	40.0488	40.0515	40.0537	40.0553

图 3-8 左上部分计算梯度原理图

```
if(X_MN_piece(i, j)<=40 && Y_MN_piece(i, j)>20)
    grade_x(t) = X_MN_piece(i-1, j+1) - X_MN_piece(i, j);
    grade_y(t) = Y_MN_piece(i-1, j+1) - Y_MN_piece(i, j);
    grade_z(t) = Z_MN_piece(i-1,j+1) - Z_MN_piece(i,j);
    if grade_z(t)>0
        grade_z(t) = -grade_z(t);
    end
end
```

图 3-9 梯度计算算法（左上部分）

在求得每一个路径点对应的梯度后，通过式子 3-1 可以求得每一个路径点的曲面法向量。算法语句如图 3-10 所示。

```
%法向里
for i =1:360
    normal_vector(i,:)=cross(gradient(i,:),speed(i,:));
end
```

图 3-10 法向量算法

计算得到每一个路径点的法向量后，通过式子 3-3 可以求出$b(s)$向量，其算法如图 3-11 所示。在求得路径点切向方向向量、路径点平面法向向量和$b(s)$向量后，统一进行单位化得到最后结果。

$$b(s) = speed(s) \times n(s) \qquad (3-3)$$

```
for i = 1:360
    b_vector(i,:) = cross(speed(i,:),normal_vector(i,:));
end
```

图 3-11 $b(s)$向量算法

第 4 章 总结

本实验先生成控制线段，然后重复步骤生成控制点网；在生成控制点网后根据单根 B 样条曲线的原理，生成单根 B 样条曲线，然后重复步骤，最后得到稀疏的 B 样条曲面。为了提高计算的精确度，将每一个节点向量 Nodevector 分成 2000 份，这样可以较为精确地求出平面与 B 样条曲面的相交路径点（在画图时为了减少计算时间，只将节点向量均分为 100 份）。

在得到 B 样条曲面后，计算 Z=40 平面与 B 样条曲面的路径交点。计算交点的方法为筛选 z=40 附近的坐标点，然后根据筛选规则判断最合适的路径交点；得到路径交点坐标后，判断路径点的稠密程度，使用空间斜率来近似每一个坐标点的切线速度；直接求出 B 样条曲面的导数难度很大，所以本实验求出了每一个路径点对应的曲面最速下降梯度，然后根据梯度和切线方向求出每一点的曲面法向量；得到每一路径点的切线向量和法向量后，直接算出$b(s)$向量。

