第六章　曲线

我们学习了直线或圆等简单图形的绘制方法，但是这对于更一般的复杂图形的表示来说仍然不够.

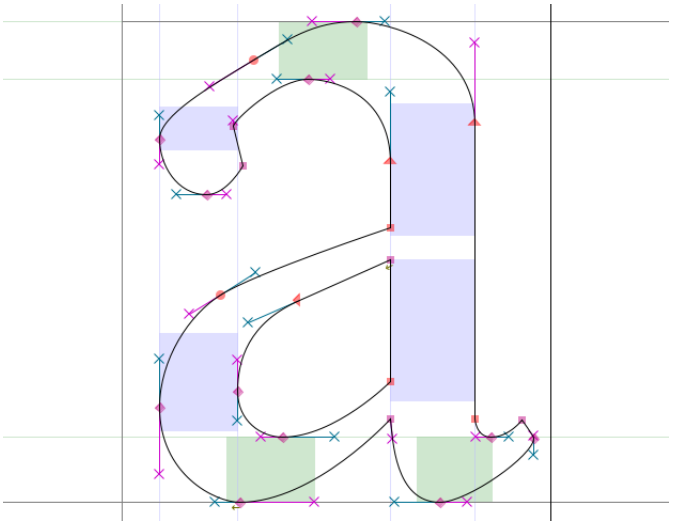


图 6.1: 利用DesignWithFontforge生成的字母'a'，其轮廓由直线段和三阶贝塞尔曲线段共同构成.

一个常见的情形是文字的显示，如图 (6.1) 中的字体'a'，该图形轮廓中除了直线，还包含了多段曲线. 理论上说，我们可以用多段直线来近似曲线，但这种近似将会消耗较多的计算和存储资源才能达到相对光滑的视觉效果，并且在进行局部放大时仍然显示出走样. 因此，为了能够更加准确、高效地绘制这些图形，我们需要一些方式来直接表示和渲染光滑曲线.

一般而言，我们可以把曲线或者曲面的表示方式可以分为显式表示和隐式表示两种.

显式表示通过给出曲线/曲面上的一系列点的坐标的值或函数表达式来表征曲线，较为直接和简单. 最简单的显示表达是点集，如二维空间中用点列表达曲线或三维空间中用点云表达曲面. 函数方程也是一种常用的显式表达的手段，如函数 $y = \sqrt{1 - x^2}$ 定义了一个二维平面内的上半圆弧. 更一般地，我们通过参数化的方式来定义曲线，一条二维平面内的**参数曲线** *(parametric curve)* 可以表达为 $f : [a, b] \to \mathbb{R}^2$，它用一个作为自变量的参数来定义一组坐标，例如，以原点为圆心、半径为 r 的圆的参数方程为：

$$\begin{cases} x(t) = r\cos(t), \\ y(t) = r\sin(t), \end{cases}$$

其中 $t \in [0, 2\pi)$，也被称为参数坐标. 参数曲线的优点在于可以容易地得到曲线上点的坐

标，应该也可以容易地可视化（只需要遍历参数坐标值就可以画出曲线的空间轨迹）．另外，将参数方程对参数坐标求导就可以得到曲线的梯度/切向方向．

隐式表示则一般给出曲线/曲面上的点所满足的性质，隐式函数方程是其中最常用的一类，例如，在二维空间中，所有满足方程 $x^2+y^2=1$ 的点 (x,y) 构成的集合就对应二维平面上的单位圆．这种表达方式的一般形式可以写为 $f(x,y)=0$．它的优点是容易对曲线的内部/外部或左侧/右侧进行区分：若存在点 (x_0,y_0)，只需将其代入表达式 $f(x_0,y_0)$ 并考察其正负即可．而缺点在于难以采样，若要从曲线上得到某点的具体坐标需要通过解方程．

不同的曲线表示方法各有优劣，需要根据具体应用来进行选择．本章中我们着重关注曲线的显式表示，隐式表示会在后文几何部分予以介绍．我们将首先介绍常见的几种曲线的数学表达，然后介绍如何进行曲线的光栅化．

6.1　从一阶曲线到高阶曲线

直线/线段是一类拥有最简单的数学形式的特殊曲线——一阶曲线，它可以通过在给定的两点间进行线性插值得到，在数学上对应着关于参数坐标 $t\in[0,1]$ 的一阶函数．

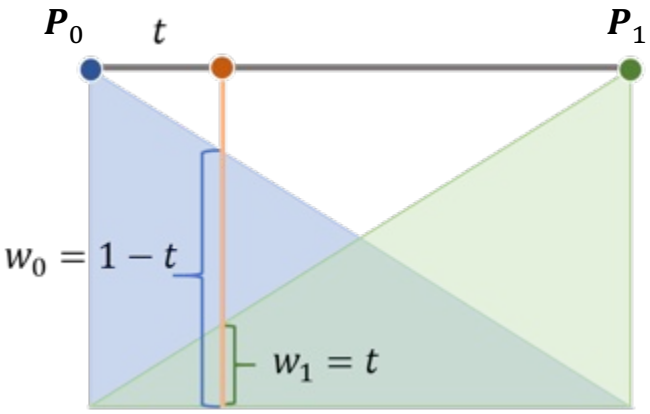


图 6.2: 对 $\mathbf{P}_0$ 和 $\mathbf{P}_1$ 进行线性插值得到一阶曲线．图中蓝色、绿色部分的高度分别表示权重函数 w_0, w_1 的大小．

