第三章　　二维图形生成技术

本 章 重 点

掌握在显示器上绘制最基本的二维图形——线和多边形的原理和方法。

主要包括：

1. 直线
2. 二次曲线（圆弧、抛物线）
3. 自由曲线
4. 多边形

难点：自由曲线的绘制，多边形的扫描转换

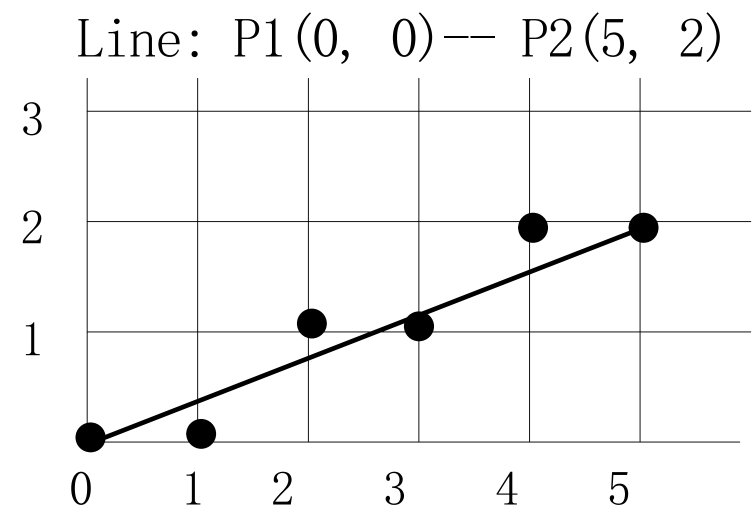
3.1 直线图形

一. 生成直线的 DDA 算法

无论是显示器还是绘图机，都可以看成有一个网格(离散单元组成的矩阵)存在，对显示器来说每一个像素就是一个网格点，对绘图机来说笔每走一步的终点也可以看成是一个网格的结点。

在显示器上表示一条直线，就是要用最靠近直线的一些网格点来代表这一直线。

这个网格就构成屏幕和绘图机纸张的一个坐标系，相邻两个网点的距离取为1，每个网格点的坐标均取整数。



假设　直线的起点坐标为$P_1(x_1, y_1)$，终点坐标为$P_2(x_2, y_2)$

x方向的增量为 $\triangle x = x_2 - x_1$ ；y方向上增量为 $\triangle y = y_2 - y_1$

直线的斜率为　$\mathbf{k = \triangle y / \triangle x}$

当 $\mathbf{\triangle x > \triangle y}$ 时，让 x 从 x_1 到 x_2 变化，每步递增 1，那么，x 的变化可以表示为 $x_{i+1} = x_i + 1$

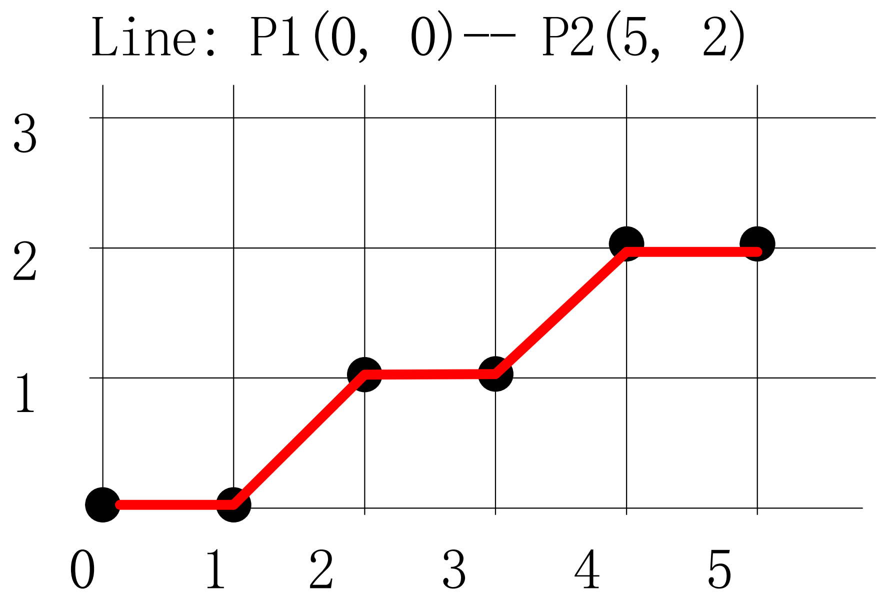
y 的变化可以表示为 $y_{i+1} = y_i + k$

用上式可求得图中直线 P_1P_2 和 y 方向网格线的交点，但显示时要用象素点(图中的网格结点)来表示，所以要用舍入的办法来找到最靠近交点处的象素点，并用其来表示直线段。

这个方法称之为数字微分分析法，简称DDA。

例：画直线段

x	y	y+0.5	int(y+0.5)
0	0	0.5	0
1	0.4	0.9	0
2	0.8	1.3	1
3	1.2	1.7	1
4	1.6	2.1	2
5	2	2.5	2



注：网格点表示象素

算法描述如下：

```
int  x1， y1， x2， y2;
int  x;
double  dx， dy， k， y;
dx＝x2－ x1
dy＝y2 － y1
k＝dy / dx
x＝x1
y＝y1
for (  ;  x<＝x2;  x＋＋)
{
    putpixel (x， (int)(y+0.5)， pixelcolor )
    y＝y＋k
}
```

该算法仅适用于 | k | ≤1 的情况，而当 | k | ＞1时，则需将 x 和 y 的位置交换。

二. 生成直线的中点画线算法

- 采用增量思想的DDA算法，每计算一个象素，只需计算一个加法，是否最优？

目标：进一步将一个加法改为一个整数加法。

- DDA算法采用点斜式，可否采用其他的直线表示方式？

直线段的隐式方程：

设直线段的起点和终点分别为：(x_0,y_0)，(x_1,y_1)