实验方法优点：算法简单，不用通过复杂的数学推导便可以得到较为精确的近似解。实验方法缺点：算法没有设置合理的运算区间，导致程序计算时间较长。

附录一

主函数:

```
clc;
clear;
%控制点矩阵
X = [0 0 0 0 0 0 0 0 0 0 0 ;
     20 20 20 20 20 20 20 20 20 20 20 ;
     40 40 40 40 40 40 40 40 40 40 40 ;
     60 60 60 60 60 60 60 60 60 60 60
     80 80 80 80 80 80 80 80 80 80 80];
Y = [-5 0 5 10 15 20 25 30 35 40 45;
      -5 0 5 10 15 20 25 30 35 40 45;
      -5 0 5 10 15 20 25 30 35 40 45;
      -5 0 5 10 15 20 25 30 35 40 45;
      -5 0 5 10 15 20 25 30 35 40 45];
  Z = [0 0 5 15 30 35 30 15 5    0 0;
      0 0 5 15 30 42 30 15 5 0 0;
      0 0 5 15 30 50 30 15 5 0 0;
      0 0 5 15 30 42 30 15 5 0 0;
      0 0 5 15 30 35 30 15 5 0 0];

[M, N] = size(X);
%  绘制控制点网
Surf_PlotCtrlMesh(M, N, X, Y, Z);
k = 2;
l = 2;
Surf_PlotSubMesh(M, N, k, l, X, Y, Z);

function Surf_PlotCtrlMesh(M, N, X, Y, Z)
%在u和v方向上绘制控制点线段
for i = 1 : M
    for j = 1 : N - 1
        hold on
        plot3([X(i, j) X(i, j+1)], [Y(i, j) Y(i, j+1)], [Z(i, j) Z(i, j+1)], 'o-k');
    end
end
for j = 1 : N
    for i = 1 : M-1
        plot3([X(i, j) X(i+1, j)], [Y(i, j) Y(i+1, j)], [Z(i, j) Z(i+1, j)], 'o-k');
    end
end
```

```
%另外一种绘制方法
%for i = 1: M
 %     for j = 1 : N
  %          hold on
   %         plot3(X(i,j), Y(i,j) ,Z(i,j),'o-k');
    %end
%end
xlabel('X');
ylabel('Y');
zlabel('Z');
view(100, -100);

function Nik_u = BaseFunction(i, k , u, NodeVector)
%计算节点向量

if k == 0            %定义0次B样条
      if (u >= NodeVector(i+1)) && (u < NodeVector(i+2))
            Nik_u = 1.0;
      else
            Nik_u = 0.0;
      end
else
      Length1 = NodeVector(i+k+1) - NodeVector(i+1);
      Length2 = NodeVector(i+k+2) - NodeVector(i+2);          % 定义区间长度
      if Length1 == 0.0            % 定义长度为0时的分母，防止出错
            Length1 = 1.0;
      end
      if Length2 == 0.0
            Length2 = 1.0;
      end
      Nik_u = (u - NodeVector(i+1)) / Length1 * BaseFunction(i, k-1, u, NodeVector) +
(NodeVector(i+k+2) - u) / Length2 * BaseFunction(i+1, k-1, u, NodeVector);
end

function Surf_PlotSubMesh(M, N, k, l, X, Y, Z)
%绘制B样条曲面
w = 1:100;
y = 1:100;
z = 40*ones(100);
% ----------------------------------------------------------------
a = zeros(360,1);
b = zeros(360,1);
c = zeros(360,1);
```

```
n = N - 1;
m = M - 1;
t = 1;
f = 1;
order = 1;

piece_u = 2000;     % 将u向节点向量矢量细分
piece_v = 2000;     % 将v向节点向量矢量细分
Nik_u = zeros(n+1, 1);   % 基函数初始化,生成(n+1)维列向量
Nik_v = zeros(m+1, 1);% 初始化，生成（m+1）维列向量
%u方向的细分
NodeVector_u = linspace(0, 1, n+k+2);         % 均匀B样条的u向节点矢量
u = linspace(k/(n+k+1), (n+1)/(n+k+1), piece_u);       %u向节点分成若干份

X_M_piece = zeros(M, piece_u);       %沿着u方向的网格，M * piece
Y_M_piece = zeros(M, piece_u);
Z_M_piece = zeros(M, piece_u);
for i = 1 : M
      for j = 1 : piece_u
            for ii = 0 : 1: n
                  Nik_u(ii+1, 1) = BaseFunction(ii, k , u(j), NodeVector_u);
            end
            X_M_piece(i, j) = X(i, :) * Nik_u;
            Y_M_piece(i, j) = Y(i, :) * Nik_u;
            Z_M_piece(i, j) = Z(i, :) * Nik_u;
      end
end

%v方向的细分
NodeVector_v = linspace(0, 1, m+l+2);         % 均匀B样条的u向节点矢量