图6.2展示了给定两点 $\mathbf{P}_0$, $\mathbf{P}_1$ 后通过插值获得一阶曲线的过程．这条曲线上的任意一点的坐标可以看成是参数 t 的函数

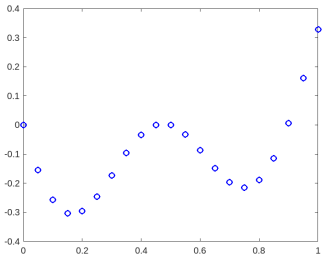
$$\mathbf{P}(t)=(1-t)\mathbf{P}_0+t\mathbf{P}_1,\quad t\in[0,1]. \tag{6.1}$$

其中，插值采用的一阶权重函数 $w_0(t)=1-t$, $w_1(t)=t$ 是关于 t 的线性函数，因此这个操作也被称为线性插值，记为 $\mathrm{lerp}(\mathbf{P}_0,\mathbf{P}_1,t)=\mathbf{P}(t)$．

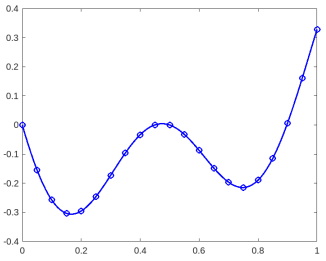
一阶曲线的优势在于数学形式简单，但它的缺陷是不够视觉上不够“光滑”．如何定义光滑？在图形学中，曲线的“光滑”通常意味着：在曲线上任意一点，其两侧曲线的一阶梯度共线且同向．我们称满足该性质的曲线是 G^1 **光滑**的．一个更严格的要求是：在曲线上任意一点，其两侧曲线的一阶梯度相等．我们称满足该性质的曲线是 C^1 **光滑**的．事实上，后者相比于前者多出的约束是两侧梯度大小相等，而一般来说只要满足了前者，我们就可以认为曲线在视觉效果上是“光滑”的．由线段连接而成的曲线是分段光滑的，但在连接点处，由于两侧线段不共线，因此它的左侧梯度和右侧梯度不共线，无法满足 G^1 光滑的条件．这是一阶插值的固有问题，是无法通过后续处理方法消除的．不光滑的后果不仅体现在视觉

上，也体现在数学量的离散表达上，如：如何定义曲线的曲率？如何定义曲线的弯曲势能？这些问题都会给下游的图形学应用带来困扰．

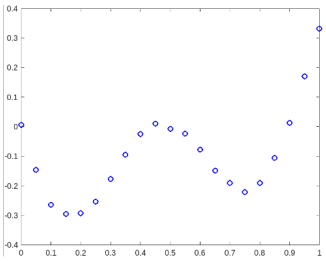
为此我们需要求助于高阶的曲线表示方法．一个最简单的高阶插值方法是拉格朗日插值 *(Lagrange interpolation)*，它利用高阶多项式对给定的点列进行平滑的连接．但这种方法在图形学中的应用并不广泛，因为随着点列数量的增加，拉格朗日插值的多项式阶数随之线性增长．另外，拉格朗日插值拟合出的曲线在边界附近波动较大，微小噪声就有可能造成曲线图像的大幅度改变，如图6.3所示，曲线两端出现剧烈的抖动，这被称为龙格现象 *(Runge's phenomenon)*．这种数值不稳定性导致拉格朗日插值很少被应用于图形学中．



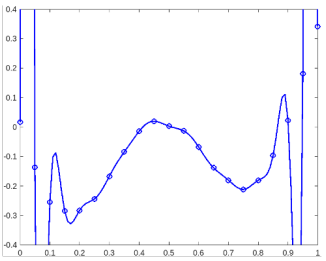
(a) 无噪声采样．



(b) 对图 (a) 采样点的拉格朗日插值拟合．



(c) 受噪声干扰的采样．



(d) 对图 (a) 采样点的拉格朗日插值拟合．

图 6.3: 从光滑曲线上采样出 21 个点，利用拉格朗日插值对采样出的点列进行拟合，拟合结果易出现龙格现象．

6.2　贝塞尔曲线

贝塞尔曲线 (Bézier curve) 是图形学、工程学、设计学等领域最常用的一类高阶曲线．它通过在端点之间额外引入一系列点来达成对曲线形状的控制，这些点 (包括端点) 被统称为控制点．通过对控制点进行调整，我们可以轻松地控制贝塞尔曲线的形状和高阶特征．