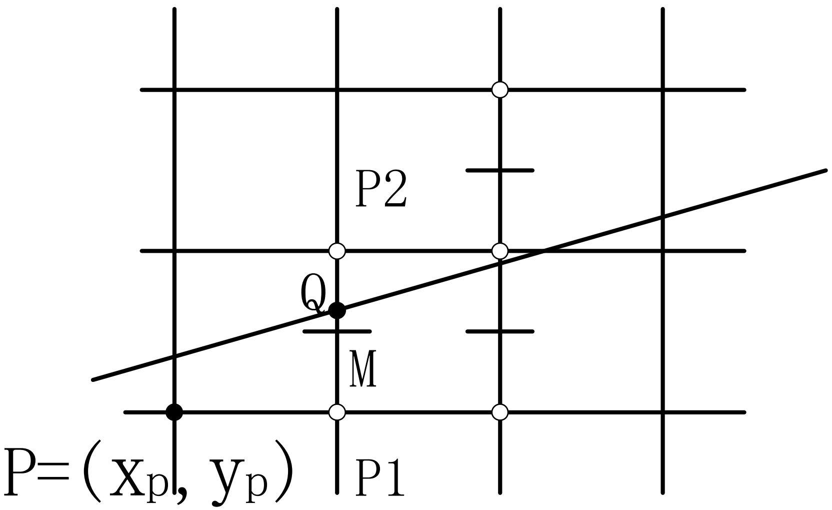
$$F(x,y)=ax+by+c=0$$

其中 $a=y_0-y_1,b=x_1-x_0,c=x_0y_1-x_1y_0$ 均为整数

- 基本思想

设当前象素点为(x_p, y_p)，下一个象素点为P1 或P2 。

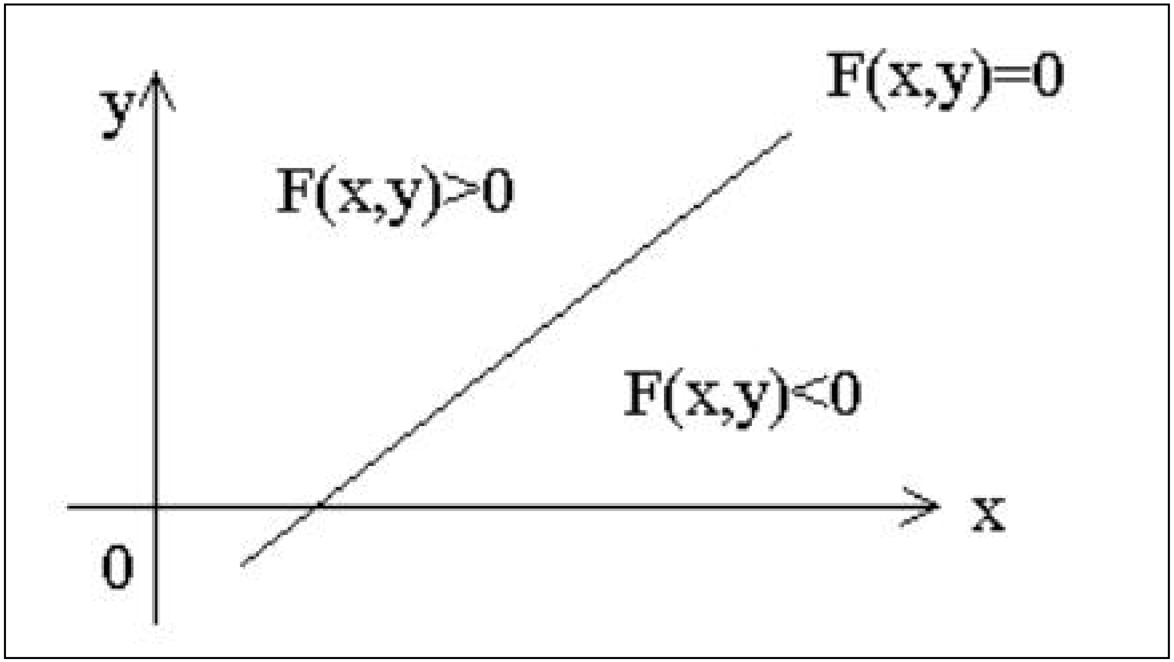
设$M=(x_p+1, y_p+0.5)$，即为p1与p2的中点，Q为实际直线与$x=x_p+1$的交点。将Q与M的y坐标进行比较。



