图形学实验 PA2：参数曲线和曲面

指导教师：胡事民　助教：曹耕晨、彭浩洋

2023 年 4 月 4 日

1　实验综述

本次作业中，你将使用课上所学的参数曲线知识，基于 OpenGL 框架绘制 Bezier 曲线、B 样条以及由他们绕固定轴旋转所产生的旋转体曲面。我们的测例将向你提供参数曲线的控制点个数和坐标，相关的测例总共有 3 个。

2　细节说明

2.1　参数曲线和导数

三维图形学中的参数曲线（Parametrized Curve）指的是三维欧式空间中的一条轨迹，是给定参数 t 在某个区间 I（例如 $[0,1]$）内变动时所描绘出的一条曲线。曲线上每个点的坐标为 $\boldsymbol{f}(t) = [x(t), y(t), z(t)]^\top$，其中 $x(\cdot), y(\cdot), z(\cdot)$ 都是由 I 映射到 $\mathbb{R}$ 的实值函数。通过定义不同形式的 $\boldsymbol{f}(t)$，就能绘制出不同形状的参数曲线。由于计算机需要对图形进行离散化表示，在实际绘制的时候，我们经常在 I 中等距地采样足够多的点 $t_0, t_1, \ldots, t_L$，并依次计算 $\boldsymbol{f}(t_0), \boldsymbol{f}(t_1), \ldots, \boldsymbol{f}(t_L)$，再把这些点用直线从头到尾连接起来即可。显然采样点数 L 越大，绘制出的曲线就越精细。

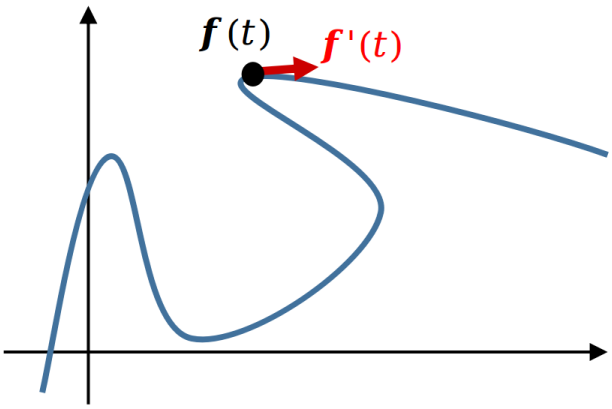


图 1: 参数曲线示例

在许多应用中，仅仅知道 $\boldsymbol{f}(t)$ 是不够的，有时我们还需要知道参数曲线的导数 $\boldsymbol{f}'(t) = [x'(t), y'(t), z'(t)]^\top$。导数（Derivative/Jacobian）代表了一个函数在某点的切线方向。设想我们在制作游戏动画的时候，需要让一架飞机沿着参数曲线进行飞行，简单的平移是不科学的，飞机头的朝向必须要和参数曲线切线方向一致才足够真实。

2.2 样条曲线和基函数

样条曲线（Spline Curve）指的是分段定义的多项式参数曲线，由于其构造简便，使用方便，能够准确地拟合复杂形状，在计算机辅助设计领域非常受欢迎，是目前的几何软件中必备功能之一。三维空间中一般的样条曲线定义如下：

$$\boldsymbol{f}(t) = \sum_{i=0}^{n} B_{i,k}(t)\boldsymbol{P}_i \tag{1}$$

其中 $B_{i,k}(t)$ 是一维实数值多项式函数，常常被称为基函数（Basis Function）。k 是基函数多项式的次数（degree，也译作度数）[1]，也被称为样条曲线的次数，次数决定了样条函数的连续性，是描述样条的一个重要性质。$\boldsymbol{P}_i$ 是一个个三维点，又被称为控制点。当基函数固定的时候，改变控制点的位置，样条曲线的形状就能够发生改变，而曲线具体每一部分受每个控制点的影响大小则取决于基函数具体的数值。

Bezier 曲线（贝塞尔曲线）是一种常用的样条曲线，对于一个有 $n+1$ 个控制点的 Bezier 曲线而言，次数 $k=n$，其基函数定义如下：

$$B_{i,k}(t) = B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad t \in [0,1], \tag{2}$$