6.2.1　数学形式

我们先来看如何构造出一条贝塞尔曲线．如图6.4，一条二阶贝塞尔曲线是通过两轮线性插值得到的：

1. 第一轮，由参数 t 对 $\overrightarrow{\mathbf{P}_0\mathbf{P}_1}$ 和 $\overrightarrow{\mathbf{P}_1\mathbf{P}_2}$ 分别做线性插值，得到由同一个参数 t 分别在上述两条一阶曲线上对应的点

$$\mathbf{Q}_0 = \mathrm{lerp}(\mathbf{P}_0, \mathbf{P}_1, t) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1, \tag{6.2}$$

$$\mathbf{Q}_1 = \mathrm{lerp}(\mathbf{P}_1, \mathbf{P}_2, t) = (1-t)\mathbf{P}_1 + t\mathbf{P}_2. \tag{6.3}$$

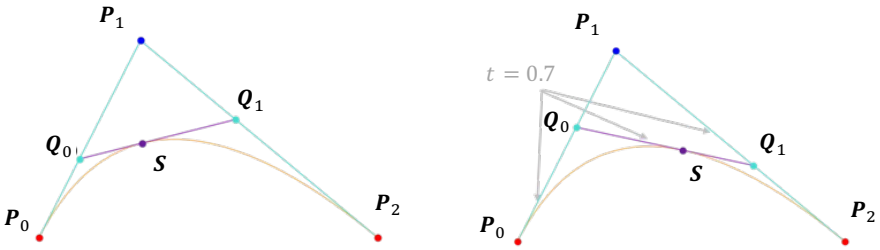


图 6.4: 德卡斯特里奥算法构造二阶贝塞尔曲线.

2. 第二轮，通过对 $\overrightarrow{\mathbf{Q}_0\mathbf{Q}_1}$ 做线性插值得到，得到同一个参数 t 对应的点

$$\mathbf{S} = \mathrm{lerp}(\mathbf{Q}_0, \mathbf{Q}_1, t) = (1-t)\mathbf{Q}_0 + t\mathbf{Q}_1. \tag{6.4}$$

3. 令 t 遍历 $[0,1]$ 取值范围，即得到由 $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ 控制的二阶贝塞尔曲线.

这个算法称为德卡斯特里奥算法 *(de Casteljau's algorithm)*，效率较高，编程方便且数值稳定性较好，因而得到了广泛的使用.

类似地，我们同样可以定义三阶贝塞尔曲线. 三阶贝塞尔曲线由四个控制点给出，可以通过三轮线性插值得到，如图6.5所示.

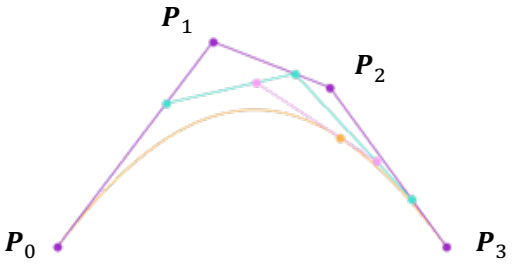


图 6.5: 德卡斯特里奥算法构造三阶贝塞尔曲线.

进而我们可以将贝塞尔曲线的阶数推广到任意高，n 阶贝塞尔曲线由 $n+1$ 个控制点给出. 但贝塞尔曲线每个控制点对曲线的影响不是局部的，如图 (6.6) 所示，因此随着阶数的提高，通过控制点对曲线形状做出有效控制的难度也随之提高.

实践中最常用的依然是三阶贝塞尔曲线，Powerpoint 中的曲线，PS 软件中的钢笔，以及字体设计软件 (图6.1) 都是采用三阶贝塞尔曲线.

由上述构造方法我们可以求出贝塞尔曲线的数学表达式. 以二阶贝塞尔曲线为例，将式 (6.2)(6.3) 代入式 (6.4) 得：

$$\begin{aligned}
\mathbf{S} &= (1-t)\mathbf{Q}_0 + t\mathbf{Q}_1 && (6.5)\\
&= (1-t)((1-t)\mathbf{P}_0 + t\mathbf{P}_1) + t((1-t)\mathbf{P}_1 + t\mathbf{P}_2) && (6.6)\\
&= (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2. && (6.7)
\end{aligned}$$