– 若M在Q的下方，应取P2为下一点

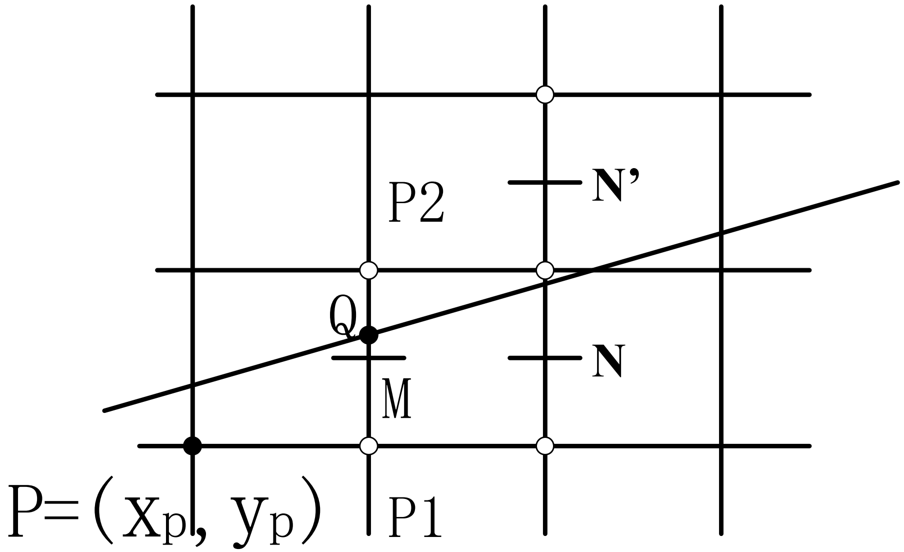
– 若M在Q的上方，应取P1为下一点。

- 直线的正负划分性



直线上方点：F(x,y)＞0

直线下方的点：F(x,y)＜0



构造判别式:$d=F(M)=F(x_p+1,y_p+0.5)$

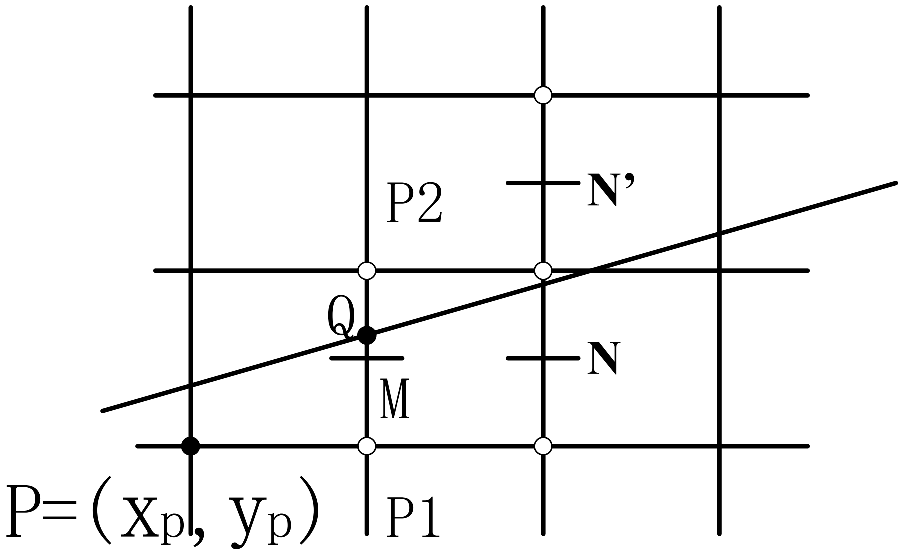
$$=a(x_p+1)+b(y_p+0.5)+c$$

其中 $a=y_0-y_1,b=x_1-x_0,c=x_0y_1-x_1y_0$

若$d<0$，M在直线(Q点)下方，取右上方P2为下一个象素；

若$d>0$，M在直线(Q点)上方，取右方P1为下一个象素；

若$d=0$，选P1或P2均可，约定取P1为下一个象素；



- 增量算法

$d=a(x_p+1)+b(y_p+0.5)+c$

若当前象素处于$d \geq 0$情况，则取正右方象素P_1 (x_p+1, y_p), 要判下一个象素位置，应计算

$d_1=F(x_p+2, y_p+0.5)=a(x_p+2)+b(y_p+0.5)+c=d+a$； 增量$delta_1$为a

若$d<0$时，则取右上方象素P_2 (x_p+1, y_p+1)。要判断再下一象素，则要计算

$d_2= F(x_p+2, y_p+1.5)=a(x_p+2)+b(y_p+1.5)+c=d+a+b$； 增量$delta_2$为$a+b$

- 实现整数运算

画线从起点(x_0, y_0)开始，d的初值

$d_0=F(x_0+1, y_0+0.5)=F(x_0, y_0)+a+0.5b=a+0.5b$。

解决方法：2d代替d，则

$d_0=2d_0=2a+b$

$d_1=2d_1=2d+2a$，增量$delta_1=2a$

$d_2=2d_2=2d+2a+2b$，增量$delta_2=2a+2b$

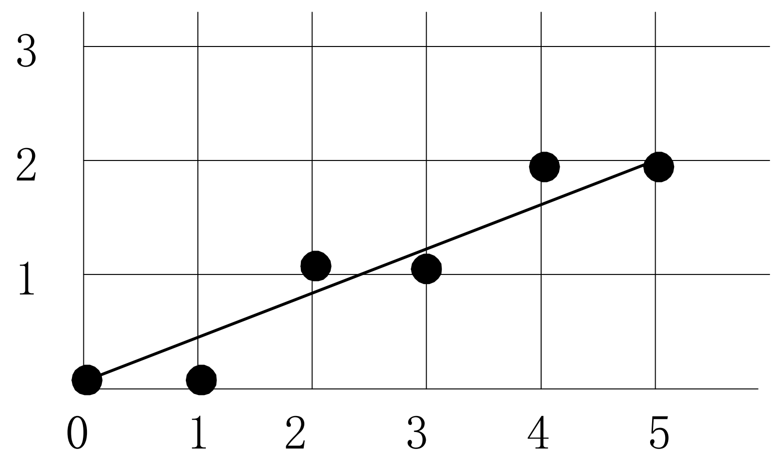
```
void Midpoint Line (int x0,int y0,int x1, int y1,int color)
{   int a, b, delta1, delta2, d, x, y;
    a=y0-y1, b=x1-x0, d=2*a+b;
    delta1=2*a, delta2=2* (a+b);
    x=x0, y=y0;
    putpixel(x, y, color);
    while (x<x1)
    { if (d<0)       {x++, y++, d+=delta2; }//选择右上方象素
       else       {x++, d+=delta1;}//选择正右方象素
       putpixel (x, y, color);
    }  /* while */
 } /* mid PointLine */
```

例：用中点画线法 $P_0(0,0)-P_1(5,2)$

$$a = y_0 - y_1 = -2, b = x_1 - x_0 = 5$$

$$d_0 = 2a + b = 1, delta_1 = 2a = -4, delta_2 = 2(a+b) = 6,$$

i	x_i	y_i	d	
1	0	0	1	
				$+delta_1$
2	1	0	-3	
				$+delta_2$
3	2	1	3	
				$+delta_1$
4	3	1	-1	
				$+delta_2$
5	4	2	5	
				$+delta_1$
6	5	2	1	



三. 生成直线的 Bresenham 算法

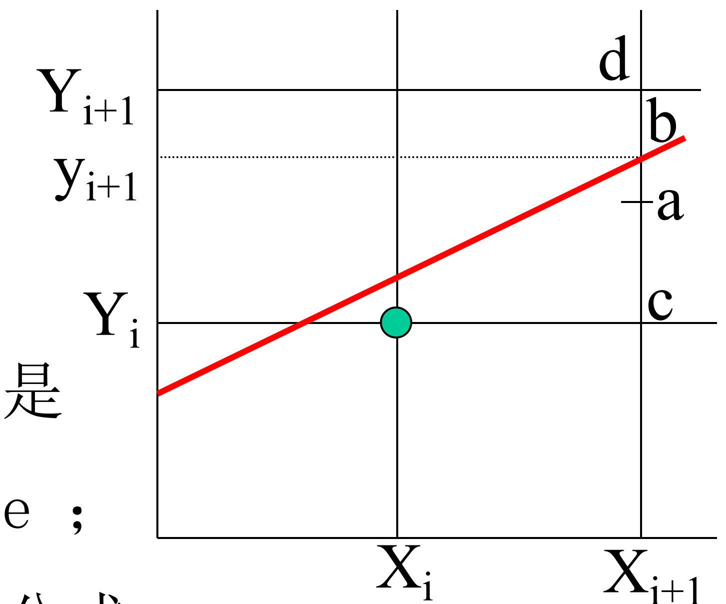
设 $k=\triangle y/\triangle x$ ， 先讨论 $0\leqslant k\leqslant 1$的情况：

若以屏幕上x方向的象素点作为横坐标，则有 $x_{i+1}-x_i=1$

而 $y_{i+1}=y_i+k(x_{i+1}-x_i)=y_i+k$ (1)