其中 $\binom{n}{i}$ 是多项式函数[2]

对 Bezier 曲线的基函数求导可以得到：

$$B_{i,n}^{'}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t)). \tag{3}$$

B 样条曲线是对 Bezier 的一个推广，它的基函数形式更为一般，由 de Boor 递推公式实现（参见下节）。B 样条不仅可以任意选择次数 k 和控制点个数 $n+1$，还提供了一组节点（Knots）可供用户更加细致地控制曲线形状。由于其良好的性质，在数学界也常被用于差值或拟合计算。

2.3 de Boor-Cox 算法

一个控制点个数为 $n+1$，次数为 k 的 B 样条的节点（Knots）个数为 $n+k+2$，定义这些节点组成的序列（又称作 Knot Vector）[3]为：

$$0 \le t_0 \le t_1 \le \cdots \le t_{n+k+1} \le 1. \tag{4}$$

由 de Boor 递推公式定义的 B 样条基函数形式为：

$$\begin{aligned} B_{i,0}(t) &= \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}, \quad 0 \le i < n+k+1 \\ B_{i,p}(t) &= \frac{t-t_i}{t_{i+p}-t_i}B_{i,p-1}(t) + \frac{t_{i+p+1}-t}{t_{i+p+1}-t_{i+1}}B_{i+1,p-1}(t), \quad 1 \le p \le k \end{aligned} \tag{5}$$

由于最终我们要求出基函数 $B_{i,k}(t)$，由第二个递推式可以看出它由两个低一次幂的基函数 $B_{i,k-1}$ 和 $B_{i+1,k-1}$ 的加权和组成，再分别计算这两个 $k-1$ 次多项式函数，会发现他们分

[1] 课件中将字母 k 定义为阶数（order），阶数 = 次数 +1，我发现许多参考资料（包括维基百科）都喜欢将 k 定义为次数，这样的好处是 Bezier 和 B 样条的基函数表达具有统一性，所以本教程中也使用这种定义方式。

[2] 即组合数，部分地区高中教材写作 C_n^i.

[3] 这里的公式为了和大部分网络百科与文献统一，也使用 t_i 符号，请注意区分2.1节中的计算机绘制所需的采样点符号。

别又由两个 $k-2$ 次多项式函数组成。以此类推，能够溯源到 0 次平凡多项式。整套程序可以简单使用递归完成，只要将上述边界条件和递推公式原封不动地抄上去就可以了。但是，为了避免中间阶数的基函数被重复计算太多次，拖慢程序效率，可以使用正向推导（动态规划）的写法，或者使用备忘录式的递归记录固定参数下的函数执行结果。

大家可以验证 de Boor 递推公式的如下性质，具体证明留作自我练习：

- 当 $t \in [t_k, t_{n+1}]$ 时，$\sum_{i=0}^{n} B_{i,k}(t) = 1$，其余区间不成立，这说明 B 样条曲线的合理参数范围 $I = [t_k, t_{n+1}]$。当 $t_k \neq 0, t_{n+1} \neq 1$ 时，样条曲线的首尾和控制点不相接。
- 当 $n = k, t_0 = \cdots = t_n = 0, t_{n+1} = \cdots = t_{2n+1} = 1$ 时，$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i$，即这时的 B 样条实际上是一个次数为 n，控制点个数为 $n+1$ 的 Bezier 曲线。进一步说明 B 样条是 Bezier 曲线的推广。
- 当 $t_i \leq t < t_{i+1}$ 时，$B_{u,k}(t) = 0, u \in [0, i-k-1] \cup [i+1, n+1]$，这说明参数曲线中的一段 $[t_i, t_{i+1}]$ 实际上只受 $k+1$ 个控制点 $\boldsymbol{P}_{i-k}, \ldots, \boldsymbol{P}_i$ 的控制，意味着一个带有 $n+1$ 的控制点的 k 次 B 样条可以分解成 $n+1-k$ 个 k 次 B 样条平滑拼接的结果[4]。