不难发现，上式中对各控制点的权重系数正是二项式 $(t+(1-t))^2$ 展开后的各项. 更一般

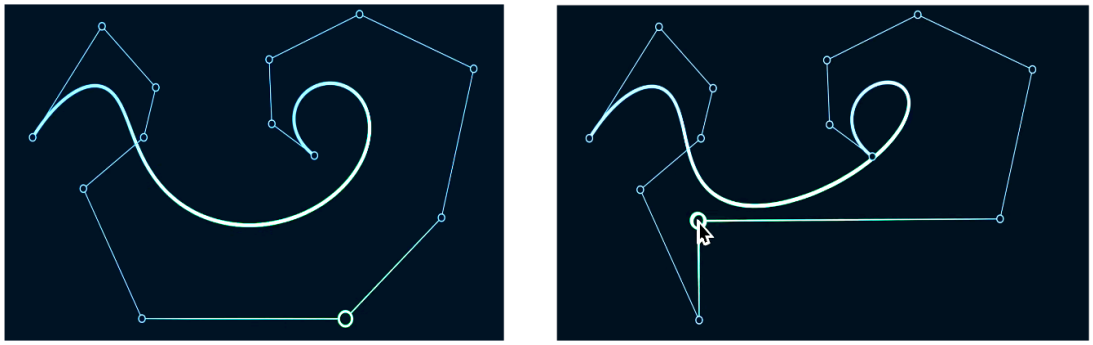


图 6.6: 一条十二阶贝塞尔曲线，挪动其中一个顶点会改变整条曲线的形状，因为难以对曲线形状进行局部调整 (图源：*The continuity of splines*).

地，n 阶贝塞尔曲线的数学表达式为：

$$\mathbf{S}(t) = \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-k} t^k \mathbf{P}_k, \tag{6.8}$$

其中，系数多项式

$$B_{n,k}(t) = \binom{n}{k} (1-t)^{n-k} t^k, \quad k = 0, 1, ..., n, \tag{6.9}$$

被称为 n 阶伯恩斯坦多项式 *(Bernstein polynomials)*.

6.2.2　性质

首先，我们考察一条贝塞尔曲线的端点．根据式 (6.8) 可得 $\mathbf{S}(0) = \mathbf{P}_0$, $\mathbf{S}(1) = \mathbf{P}_1$，即，贝塞尔曲线必然经过两端控制点，但一般来说不会经过中间其余控制点．式 (6.8) 对参数 t 求导可得：

$$\begin{aligned} \mathbf{S}'(t) &= \frac{\mathrm{d}}{\mathrm{d}t} \sum_{k=0}^{n} B_{n,i}(t) \mathbf{P}_k \\ &= n \sum_{k=0}^{n-1} B_{n-1,i}(t) (\mathbf{P}_{k+1} - \mathbf{P}_k). \end{aligned} \tag{6.10}$$

故端点处梯度为

$$\mathbf{S}'(0) = n(\mathbf{P}_1 - \mathbf{P}_0), \tag{6.11}$$

$$\mathbf{S}'(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}). \tag{6.12}$$

可见，端点处切线分别沿着 $\overrightarrow{\mathbf{P}_0\mathbf{P}_1}$ 连线和 $\overrightarrow{\mathbf{P}_n\mathbf{P}_{n-1}}$ 连线的方向．因此，如果我们希望两条相连的贝塞尔曲线在连接点处光滑过渡，只需要施加边界条件令它们在公共端点两旁的控制点在同一条直线上即可，如图6.7．这一条件保证了 G^1 光滑，而若要达到 C^1 光滑，则还需要令 $\|\mathbf{P}_n - \mathbf{P}_{n-1}\| = \|\mathbf{P}_{n+1} - \mathbf{P}_n\|$，即，约束加强为 $\mathbf{P}_n - \mathbf{P}_{n-1} = \mathbf{P}_{n+1} - \mathbf{P}_n$．因此，一条三阶贝塞尔曲线的四个控制点刚好可以让我们自由控制两端的切线，这也是我们一般使用三阶而不是二阶贝塞尔曲线的原因．

贝塞尔曲线另一条重要的性质是凸包性质 (convex-hull property)．由于贝塞尔曲线的权重均非负，且曲线上任意一点关于控制点的权重和为 1，故而由凸包的定义可知，贝塞尔曲线被包围在由它所有控制点组成的最小凸包当中，即控制点凸包给出了一个贝塞尔曲线

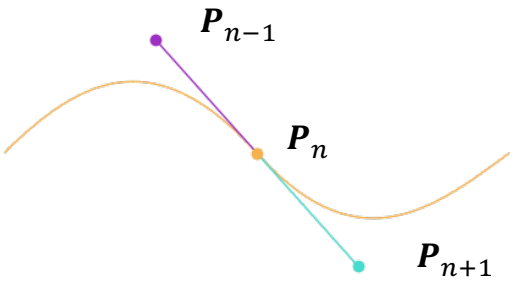


图 6.7: 两条贝塞尔曲线的光滑连接.

的包围盒 (尽管不是最小包围盒). 这个性质能帮助我们很好地规划贝塞尔曲线的路径. 比如在规划虚拟场景中相机的轨迹时，我们可以通过规划控制点的凸包，避免相机走到我们不想其到达的位置.

6.3　样条曲线

样条曲线 *(spline curve)* 是另一类被广泛运用于数值分析、工程设计等领域的曲线. 它结构简单、易于分析，同时可以对复杂的曲线形状进行拟合，便于进行交互式设计. 它的名称来源于早期的船舶和飞机制造工业中的一种技术：通过将细木条 (称为“样条”) 穿过布置在大型设计阁楼地板上的点来构建飞机模板 (如图6.8).