设 b 点是直线上的点，其坐标是(x_{i+1}, y_{i+1})，显然，该点只能用屏幕上的象素点c 或 d 来表示。

设 a 为c、d 的中点，若 b 在 a 的上面则应取 d，否则应取 c。



关键问题：

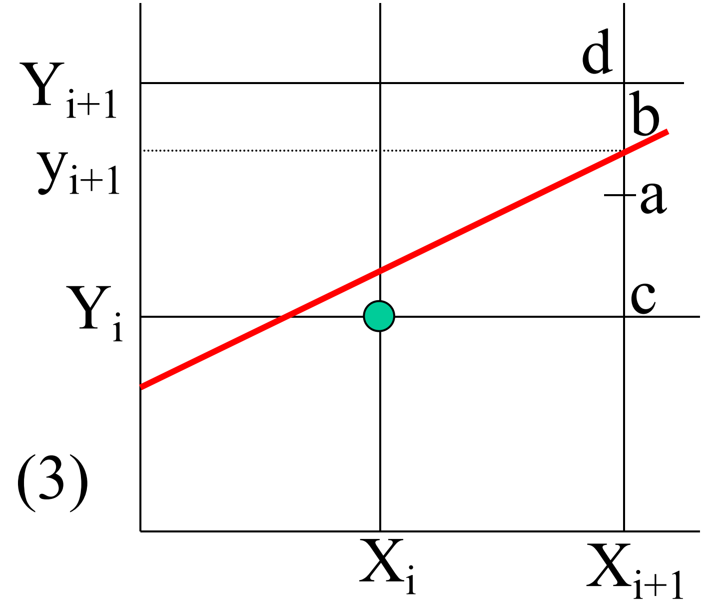
(1) 如何判断 b是在 a 的上面还是下面，设置一个标志变量 e ；

(2) 如何建立标志变量 e 的递推公式。

设 $e_{i+1}=y_{i+1}-Y_i-0.5$ (2)

若b在a的上面，则有 $e_{i+1}>0$

若b在a的上面，则有 $e_{i+1}<0$



由图中可知：当 $\left.\begin{array}{ll} e_{i+1}\geqslant 0 \text{时} & Y_{i+1}=Y_i+1 \\ e_{i+1}<0 \text{ 时} & Y_{i+1}=Y_i \end{array}\right\}$ (3)

递推：

由(2)、(3)式可得：

$$e_{i+2}=y_{i+2}-Y_{i+1}-0.5=y_{i+1}+k-Y_{i+1}-0.5$$

$$=\begin{cases} y_{i+1}-Y_i-0.5+k-1 & e_{i+1}\geqslant 0 \\ y_{i+1}-Y_i-0.5+k & e_{i+1}<0 \end{cases}$$

$$=\begin{cases} e_{i+1}+k-1 & e_{i+1}\geqslant 0 \\ e_{i+1}+k & e_{i+1}<0 \end{cases}$$

算法描述如下：

$e_{i+1} = y_{i+1} - Y_i - 0.5$

当 $e_{i+1} \geqslant 0$ 时 $Y_{i+1} = Y_i + 1$, $e_{i+2} = e_{i+1} + k - 1$

当 $e_{i+1} < 0$ 时 $Y_{i+1} = Y_i$, $e_{i+2} = e_{i+1} + k$

```
△x＝x2－x1
△y＝y2－y1
k＝△y / △x
e ＝k－0.5
x＝x1
y=y1
for ( ;  x<＝ x2 ;  x＋＋)
{
    putpixel (x， y， pixelcolor);
    if (e<0)  e＝e＋k;
    else { y＝y＋1;   e＝e＋k－1; }
}
```

讨论：

斜率不同时：

以上讨论的是 $0 \leqslant k \leqslant 1$ 的情况，即 $0<\triangle y<\triangle x$ 的情况；

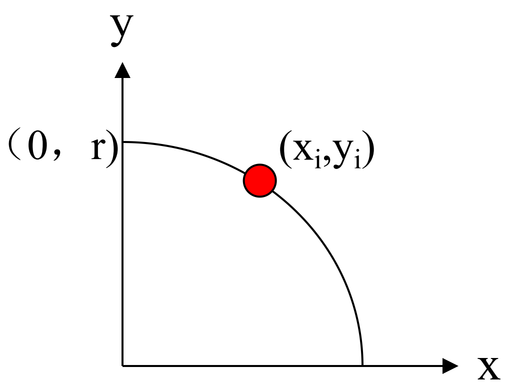
若是 $0<\triangle x<\triangle y$ 的情况，则需将 x 和 y 的位置交换。

方向不同时：

若$\triangle y<0$或$\triangle x<0$时，要将算法中的 $y=y+1$换成$y=y-1$、 $x=x+1$换成$x=x-1$。

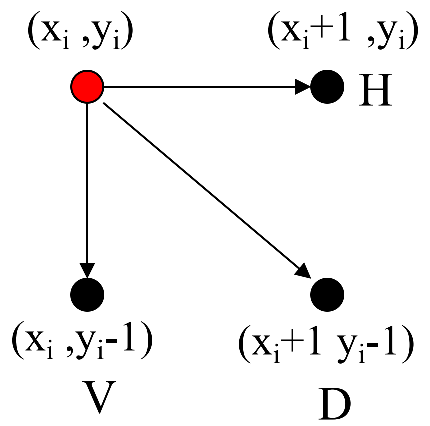
3.2 二次曲线

一.Bresenham画圆算法



该算法以点(0,r)为起点，按顺时针方向生成圆时，相当于在第一象限内，所以 y 是 x 的单调递减函数。

从圆上任一点出发，按顺时针方向生成圆时，为了最佳地逼近该圆，对于下一个象素的取法只有三种可能的选择，即右方象素(H)、右下角象素(D)、下方象素(V)。



设 $\triangle_i=(x_i+1)^2+(y_i-1)^2-r^2$

若$\triangle_i<0$，则右下角点在圆内，此时只可能取象素点 H 或 D

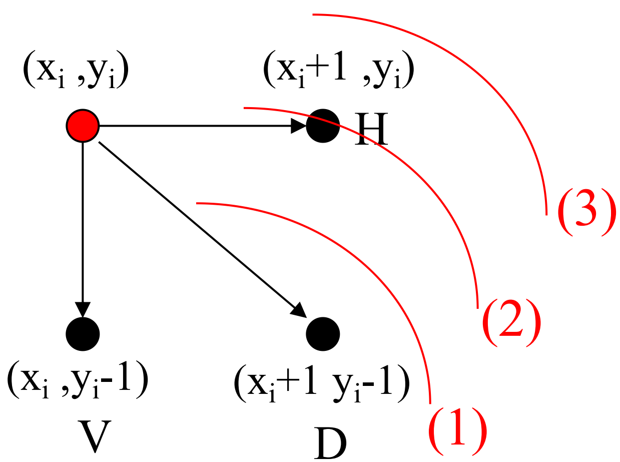
设$\delta_1=|(x_i+1)^2+(y_i)^2-r^2|-|(x_i+1)^2+(y_i-1)^2-r^2|$