B 样条的导数同样可以使用 de Boor 递推公式推导出来，形式如下：

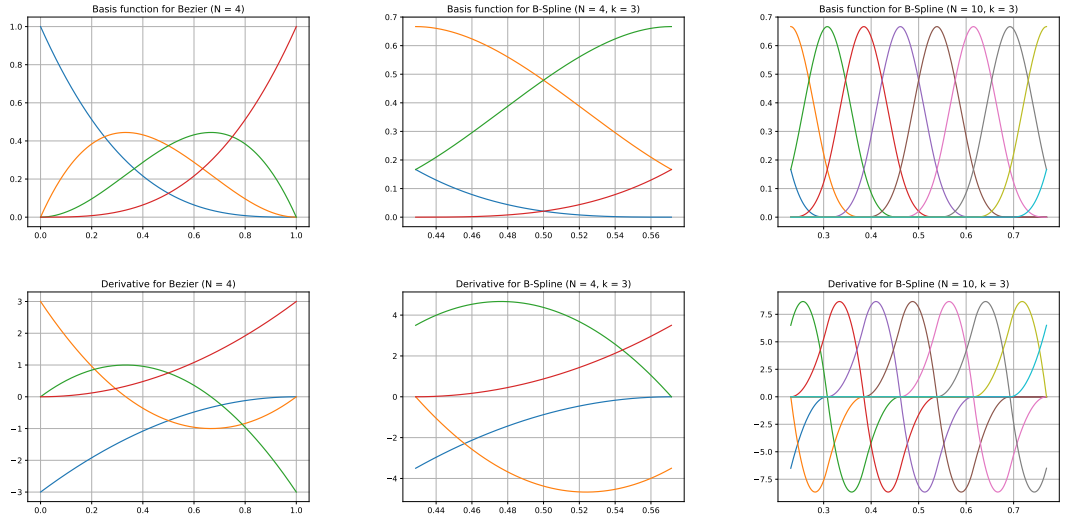
$$B'_{i,p}(t) = p \cdot \left(\frac{B_{i,p-1}(t)}{t_{i+p} - t_i} - \frac{B_{i+1,p-1}(t)}{t_{i+p+1} - t_{i+1}}\right) \tag{6}$$

在本次作业中，为了统一大家的实验结果，我们要求 B 样条的节点 Knot Vector 采用如下定义：

$$t_i = \frac{i}{n+k+1}, \quad 0 \leq i \leq n+k+1, \tag{7}$$

同时我们约定所有例子里 $k = 3$.

为了帮助大家更好地理解基函数的计算过程以及其最终计算结果得到的形状，我们为大家提供了一个 Python 脚本用于绘制基函数及其导数，一些结果如下所示：



脚本中的编码和上述数学描述中有一个不同的地方，那就是使用了变量 `N/n` 来表示控制点的个数，而我们的叙述中控制点的个数是 $n+1$，因此在对照程序实现和公式时需要有一个转换。

[4]使用过 Photoshop 抠图的同学都熟悉“钢笔工具”，该工具绘制的就是分段连续的三次 Bezier 曲线，你也可以把它看成是一个有很多控制点的三次 B 样条。

如果你想了解更多有关 Bezier 曲线和 B 样条的内容，推荐你仔细研读课件 PPT 以及维基百科上的相关资料。

2.4 旋转曲面

一般的参数曲面是由两个参数定义的函数 $\boldsymbol{S}(\cdot,\cdot):\mathbb{R}^2\mapsto\mathbb{R}^3$，当这两个参数在可行范围内取值时，对应的函数值 $\boldsymbol{S}$ 能张成一个曲面。例如：

$$\boldsymbol{S}(\theta,\phi)=[\sin\theta\cos\phi,\sin\theta\sin\phi,\cos\theta]^\top,\quad \theta\in[0,\pi),\phi\in[0,2\pi) \tag{8}$$