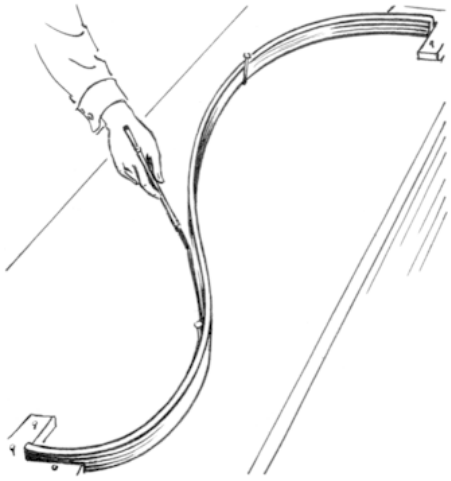


图 6.8: 早期的样条技术 (图源：维基百科).

样条曲线不依靠另加约束就可以满足整条曲线的 C^1 光滑的性质. 样条函数作为构成样条曲线的函数基底，具有紧支撑 (compact support) 的特性，即，每个样条函数只会在一个子区间内取非零值，而在该子区间外取值为 0. 因此每个样条函数只会对曲线的某个局部产生影响，这就为对曲线形状进行局部的控制与调整提供了便利. 也因此，样条函数的阶数的选取与控制点的具体数目无关.

用于构造样条曲线的样条函数多种多样，包括曲线近似的样条 (approximation splines) 和曲线插值的样条 (interpolation splines). 如图 (6.9) 所示，前者不保证拟合出的曲线过每个控制点，但拥有凸包性质，如 B 样条；后者保证拟合出的曲线过每个控制点，但不具有凸包性质，如二次样条、三次样条.

(a) 插值样条　　(b) 近似样条

图 6.9: 不同种类的样条曲线.

我们这里以最常用的 B 样条和三次样条为例来分别介绍样条曲线.

6.3.1　B 样条与 NURBS

B 样条是样条函数的一种，它是基函数样条 (basis splines) 的简称.

选定参数区间 $[a,b]$ 内一列非减的数列，满足：$a=t_0\le t_1\le\cdots\le t_m=b$，这 $m+1$ 个数将参数区间划分成 m 段，被称为节点 *(knots)*，由这 $m+1$ 个节点组成的集合被称为节点向量 *(knot vector)*.

一个 n 阶 B 样条是由所有次数不超过 n 的 B 样条基 (B-spline basis functions) 组成的，记第 i 个 p 次 B 样条基为 $N_{i,p}(t)$, $0\le i\le m$, $0\le p\le n$，它们是一组根据 Cox de Boor recursion 递归定义的函数：

$$N_{i,0}(t)=\begin{cases}1, & t_i\le t<t_{i+1},\\ 0, & \text{otherwise},\end{cases}\tag{6.13}$$

$$N_{i,p}(t)=\frac{t-t_i}{t_{i+p}-t_i}N_{i,p-1}(t)+\frac{t_{i+p+1}-t}{t_{i+p+1}-t_{i+1}}N_{i+1,p-1}(t),\tag{6.14}$$

并且满足性质：

$$\sum_{i=0}^{m}N_{i,p}(t)=1,\quad\forall t\in[0,1].\tag{6.15}$$

图 (6.10) 展示了 0 至 5 阶的 B 样条基的函数图像.

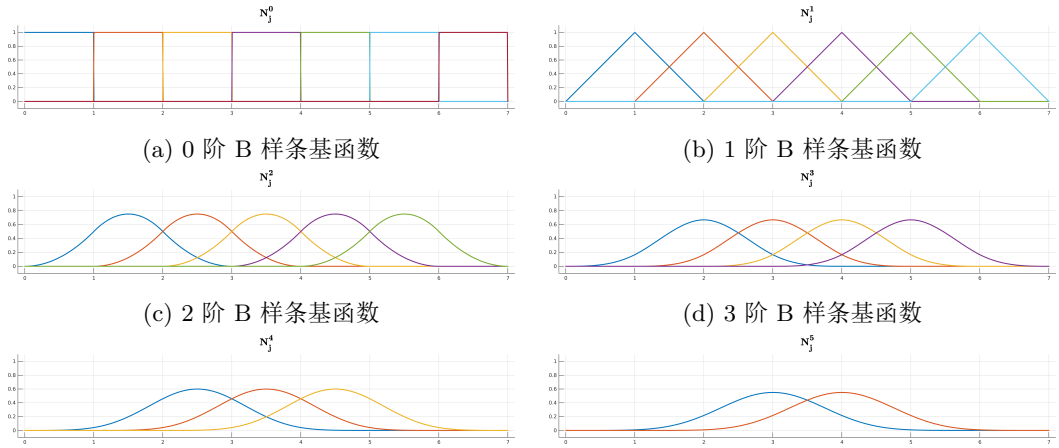
给定控制点点列 $\{\mathbf{P}_0,\mathbf{P}_1,\ldots,\mathbf{P}_m\}$，那么利用 n 次 B 样条得到的拟合结果即为：

$$\mathbf{b}(t)=\sum_{i=0}^{m}N_{i,n}(t)\mathbf{P}_i\tag{6.16}$$

事实上，这个过程既可以看成是以 B 样条基为权重函数对点列 $\{\mathbf{P}_i\}$ 进行拟合，也可以看成是以 $\{\mathbf{P}_i\}$ 为权重对 B 样条基进行线性组合，从而将局部曲线组装成全局曲线.