v = linspace(l/(m+l+1), (m+1)/(m+l+1), piece_v);       %   v向节点分成若干份
X_MN_piece = zeros(piece_v, piece_u);
Y_MN_piece = zeros(piece_v, piece_u);
Z_MN_piece = zeros(piece_v, piece_u);
for i = 1 : piece_u
      for j = 1 : piece_v
            for ii = 0 : 1 : m
                  Nik_v(ii+1, 1) = BaseFunction(ii, l, v(j), NodeVector_v);
            end
      X_MN_piece(j, i) = Nik_v' * X_M_piece(:, i);
      Y_MN_piece(j, i) = Nik_v' * Y_M_piece(:, i);
      Z_MN_piece(j, i) = Nik_v' * Z_M_piece(:, i);
      end
```

```
end

%%
%筛选边界点
for i = 2:piece_u-1
    for j = 2:piece_v-1
        if((Z_MN_piece(i, j)>=40 && Z_MN_piece(i-1, j-1)<40 && Z_MN_piece(i-1, j)<40
&& Z_MN_piece(i, j-1)<40) ...
                || (Z_MN_piece(i, j)>=40 && Z_MN_piece(i-1, j+1)<40 && Z_MN_piece(i,
j+1)<40   && Z_MN_piece(i-1, j)<40) ...
                ||(Z_MN_piece(i, j)>=40 && Z_MN_piece(i+1, j)<40 && Z_MN_piece(i+1,
j-1)<40 && Z_MN_piece(i, j-1)<40) ...
            ||(Z_MN_piece(i, j)>=40 && Z_MN_piece(i+1, j+1)<40 && Z_MN_piece(i+1,
j)<40 && Z_MN_piece(i, j+1)<40))
        a(t)= X_MN_piece(i, j);
        b(t)= Y_MN_piece(i, j);
        c(t)= Z_MN_piece(i, j);
        if(X_MN_piece(i, j)<=40 && Y_MN_piece(i, j)>20)
            grade_x(t) = X_MN_piece(i-1, j+1) - X_MN_piece(i, j);
            grade_y(t) = Y_MN_piece(i-1, j+1) - Y_MN_piece(i, j);
            grade_z(t) = Z_MN_piece(i-1,j+1) - Z_MN_piece(i,j);
            if grade_z(t)>0
                grade_z(t) = -grade_z(t);
            end
        end
        if(X_MN_piece(i, j)>40 && Y_MN_piece(i, j)>20)
            grade_x(t) = X_MN_piece(i+1, j+1) - X_MN_piece(i, j);
            grade_y(t) = Y_MN_piece(i+1, j+1) - Y_MN_piece(i, j);
            grade_z(t) = Z_MN_piece(i+1,j+1) - Z_MN_piece(i,j);
             if grade_z(t)>0
                grade_z(t) = -grade_z(t);
            end
        end
        if(X_MN_piece(i, j)<=40 && Y_MN_piece(i, j)<=20)
            grade_x(t) = X_MN_piece(i-1, j-1) - X_MN_piece(i, j);
            grade_y(t) = Y_MN_piece(i-1, j-1) - Y_MN_piece(i, j);
            grade_z(t) = Z_MN_piece(i-1,j-1) - Z_MN_piece(i,j);
             if grade_z(t)>0
                grade_z(t) = -grade_z(t);
            end
        end
        if(X_MN_piece(i, j)>40 && Y_MN_piece(i, j)<=20)
            grade_x(t) = X_MN_piece(i+1, j-1) - X_MN_piece(i, j);
```

```
                grade_y(t) = Y_MN_piece(i+1, j-1) - Y_MN_piece(i, j);
                grade_z(t) = Z_MN_piece(i+1,j-1) - Z_MN_piece(i,j);
                 if grade_z(t)>0
                     grade_z(t) = -grade_z(t);
                end
            end
            t = t + 1;
            end
        end
end

%point
for i = 1: 360
    point(i,:) = [a(i) b(i) c(i) grade_x(i) grade_y(i) grade_z(i)];
end


%making the trajectory
for i = 1:360
    if(point(i,1) <= 40 && point(i,2) <=20)
        point_right_1(order,:) = [point(i,1) point(i,2) point(i,3) point(i,4) point(i,5) point(i,6)];
        order = order + 1;
    end
end

order = 1;

for i = 1:360