代表的是三维空间中一个中心在原点，半径为 1 的球面。

本次作业中出现的旋转曲面（Surface of Revolution），定义如下：

$$\boldsymbol{S}(t,\theta)=[\boldsymbol{f}(t)_x\cos\theta,\boldsymbol{f}(t)_y,\boldsymbol{f}(t)_x\sin\theta]^\top,\quad t\in I,\theta\in[0,2\pi) \tag{9}$$

其中 $\boldsymbol{f}(t)$ 是2.2节定义的样条曲线，这里我们规定所有控制点 $\boldsymbol{P}_i$ 的 z 坐标均为 0，即样条曲线只存在 x-y 平面上。$\boldsymbol{f}(t)_x$ 和 $\boldsymbol{f}(t)_y$ 分别代表 $\boldsymbol{f}(t)$ 的 x 坐标和 y 坐标。整体最终形成的旋转曲面实际上是样条曲线 $\boldsymbol{f}(t)$ 以 y 轴为中心，旋转 360° 扫掠出的光滑曲面。

3 框架代码说明

3.1 环境配置与编译

我们非常推荐使用带有 `CMake` 套件的 `Ubuntu` 系统进行编程，`Windows 10` 下可以使用 Ubuntu Subsystem。我们的框架代码依赖于 `GL` 和 `GLUT`，前者一般在安装显卡驱动的时候会自动配置好，如果 `CMake` 配置出错可以尝试使用 `apt` 安装 `mesa-common-dev`；后者请使用 `apt` 安装 `freeglut3-dev`。安装完成之后，请在包含有 `run_all.sh` 的文件夹下打开终端，并执行：

```
bash ./run_all.sh
```

这段脚本会自动设置编译，并在提供的 6 个测例上运行你的程序。你的程序最终会被编译到 `bin/`文件夹中。程序使用 OpenGL 进行渲染，交互操作、常见问题等更多事项请参考学习材料：OpenGL 绘制。

3.2 样例 Python 脚本

我们在 `ref` 文件夹中提供了一个 Python 脚本程序用于参照学习，当然你也可以把它翻译成 C++ 实现在你的作业里面。该脚本使用 Python3 解释器就可以运行，它默认会调用 `matplotlib` 绘图库绘制三次 Bezier 曲线的 4 个基函数 $B_{0,3}(t),B_{1,3}(t),B_{2,3}(t),B_{3,3}(t)$ 以及其导数，程序可选参数如下：

- `--N <count>`: 样条曲线的控制点个数 $n+1$；
- `--k <order>`: 样条曲线的次数 k；
- `--bspline`: 绘制 B 样条，如果不加该选项默认绘制 Bezier 曲线；
- `--valid`: 仅绘制 B 样条的有效参数 $I=[t_k,t_{n+1}]$ 部分，如果不加该选项默认绘制 $t\in[0,1]$ 参数区间。