从图像中可以看出，当 B 样条的阶数增加时，单个 B 样条基的作用范围也在扩大，每个 p 阶的 B 样条基在 $p+1$ 个连续子区间上取非零值，也将单个节点的对曲线的影响效果扩展到 $p+1$ 个附近子区间. 例如，对一个三阶 B 样条曲线来说，为了定义 $[t_i,t_{i+1}]$ 段的曲线，它将会使用到 $\mathbf{P}_{i-2}$, $\mathbf{P}_{i-1}$, $\mathbf{P}_i$, $\mathbf{P}_{i+1}$ 四个点的位置信息，并且在每个节点处保证了一阶、二阶梯度连续.

节点的数值是可以相同的，若存在 $t_i=t_{i+1}=\cdots=t+i+k-1$, $k>1$，则称其为一个重复度为 k 的多重节点 (multiple knot)，否则称其为一个简单节点 (simple knot). 节点重复会降低该节点处的光滑度要求，p 阶 B 样条曲线在重复度为 k 的节点处是 C^{p-k} 光滑的. 一般来说，B 样条曲线不会经过控制点，如图 (6.11(a)) 所示，是一条开放 B 样条 (open



(a) 0 阶 B 样条基函数　(b) 1 阶 B 样条基函数

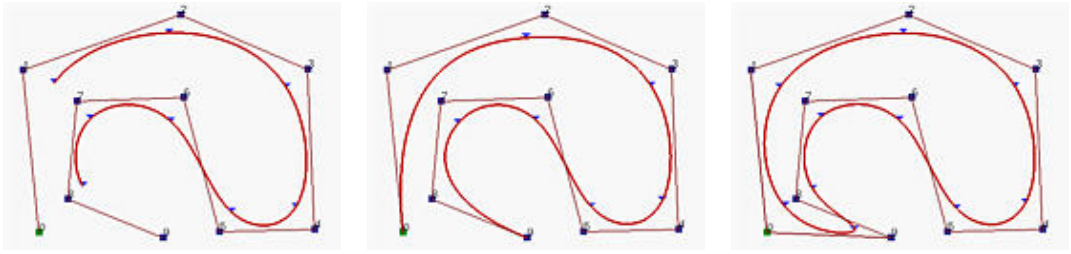
(c) 2 阶 B 样条基函数　(d) 3 阶 B 样条基函数

(e) 4 阶 B 样条基函数　(f) 5 阶 B 样条基函数

图 6.10: 0 至 5 阶的 B 样条基函数.

B-splines). 若令两端的节点重复 $p+1$ 次，B 样条曲线将会经过两端端点并且两端切线平行于控制点连线，这类曲线被称为剪裁 B 样条 (clamped B-splines)，如图 (6.11(b)) 所示. 通过重复首尾的节点和控制点，还可以构造出闭合 B 样条 (closed B-splines)，如图 (6.11(c)) 所示.

若一条剪裁 B 样条曲线除两端端点外剩余内部控制点数为 $p-1$，那么它就等价于一条 p 阶贝塞尔曲线. 因此 B 样条曲线可以看成是贝塞尔曲线的一般化.



(a) 开放 B 样条曲线　(b) 剪裁 B 样条曲线　(c) 闭合 B 样条曲线

图 6.11: 不同种类的 B 样条曲线 (图源：MTU cs3621 course).

对 B 样条的一个重要扩展是非均匀有理 B 样条 (non-uniform ration B-splines，简称 NURBS)，在计算机辅助设计、计算机辅助制造、计算机图形学等领域中，NURBS 有着十分重要的应用，为曲线、曲面设计提供了较高的灵活度. “非均匀”指的是 B 样条的节点在参数区间内不等距分布，“有理”指的是 B 样条对每个控制点也额外施加了一个权值，当权值均等于 1 时，NURBS 曲线就退化成了普通的 b 样条曲线. 因此 NURBS 实质上是齐次坐标 (homogeneous coordinate) 思想下的 B 样条，用权值的形式取代了多次重复设置节点这一笨重的操作，一条 NURBS 曲线的数学形式一般为：

$$\mathbf{S}(t)=\frac{\sum_{i=0}^{m}N_{i,n}(t)w_i\mathbf{P}_i}{\sum_{i=0}^{m}N_{i,n}(t)w_i}. \tag{6.17}$$

其中 w_i, $i=0,\ldots,m$ 是对控制点的权值，分母被用于对系数进行归一化.