    if(point(i,1) > 40 && point(i,2) <=20)
         point_right_2(order,:) = [point(i,1) point(i,2) point(i,3) point(i,4) point(i,5) point(i,6)];
         order = order + 1;
    end
end

order = 1;

for i = 1:360
    if(point(i,1) > 40 && point(i,2) > 20)
         point_3(order,:) = [point(i,1) point(i,2) point(i,3) point(i,4) point(i,5) point(i,6)];
         order = order + 1;
    end
end
```

```
order = 1;

for i = 1:360
    if(point(i,1) <= 40 && point(i,2) > 20)
        point_4(order,:) = [point(i,1) point(i,2) point(i,3) point(i,4) point(i,5) point(i,6)];
        order = order + 1;
    end
end

[point_3_size,~]=size(point_3);
[point_4_size,~]=size(point_4);

for i =1: point_3_size
    point_right_3(i,:) = point_3(end-i+1,:);
end

for i =1: point_4_size
    point_right_4(i,:) = point_4(end-i+1,:);
end

%%
%计算结果
%坐标点
points = [point_right_1;point_right_2;point_right_3;point_right_4];

%梯度
gradient = [points(:,4) points(:,5) points(:,6)];

%速度
speed(1,:) = [points(1,1)-points(360,1) points(1,2)-points(360,2) 0];
for i = 2:360
    speed(i,:)=[points(i,1)-points(i-1,1) points(i,2)-points(i-1,2) 0];
end

%法向量
for i =1:360
    normal_vector(i,:)=cross(gradient(i,:),speed(i,:));
end


%b(s)向量
for i = 1:360
    b_vector(i,:) = cross(speed(i,:),normal_vector(i,:));
```

```
end

%单位化
for i = 1:360
    speed_norm(i,:)=speed(i,:)/norm(speed(i,:));
end

for i = 1:360
    gradient_norm(i,:)=gradient(i,:)/norm(gradient(i,:));
end

for i = 1:360
    normal_vector_norm(i,:)=normal_vector(i,:)/norm(normal_vector(i,:));
end

for i = 1:360
    b_vector_norm(i,:)= b_vector(i,:)/norm(b_vector(i,:));
end

%%
%%绘制曲面图
  for j = 1 : piece_u
      for i = 1 : piece_v -1
          hold on
          plot3([X_MN_piece(i, j) X_MN_piece(i+1, j)],...
              [Y_MN_piece(i, j) Y_MN_piece(i+1, j)],...
              [Z_MN_piece(i, j) Z_MN_piece(i+1, j)], 'b-');
      end
  end
 for i = 1 : piece_v
      for j = 1 : piece_u -1
          hold on
          plot3([X_MN_piece(i, j) X_MN_piece(i, j+1)],...
              [Y_MN_piece(i, j) Y_MN_piece(i, j+1)],...
              [Z_MN_piece(i, j) Z_MN_piece(i, j+1)], 'b-');
      end
end
hold on
mesh(w,y,z);
hold on
plot3(a,b,c,'o')
```

```
%Answer of Question 1
disp([points(:,1) points(:,2) points(:,3) normal_vector_norm]);

%Answer of Question 2
disp([points(:,1) points(:,2) points(:,3) speed_norm normal_vector_norm b_vector_norm]);

%%
% 输出文档
fid_1 = fopen('Answer of Question_1.txt','w');
for i=1:360
fprintf(fid_1,'%f\t%f\t%f\t%f\t%f\t%f\n',points(i,1),points(i,2),points(i,3),normal_vector_norm(i,1
),normal_vector_norm(i,2),normal_vector_norm(i,3));
end
fclose(fid_1);
fid_2 = fopen('Answer of Question_2.txt','w');
for i=1:360
fprintf(fid_2,'%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n',points(i,1),points(i,2),points(i,3
),speed_norm(i,1),speed_norm(i,2),speed_norm(i,3),...
normal_vector_norm(i,1),normal_vector_norm(i,2),normal_vector_norm(i,3),b_vector_norm(i,1),
b_vector_norm(i,2),b_vector_norm(i,3));
end
fclose(fid_2);
```