若 $\delta_1<0$ 取H

$\delta_1>0$ 取D

$\delta_1=0$ 二者距离相等，规定取右方象素 H

并可将δ_1进一步化简成 $\delta_1=2(\triangle_i+y_i)-1$



$\delta_1=|S(H)|-|S(D)|$

对于（1），$S(H)>0$， $S(D)<0$

$\delta_1=S(H)+S(D)$

对于（2），$S(H)=0$， $S(D)<0$

$\delta_1=S(H)+S(D)$

对于（3），$S(H)<0$， $S(D)<0$

$\delta_1=-S(H)+S(D)$

表示为$\delta_1=S(H)+S(D)$ 不影响判断

故， $\delta_1=S(H)+S(D)$

$$\delta_1 = |(x_i+1)^2+(y_i)^2-r^2| - |(x_i+1)^2+(y_i-1)^2-r^2|$$

$$= (x_i+1)^2+(y_i)^2-r^2 + (x_i+1)^2+(y_i-1)^2-r^2$$

$$= (x_i+1)^2+(y_i-1)^2-r^2 + (x_i+1)^2+(y_i-1)^2-r^2 + 2y_i-1$$

$$= 2\triangle_i+2y_i-1$$

$$= 2(\triangle_i+y_i)-1$$

$\triangle_i = (x_i+1)^2 + (y_i-1)^2 - r^2$

若$\triangle_i>0$，则右下角点在圆外，此时只可能取象素点 D 或 V

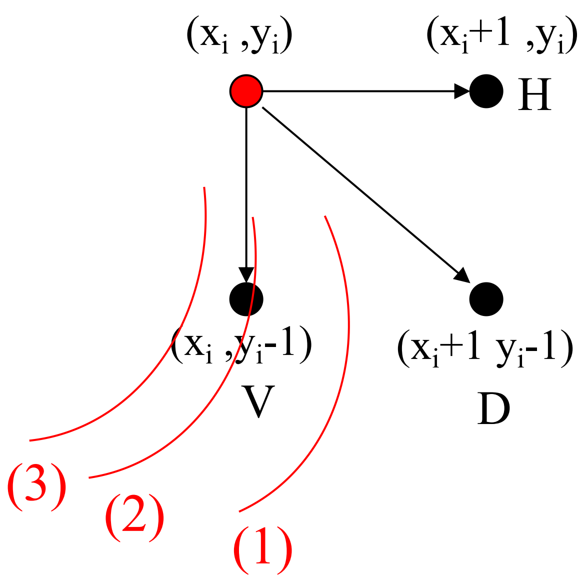
设$\delta_2 = |(x_i+1)^2+(y_i-1)^2-r^2| - |(x_i)^2+(y_i-1)^2-r^2|$

若$\delta_2<0$　取D

$\delta_2>0$　取V

$\delta_2=0$　二者距离相等，规定取右下角象素D

并可将δ_2进一步化简成　$\delta_2=2(\triangle_i-x_i)-1$



$\delta_2=|S(D)|-|S(V)|$

对于（1），　$S(D)>0$，　$S(V)<0$

$\delta_2=\ S(D)+S(V)$

对于（2），　$S(D)>0$，　$S(V)=0$

$\delta_2=\ S(D)+S(V)$

对于（3），　$S(D)>0$，　$S(V)>0$

$\delta_2=\ S(D)-S(V)$

表示为$\delta_2=\ S(D)+S(V)$　不影响判断

故，　$\delta_2=\ S(D)+S(V)$

$$\delta_2 = |(x_i+1)^2+(y_i-1)^2-r^2| - |(x_i)^2+(y_i-1)^2-r^2|$$

$$= (x_i+1)^2+(y_i-1)^2-r^2 + (x_i)^2+(y_i-1)^2-r^2$$

$$= (x_i+1)^2+(y_i-1)^2-r^2 + (x_i+1)^2+(y_i-1)^2-r^2 - 2x_i-1$$

$$= 2\triangle_i-2x_i-1$$

$$= 2(\triangle_i-x_i)-1$$

$\triangle_i=(x_i+1)^2+(y_i-1)^2-r^2$

若$\triangle_i=0$，此时圆上点正好是 D，即取 D。

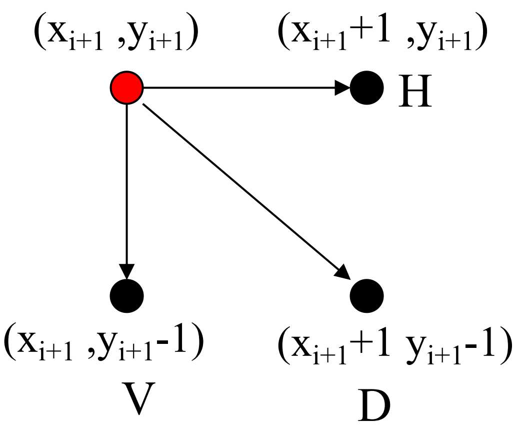
可导出简单增量算法的递推公式：

若设当前圆上点所在的象素为第i个象素，下一个新象素为第i+1个象素，则新象素的坐标及△值的递推公式是：

新象素为H时：$x_{i+1}=x_i+1$

$y_{i+1}=y_i$

$\triangle_{i+1}=\triangle_i+2x_{i+1}+1$



$$\triangle_{i+1}=(x_i+1+1)^2+(y_i-1)^2-r^2$$

$$=(x_i+1)^2+2(x_i+1)+1+(y_i-1)^2-r^2$$

$$=\triangle_i+2(x_i+1)+1$$

$$=\triangle_i+2x_{i+1}+1$$

新象素为D时： $x_{i+1}=x_i+1$

$y_{i+1}=y_i-1$

$\triangle_{i+1} = \triangle_i+2x_{i+1}-2y_{i+1}+2$

新象素为V时： $x_{i+1}=x_i$

$y_{i+1}=y_i-1$

$\triangle_{i+1} = \triangle_i-2y_{i+1}+1$

算法描述如下：

起点：(0，r)

$$\triangle_i = (x_i+1)^2 + (y_i-1)^2 - r^2$$
$$= (0+1)^2 + (r-1)^2 - r^2$$
$$= 2(1-r)$$

```
x=0;        y=r;
delta=2*（1-r）；
while（y>=0） {
   putpixel（x,y,pixelcolor）；
   if（delta<0） {                  // H或D
      delta1=2*（delta+y）-1,
      if（delta1<=0）   direction=1;  // H
      else  direction=2;        // D
   }
   else  if（delta>0）{
      delta2=2*（delta-x）-1; // D或V
      if（delta2<=0）   direction=2;  // D
      else  direction=3;        //  V
   }
   else
      direction =2;             // D
```

```
switch (direction)  {
    case 1:   x++;                          取H
              delta+=2*x+1;
              break;
    case 2:   x++;                          取D
              y--;
              delta+=2*(x-y+1);
              break;
    case 3:   y--;                          取V
              delta+=(-2*y+1);
              break;
  } // End of switch
} // End of while
```

若绘制整个圆，则修改算法如下：

```
x＝0;      y＝r;
delta＝2*（1－r）;
while（y>＝0） {
  putpixel（x,y,pixelcolor）; //第1象限
  putpixel（-x,y,pixelcolor）;//第2象限
  putpixel（-x,-y,pixelcolor）; //第3象限
  putpixel（x,-y,pixelcolor）; //第4象限
  ……
```