6.3.2　三次样条

B 样条并不是一种插值方法，即，B 样条曲线一般不经过控制点．如果我们希望给出的点列正位于曲线上，那么我们需要采用另外的方法来拟合出曲线．而两点不能唯一决定一条曲线段，因此在进行高阶插值时，需要添加额外的约束来帮助我们决定高阶函数系数，如端点处的导数约束或二阶导数约束，从而作出合理的插值．

给出点列 $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_m\}$，其中每个点的坐标为 $\mathbf{P}_i = (x_i, y_i)^{\mathrm{T}}$, $i = 0, \ldots, m$．三次样条曲线假设每个子区间 $[x_{i-1}, x_i]$, $i = 1, \ldots, m$ 内的曲线段可以用一个三次函数来刻画：$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$，并且满足的如下的连续性要求：

1. $f_i(x)$ 两端满足插值特性．

$$f_i(x_{i-1}) = y_{i-1},\ f_i(x_i)) = y_i,\ i = 1, \cdots, m \tag{6.18}$$

2. 公共顶点的一阶导数连续．

$$f_i'(x_i) = f_{i+1}'(x_i),\ i = 1, \cdots, m-1 \tag{6.19}$$

3. 公共顶点的二阶导数连续．

$$f_i''(x_i) = f_{i+1}''(x_i),\ i = 1, \cdots, m-1 \tag{6.20}$$

这里，a_i, b_i, c_i, d_i 为待定系数．对 $m+1$ 个点做三次样条插值，m 段曲线共包含 $4m$ 个待定系数，相应地，需要 $4m$ 个约束来帮助我们确定系数．以上三组连续性要求共提供了 $2m + (m-1) + (m-1) = 4m - 2$ 个约束，剩余两个约束来源于对最两侧的线段端点 (x_0, y_0) 和 (x_m, y_m) 的约束，我们将额外引入的对最外侧端点的约束称为*边界条件 (boundary conditions)*．常见的边界条件有以下几类：

- 给定端点处的一阶导数 $f_1'(x_0)$, $f_m'(x_m)$．
- 给定端点处的二阶导数 $f_1''(x_0)$, $f_m''(x_m)$，全为 0 时一般称为自然边界条件．
- 若点列中 $y_0 = y_m$，另外要求 $f_1'(x_0) = f_m'(x_m)$, $f_1''(x_0) = f_m''(x_m)$，则可得到周期性三次样条，这被称为周期性边界条件．

通过上述边界条件，我们可以补全 $4m$ 个约束，且每条约束是关于待定系数的一次函数，因此我们可以通过求解线性方程组的方式解出所有待定系数并得到三次样条曲线．求解线性方程组已有大量成熟的数值算法，这里我们不进行详细讨论，感兴趣的同学可以参见 C++ 代数运算库Eigen中的实现．

事实上，三次样条曲线是使得泛函 $J(f) = \int_{x_0}^{x_m} |\, f''(x)\,|^2 \mathrm{d}x$ 值最小的函数．而曲线的二阶导是对曲线曲率的近似刻画，一根弯曲弹性条的能量密度与其曲率平方成正比，因此三次样条曲线是对受到 m 个点约束的最低能量弹性条的形状近似．另外三次样条曲线保证了曲线的插值特性，并且满足全局 C^2 光滑的特性，即，曲线上一阶、二阶导数处处存在且连续．

6.3.3　曲线图形的光栅化

为了将曲线显示在屏幕上，我们同样需要对曲线进行光栅化操作将其转化成位点阵图．一个简单的想法是通过细分将曲线划分成若干段子曲线，每段子曲线的曲率小到足以用直线来近似，而后用绘制直线的方法来近似每段子曲线，不过这种做法也会遇到前面提到的问题，即，需要在光滑度和计算复杂性之间寻求一个平衡．自适应细分 (adaptive subdivision) 根据曲线不同位置的曲率大小动态地调整细分的程度，从而在弯曲程度较大的地方获得更多的细节而在平滑的地方节省算力．另一种想法则是采用曲线版本的 DDA 算法，即，根据