附录二

第一题答案（部分）：$[X\ Y\ Z\ n_x\ n_y\ n_z]$

23.206603	19.921211	40.001934	-0.338012	0.000000	0.941142
23.236618	19.853677	40.002435	-0.293414	-0.130406	0.947049
23.266633	19.808654	40.002924	-0.281712	-0.187808	0.940939
23.296648	19.763632	40.001231	-0.301427	-0.200951	0.932073
23.326663	19.741121	40.003864	-0.232031	-0.309374	0.922198
23.356678	19.718609	40.005945	-0.239085	-0.318780	0.917179
23.386693	19.696098	40.007473	-0.246030	-0.328040	0.912063
23.416708	19.651076	40.000285	-0.348434	-0.232289	0.908094
23.446723	19.628564	40.000151	-0.266078	-0.354771	0.896292
23.506753	19.606053	40.008025	-0.176447	-0.470526	0.864564
23.536768	19.583542	40.006766	-0.279072	-0.372096	0.885248
23.566783	19.561031	40.004946	-0.285341	-0.380455	0.879679
23.596798	19.538519	40.002563	-0.291497	-0.388663	0.874054
23.656828	19.516008	40.008154	-0.191551	-0.510804	0.838086
23.686843	19.493497	40.004631	-0.303603	-0.404804	0.862531
23.716858	19.470985	40.000542	-0.309423	-0.412564	0.856766
23.776888	19.448474	40.004396	-0.202038	-0.538767	0.817870
23.836918	19.425963	40.007661	-0.205428	-0.547809	0.810990
23.866933	19.403452	40.001839	-0.326497	-0.435329	0.838981
23.926963	19.380940	40.003927	-0.211896	-0.565057	0.797377
23.986993	19.358429	40.005419	-0.215058	-0.573489	0.790481
24.047024	19.335918	40.006310	-0.218146	-0.581723	0.783588
24.107054	19.313407	40.006599	-0.221161	-0.589763	0.776703
24.167084	19.290895	40.006283	-0.224104	-0.597611	0.769830
24.227114	19.268384	40.005358	-0.226976	-0.605270	0.762974
24.287144	19.245873	40.003822	-0.229779	-0.612745	0.756139
24.347174	19.223362	40.001672	-0.232514	-0.620037	0.749327
24.437219	19.200850	40.007300	-0.169762	-0.679049	0.714194
24.497249	19.178339	40.003900	-0.237868	-0.634315	0.735570
24.587294	19.155828	40.008245	-0.173242	-0.692970	0.699843
24.647324	19.133317	40.003582	-0.242963	-0.647903	0.721936