二.　抛物线的参数拟合方法

抛物线的参数向量方程：

$$\mathbf{P}(t)=\mathbf{a}t^2 + \mathbf{b}t + \mathbf{c} \qquad (0 \le t \le 1)$$

对应的参数方程：

$$\begin{cases} x(t)=a_x t^2 + b_x t + c_x & (0 \le t \le 1) \\ y(t)=a_y t^2 + b_y t + c_y \end{cases}$$

给定3个控制点$P_0(x_0, y_0)$、$P1(x_1, y_1)$和$P2(x_2, y_2)$，并规定抛物线的边界条件：

（1）当$t=0$时，抛物线过P_0点，且与$\overrightarrow{p_0p_1}$相切；

（2）当$t=1$时，抛物线过P_2点，且与$\overrightarrow{p_1p_2}$相切；

可得方程组：

$$\begin{cases} x(t)=a_x t^2 + b_x t + c_x \\ y(t)=a_y t^2 + b_y t + c_y \end{cases} \quad (0 \le t \le 1)$$

$$\begin{cases} c_x = x_0 \\ c_y = y_0 \\ a_x + b_x + c_x = x_2 \\ a_y + b_y + c_y = y_2 \\ \dfrac{b_y}{b_x} = \dfrac{y_1 - y_0}{x_1 - x_0} \\ \dfrac{2a_y + b_y}{2a_x + b_x} = \dfrac{y_2 - y_1}{x_2 - x_1} \end{cases}$$

$$f'(x) = \frac{y'(t)}{x'(t)} = \frac{2a_y t + b_y}{2a_x t + b_x}$$

抛物线的参数方程的系数：

$$\begin{cases} c_x = x_0 \\ c_y = y_0 \\ b_x = 2(x_1 - x_0) \\ b_y = 2(y_1 - y_0) \\ a_x = x_2 - 2x_1 + x_0 \\ a_y = y_2 - 2y_1 + y_0 \end{cases}$$

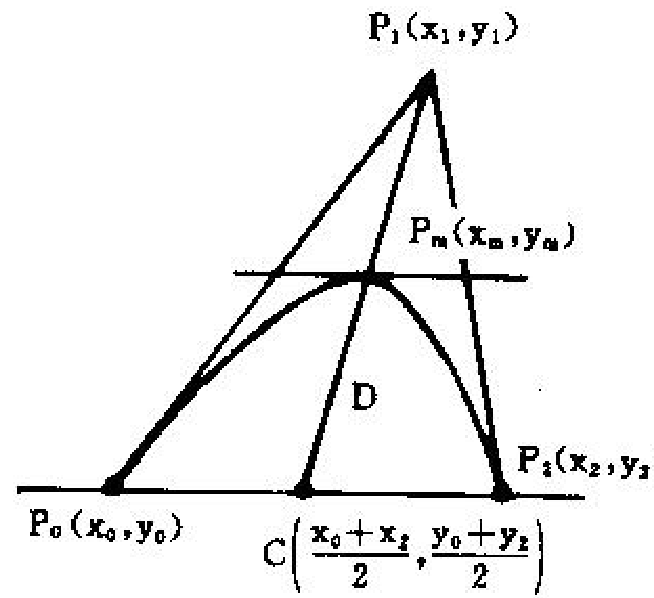
抛物线的重要性质：

1. 曲线在 t = 1/2 处的切线平行于 P_0P_2。

2. P_m点为 P_1C 直线的中点。

采用P_0、P_m、P_2三点构造抛物线时的参数方程的系数：

$$\begin{cases} c_x = x_0 \\ c_y = y_0 \\ b_x = 4x_m - x_2 - 3x_0 \\ b_y = 4y_m - y_2 - 3y_0 \\ a_x = 2(x_2 - 2x_m + x_0) \\ a_y = 2(y_2 - 2y_m + y_0) \end{cases}$$



结论：

P_0、P_1、P_2与P_0、P_m、P_2所构成的抛物线是等价的，二者确定参数方程系数的公式不同，后者产生的曲线通过给定的三点。

以p_0，p_m，p_2作为抛物线上的点，得抛物线参数方程为：

$$\begin{cases} x(t)=2(x_2-2x_m+x_0)t^2+(4x_m-x_2-3x_0)t+x_0 \\ y(t)=2(y_2-2y_m+y_0)t^2+(4y_m-y_2-3y_0)t+y_0 \end{cases}$$

$$\rightarrow \begin{cases} x(t)=(2t^2-3t+1)x_0+(-4t^2+4t)x_m+(2t^2-t)x_2 \\ y(t)=(2t^2-3t+1)y_0+(-4t^2+4t)y_m+(2t^2-t)y_2 \end{cases}$$

$$\rightarrow \overrightarrow{p(t)}=(2t^2-3t+1)\overrightarrow{p_0}+(-4t^2+4t)\overrightarrow{p_m}+(2t^2-t)\overrightarrow{p_2}$$ ，$(0\le t\le 1)$

抛物线的绘制---参数插值

无论是给定P_0、P_1、P_2或P_0、P_m、P_2，都可根据前面的公式求得抛物线参数方程的系数。

对于抛物线参数方程：

$$\begin{cases} x(t)=a_x t^2 + b_x t + c_x \\ y(t)=a_y t^2 + b_y t + c_y \end{cases} \quad (0 \le t \le 1)$$

只须将参数 **t** 从0到1按一定的步长递增，就可得到（即插入）一组对应的x、y坐标，再将每两个相邻的坐标点之间用小直线段连起来，便可画出整条抛物线。

3.3 自由曲线

一. 概述

曲线：规则曲线——可用曲线方程式表示的曲线。

不规则曲线——不能确切给出描述整个曲线的方程，而是由从实际测量中得到的一系列离散数据点采用曲线拟合的方法来逼近的。这类曲线也称之为自由曲线。

曲线的表示方法：

1. 直角坐标曲线　显式　$y = f(x)$　　隐式　$f(x, y) = 0$
2. 极坐标曲线　　$P = \rho(\theta)$
3. 参数坐标曲线　$x = x(t)$；$y = y(t)$　参变量的规格化

曲线的绘制方法：用很多短直线段来逼近曲线。曲线上点的数量取多少，直线段取多长，取决于绘制曲线的精度要求和图形输出设备的精度。

型值点：是指通过测量或计算得到的曲线上少量描述曲线几何形状的数据点。

控制点：是指用来控制或调整曲线形状的特殊点，曲线本身不一定通过控制点。

插值和逼近：这是曲线设计中的两种不同方法。插值设计方法要求建立的曲线数学模型，严格通过已知的每一个型值点。而逼近设计方法建立的曲线数学模型只是近似地接近已知的型值点。