参数 t 对曲线进行采样，并将采样对应的像素点绘制在屏幕上．为了提高效率，采样可以通过增量累加的方式进行的．

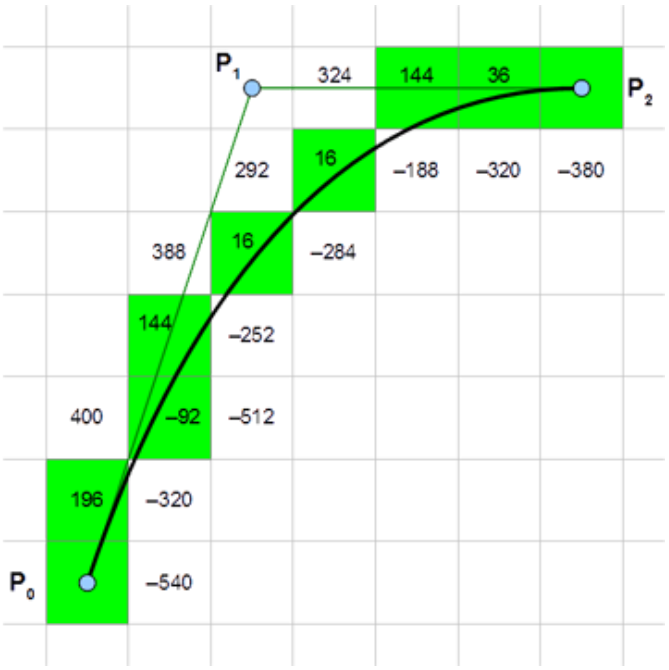


图 6.12: 一条二阶贝塞尔曲线的光栅化．

与直线类似，我们也可以采用变体的布雷森汉姆算法．在曲线的情形下，像素点 (x, y) 到曲线的偏差可以定义为 $e = F(x, y)$，这里 $F(x, y)$ 是曲线转化成对应的隐式方程，当像素点在曲线上时，偏差值为 0．因此曲线版本的布雷森汉姆算法可以如下设计 [5]：对于曲线 $F(x, y) = 0$，首先对曲线进行细分从而保证其分段单调．这里不妨假设其单增．当我们已绘制像素点 (x, y)，算法需要从 $(x+1, y)$, $(x, y+1)$, $(x+1, y+1)$ 中选取下一个需要绘制的点．算法以 $(x+1, y+1)$ 的角度考察是否需要在 x 或 y 方向上步进一个像素：分别计算偏差值 $e_x = F(x, y+1)$, $e_y = F(x+1, y)$, $e_{xy} = F(x+1, y+1)$ 若偏差 $|e_{xy}| < |e_x|$，则需要在 x 方向上加 1；若偏差 $|e_{xy}| < |e_y|$，则需要在 y 方向上加 1．而这里由于曲线单增的假设，$e_y \le e_{xy} \le e_x$ 必然成立，故可以不计算绝对值，上述两个判断条件等价于：$e_y + e_{xy} > 0$ 及 $e_{xy} + e_x < 0$，由此来确定变体布雷森汉姆算法的下一个绘制点．一个实例如图6.12所示．

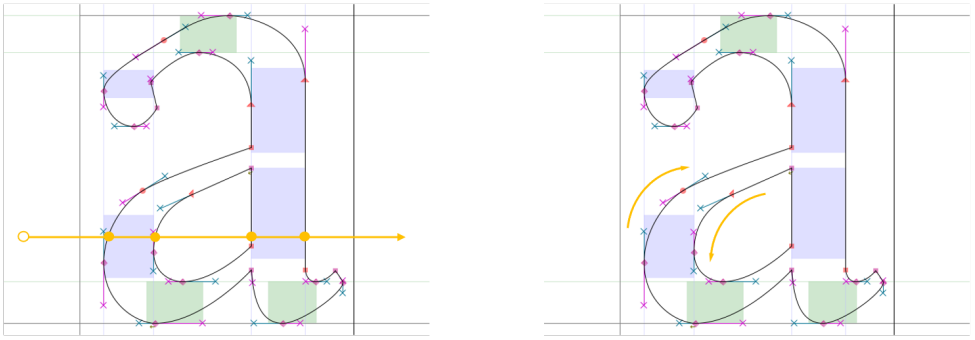


图 6.13: 渲染曲线轮廓构成的字母'a'．

而对于由曲线组成的图案，如图6.13中的字体，类似地可以利用扫描线方法绘制．当扫描线与图形产生多个交点时，第奇数交点到第偶数交点的中间线段对应需要绘制的像素，而第

偶数交点到第奇数交点间的线段则不需要绘制 (对应外部或图形的中间空洞). TTF(truetype font) 文件格式在渲染字体时将采用了类似的思想：将外部轮廓定义为顺时针环绕而内部轮廓定义为逆时针环绕，从某一像素点 P 向任意方向做射线，当自左向右跨越轮廓边界时将环绕数加 1，自右向左跨越时将环绕数减 1，于是当最终结果非零时则表示点 P 在内部，为零则表示在外部. 求解曲线与扫描线的交点需要求解一元方程，若曲线阶数大于三，则需要借助于牛顿下降法等数值工具进行求解. 也因此图形学中更多应用低阶曲线，例如 TTF 字体由线段和二阶贝塞尔曲线组成.

(a) 简单光栅化.

(b) 无提示的抗锯齿光栅化.

(c) 带反锯齿与提示的光栅化.

(d) RGB 平板显示器上的带提示和亚像素渲染的光栅化.

图 6.14: 使用不同算法的字体光栅化 (图源：维基百科).

另外，现代图形学中也为更高质量的曲线渲染引入了反走样算法、亚像素算法等改进方案，如图6.14，这里不再展开.