4 测试用例

为了测试代码是否正确无误，我们总共提供了 6 个测试用例（如图2-7所示），你也可以根据场景的文件格式构造样例进行自我测试。

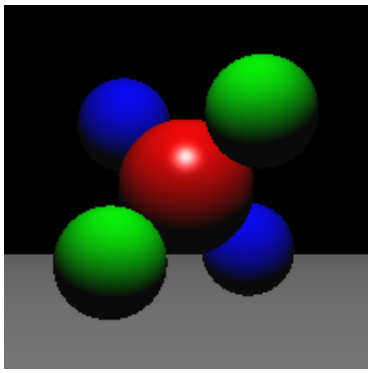


图 2: 五个球体

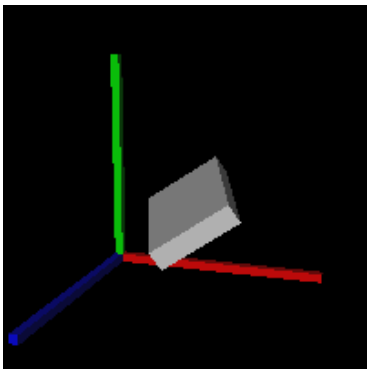


图 3: 坐标系

图 4: 复杂兔子

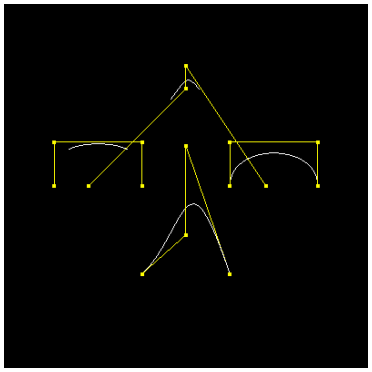


图 5: 参数曲线

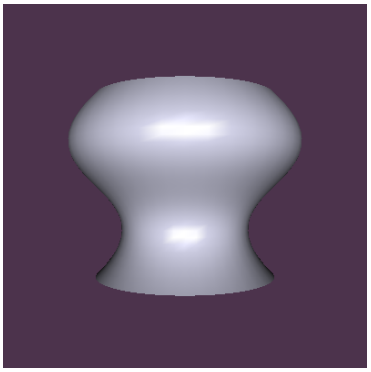


图 6: 花瓶-简单旋转面

图 7: 碰杯-复杂旋转面

5 作业要求

首先，你需要将 PA1 的代码合并过来，确保前三个测例能够正常运行输出结果。

本次作业所有需要新填写的地方都被标记了 `TODO`，请全文查找该字样，每实现完成一个功能，你就可以去掉一个 TODO。你需要实现的主要的函数是 `curve.hpp` 中 BezierCurve 和 BsplineCurve 类的 `discretize(int resolution, std::vector<CurvePoint>& data)` 函数，该函数要求使用 `Curve` 类中提供的成员变量：控制点数组 `controls`，在 `data` 中填入曲线离散化（采样）表示之后的每一个点坐标，参数 `resolution` 指定了离散化的分辨率，分辨率越高曲线显得越光滑，但相应计算时间也越长。我们推荐将分辨率理解成每一个小段 $[t_i, t_{i+1})$ 中等距采样的点的个数。除了点之外，还需要填入相应位置的切向，该方向将被用作绘制旋转曲面时计算面片的法向。

具体旋转曲面的实现在 `revsurface.hpp` 中，已经为你写好，理论上只要正确实现了 `Curve::discretize` 函数，程序就能够自动绘制出旋转曲面。当然，我们也鼓励你阅读旋转曲面相关绘制代码，这对大作业相关部分的实现有帮助。

在确认执行 `bash ./run_all.sh` 可以输出正确的图片后，请将你的代码放入 `code` 文件

夹（请包含且只包含下面文件树中的目录），和一个 `REPORT.pdf` 文档一起打包成 `zip` 文件提交至网络学堂。具体文件树结构如下所示，不遵循该结构的提交会被扣分：

```
姓名-学号-PA2.zip
├─ REPORT.pdf
└─ code
   ├─ run_all.sh
   ├─ CMakeLists.txt
   ├─ src/
   ├─ include/
   └─ deps/
```

报告 `REPORT.pdf` 中应包含且只包含以下几个部分：

1. 曲线性质：Bezier 曲线和 B 样条曲线有什么异同？怎样绘制一个首尾相接且接点处也有连续性质的 B 样条？
2. 代码逻辑：阅读 `revsurface.hpp` 中绘制旋转曲面的代码，简述其主要绘制逻辑。
3. 代码参考：完成作业的时候和哪些同学进行了怎样的讨论？是否借鉴了网上/别的同学的代码？
4. （可选）问题：你在实现过程中遇到了哪些问题？
5. （可选）未解决的困难：你的代码有哪些未解决的 bug？如果给你更多时间来完成作业，你将会怎样进行调试？
6. （可选）建议：你对本次作业有什么建议？文档或代码中有哪些需要我们改进的地方？

本次作业的 Deadline 以网络学堂为准。迟交的同学将得到一定的惩罚：晚交 3 天内分数将降低为 80%，3 天以上 1 周以内分数降为 50%，迟交一周以上的同学分数为 0。

6 致谢

本实验部分测例借鉴于MIT Open Courseware，在此表示感谢。按照其发布协议，这些测例原则上允许同学们以CC BY-NC-SA 4.0协议共享引用。