第二题答案（部分）：$[X\ Y\ Z\ t_x\ t_y\ t_z\ n_x\ n_y\ n_z\ b_x\ b_y\ b_z]$

23.206603	19.921211	40.001934	0.000000	-1.000000	0.000000	-0.338012	0.000000	0.941142	-0.941142	-0.000000	-0.338012
23.236618	19.853677	40.002435	0.406138	-0.913812	0.000000	-0.293414	-0.130406	0.947049	-0.865425	-0.384633	-0.321088
23.266633	19.808654	40.002924	0.554700	-0.832050	0.000000	-0.281712	-0.187808	0.940939	-0.782909	-0.521939	-0.338576
23.296648	19.763632	40.001231	0.554700	-0.832050	0.000000	-0.301427	-0.200951	0.932073	-0.775532	-0.517021	-0.362270
23.326663	19.741121	40.003864	0.800000	-0.600000	0.000000	-0.232031	-0.309374	0.922198	-0.553319	-0.737758	-0.386718
23.356678	19.718609	40.005945	0.800000	-0.600000	0.000000	-0.239085	-0.318780	0.917179	-0.550307	-0.733743	-0.398475
23.386693	19.696098	40.007473	0.800000	-0.600000	0.000000	-0.246030	-0.328040	0.912063	-0.547238	-0.729651	-0.410049
23.416708	19.651076	40.000285	0.554700	-0.832050	0.000000	-0.348434	-0.232289	0.908094	-0.755580	-0.503720	-0.418766
23.446723	19.628564	40.000151	0.800000	-0.600000	0.000000	-0.266078	-0.354771	0.896292	-0.537775	-0.717034	-0.443463
23.506753	19.606053	40.008025	0.936329	-0.351123	0.000000	-0.176447	-0.470526	0.864564	-0.303569	-0.809517	-0.502522
23.536768	19.583542	40.006766	0.800000	-0.600000	0.000000	-0.279072	-0.372096	0.885248	-0.531149	-0.708198	-0.465120
23.566783	19.561031	40.004946	0.800000	-0.600000	0.000000	-0.285341	-0.380455	0.879679	-0.527807	-0.703743	-0.475569
23.596798	19.538519	40.002563	0.800000	-0.600000	0.000000	-0.291497	-0.388663	0.874054	-0.524432	-0.699243	-0.485829
23.656828	19.516008	40.008154	0.936329	-0.351123	0.000000	-0.191551	-0.510804	0.838086	-0.294272	-0.784724	-0.545538
23.686843	19.493497	40.004631	0.800000	-0.600000	0.000000	-0.303603	-0.404804	0.862531	-0.517518	-0.690025	-0.506005
23.716858	19.470985	40.000542	0.800000	-0.600000	0.000000	-0.309423	-0.412564	0.856766	-0.514060	-0.685413	-0.515705
23.776888	19.448474	40.004396	0.936329	-0.351123	0.000000	-0.202038	-0.538767	0.817870	-0.287173	-0.765795	-0.575404
23.836918	19.425963	40.007661	0.936329	-0.351123	0.000000	-0.205428	-0.547809	0.810990	-0.284758	-0.759353	-0.585060
23.866933	19.403452	40.001839	0.800000	-0.600000	0.000000	-0.326497	-0.435329	0.838981	-0.503388	-0.671185	-0.544161
23.926963	19.380940	40.003927	0.936329	-0.351123	0.000000	-0.211896	-0.565057	0.797377	-0.279978	-0.746607	-0.603481
23.986993	19.358429	40.005419	0.936329	-0.351123	0.000000	-0.215058	-0.573489	0.790481	-0.277556	-0.740151	-0.612486
24.047024	19.335918	40.006310	0.936329	-0.351123	0.000000	-0.218146	-0.581723	0.783588	-0.275136	-0.733697	-0.621280
24.107054	19.313407	40.006599	0.936329	-0.351123	0.000000	-0.221161	-0.589763	0.776703	-0.272719	-0.727250	-0.629867
24.167084	19.290895	40.006283	0.936329	-0.351123	0.000000	-0.224104	-0.597611	0.769830	-0.270306	-0.720815	-0.638248
24.227114	19.268384	40.005358	0.936329	-0.351123	0.000000	-0.226976	-0.605270	0.762974	-0.267898	-0.714395	-0.646429
24.287144	19.245873	40.003822	0.936329	-0.351123	0.000000	-0.229779	-0.612745	0.756139	-0.265498	-0.707995	-0.654412
24.347174	19.223362	40.001672	0.936329	-0.351123	0.000000	-0.232514	-0.620037	0.749327	-0.263106	-0.701617	-0.662200
24.437219	19.200850	40.007300	0.970143	-0.242536	0.000000	-0.169762	-0.679049	0.714194	-0.173217	-0.692870	-0.699948
24.497249	19.178339	40.003900	0.936329	-0.351123	0.000000	-0.237868	-0.634315	0.735570	-0.258276	-0.688736	-0.677448
24.587294	19.155828	40.008245	0.970143	-0.242536	0.000000	-0.173242	-0.692970	0.699843	-0.169737	-0.678948	-0.714297
24.647324	19.133317	40.003582	0.936329	-0.351123	0.000000	-0.242963	-0.647903	0.721936	-0.253489	-0.675969	-0.691960
24.737369	19.110805	40.006629	0.970143	-0.242536	0.000000	-0.176537	-0.706149	0.685703	-0.166307	-0.665230	-0.727881
24.797399	19.088294	40.000688	0.936329	-0.351123	0.000000	-0.247810	-0.660828	0.708447	-0.248752	-0.663339	-0.705764