拟合：是指在曲线的设计过程中，用插值或逼近的方法使生成的曲线达到某些设计要求。

连续性：

C^0连续（0阶参数连续）——前一段曲线的终点与后一段曲线的起点相同。

C^1连续（一阶参数连续）—— 两相邻曲线段的连接点处有相同的一阶导数。

C^2连续（二阶参数连续）—— 两相邻曲线段的连接点处有相同的一阶导数和二阶导数。

自由曲线的绘制方法：

在拟合生成曲线的众多方法中，一般总要选择一种简单一些的曲线，作为拟合生成其它曲线的基本曲线，然后对这种基本曲线作一些适当的数学处理，来完成完整的拟合曲线。

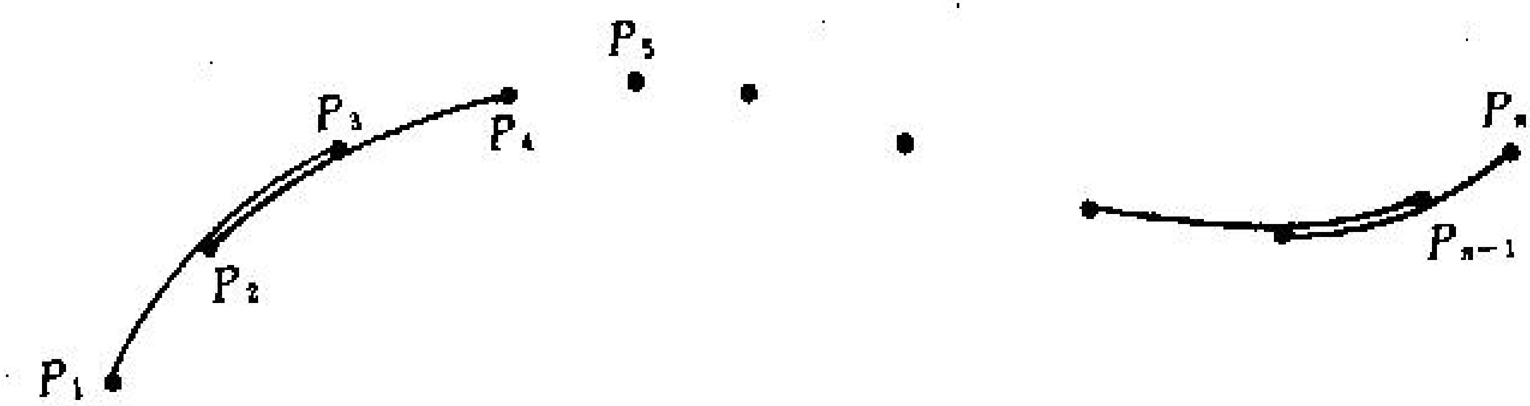
自由曲线 $\longrightarrow$ 样条曲线 $\longrightarrow$ 小直线段

二． 抛物线参数样条曲线

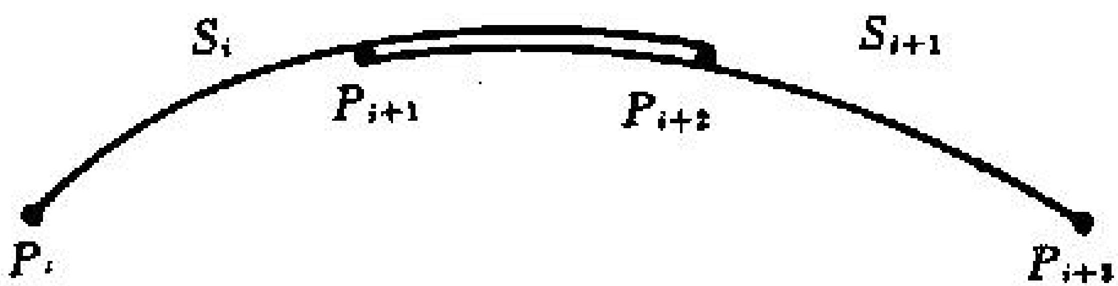
根据给定的型值点列，以抛物线作为基本曲线拟合生成自由曲线。

特点：采用插值方法生成，曲线通过每个型值点。

给定型值点列 P_i（ $i=1, 2, \ldots\ldots, n$ ），按抛物线的参数拟合方法，每经过相邻三点可作一段抛物线，共可作出 $n-2$ 条。



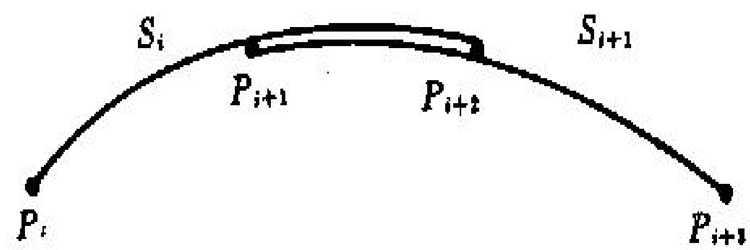
一般情况下，每两段曲线之间的搭接区间，两段抛物线是不可能重合的。



但对于拟合曲线来说，整个型值点列必须用一条光滑的曲线连接起来。

解决的办法：将两段曲线之间的搭接区间采用**加权合成**的方法合成一条曲线。也就是说，由 P_i 、P_{i+1} 、P_{i+2} 、P_{i+3} 这四个型值点采用加权合成的方法可以确定 P_{i+1}和 P_{i+2} 之间的一段曲线。

抛物线样条曲线之加权合成



$$\overrightarrow{p_{i+1}(t)} = (1-T)\overrightarrow{s_i(t_i)} + T\overrightarrow{s_{i+1}(t_{i+1})}$$

令t=t_{i+1}，则T=2t，t_i=t+0.5，得

$$\overrightarrow{p_{i+1}(t)} = (1-2t)\overrightarrow{s_i(t+0.5)} + 2t\overrightarrow{s_{i+1}(t)}$$

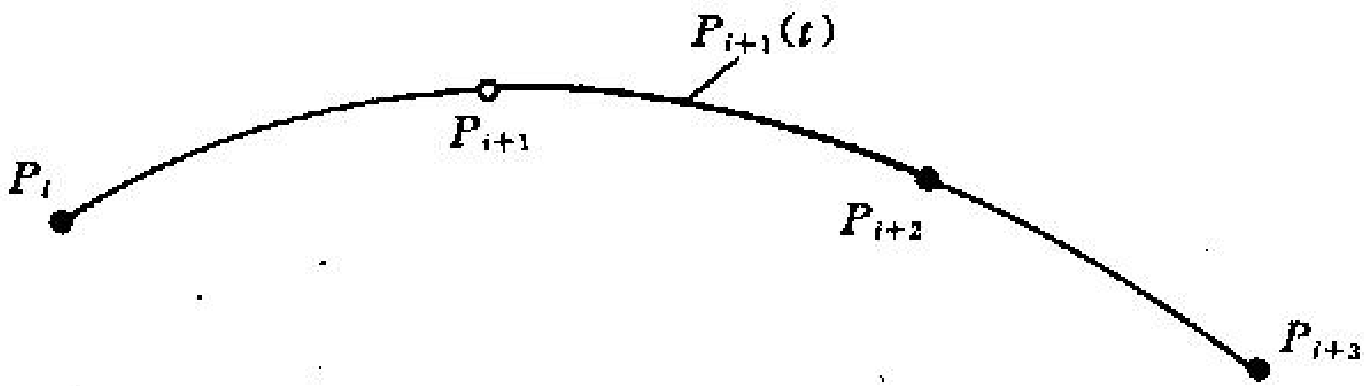
又 $\overrightarrow{s_i(t_i)} = (2t_i^{\ 2}-3t_i+1)\overrightarrow{p_i} + (-4t_i^{\ 2}+4t_i)\overrightarrow{p_{i+1}} + (2t_i^{\ 2}-t_i)\overrightarrow{p_{i+2}}$

$$\overrightarrow{s_{i+1}(t_{i+1})} = (2t_{i+1}^2-3t_{i+1}+1)\overrightarrow{p_{i+1}} + (-4t_{i+1}^2+4t_{i+1})\overrightarrow{p_{i+2}} + (2t_{i+1}^2-t_{i+1})\overrightarrow{p_{i+3}}$$

得

$$\overrightarrow{p_{i+1}(t)} = (-4t^3+4t^2-t)\overrightarrow{p_i} + (12t^3-10t^2+1)\overrightarrow{p_{i+1}} + (-12t^3+8t^2+t)\overrightarrow{p_{i+2}}$$

$$+(4t^3-2t^2)\overrightarrow{p_{i+3}} \quad ,(0 \le \mathrm{t} \le 0.5)$$



结论：

对于给定的型值点列 P_i（ i= 1，2，.....，n ），从 P_1 开始依次每取四个型值点即可画出一段曲线，直到 P_n 为止。

按这样的方法，在P_i（ i= 1，2，.....，n ）个型值点列中只能得到n–3 段曲线，但 n 个型值点列之间应有 n–1 个区段。

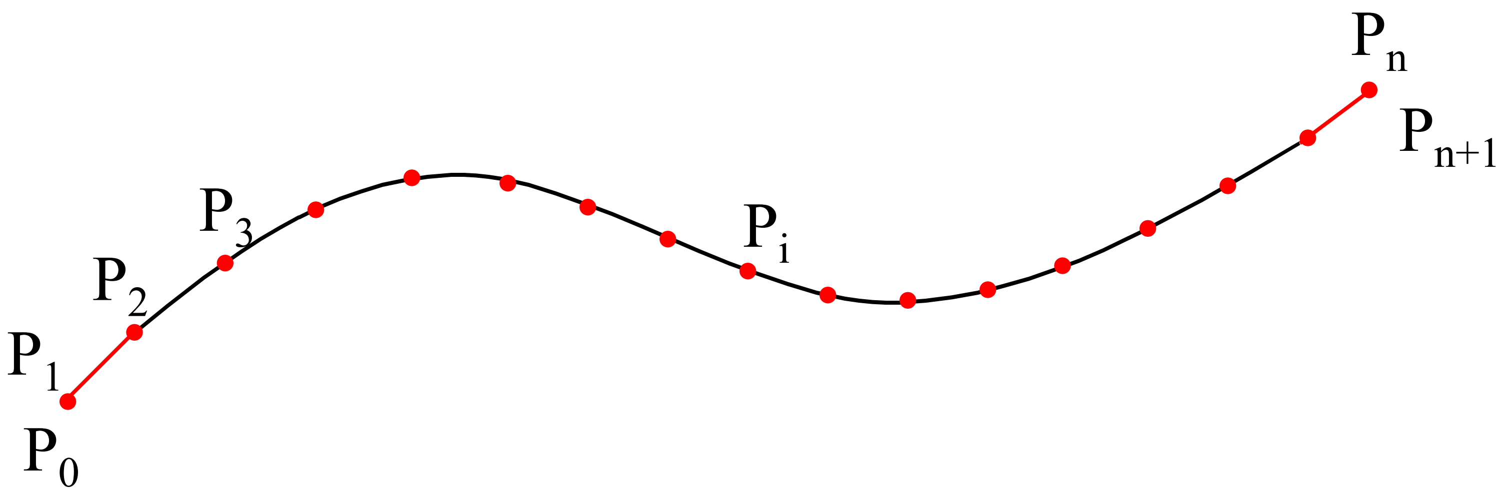
如何得到首、末两个区段的曲线呢？

解决的办法：

添加“端点条件”(也称“边界条件”)。

其中最简单的一种称为“自由端条件”，即在首、末两端各添加一个辅助点P_0 和P_{n+1}，并使 $P_0=P_1$, $P_{n+1}=P_n$ 。.

这种方法适用于对曲线的两端没有什么特殊要求的情况。



三. Hermite 曲线

一条三次参数曲线的代数形式是：

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \qquad (0 \le t \le 1) \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases}$$

上式写成矢量形式是：

$$P(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d} \qquad (0 \leqslant t \leqslant 1)$$

其中P(t)表示曲线上任意一点的位置矢量，其分量对应于直角坐标系中该点的坐标； a 、 b 、 c 、 d是代数系数矢量。

上式写成矩阵形式是：

$$\mathbf{P}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

其 x 方向上的分量可表示为：

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x$$

令： $T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$

$C_x = \begin{bmatrix} a & b & c & d \end{bmatrix}_x^T$

则： $x(t) = T \cdot C_x$

且： $x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \cdot C_x$

Hermite曲线是给定曲线段的两个端点坐标P_0 、 P_1以及两端点处的切线矢量R_0 、 R_1来描述曲线的。即：

$T = [t^3 \quad t^2 \quad t \quad 1]$

$C_x = [a \quad b \quad c \quad d]_x^T$

$x(t) = T \cdot C_x$

$x'(t) = [3t^2 \quad 2t \quad 1 \quad 0] \cdot C_x$

$x(0) = P_{0x}$，　$x(1) = P_{1x}$，

$x'(0) = R_{0x}$，　$x'(1) = R_{1x}$

将上述边界条件代入前式，得：

$P_{0x} = [0 \quad 0 \quad 0 \quad 1] \cdot C_x$　　$P_{1x} = [1 \quad 1 \quad 1 \quad 1] \cdot C_x$

$R_{0x} = [0 \quad 0 \quad 1 \quad 0] \cdot C_x$　　$R_{1x} = [3 \quad 2 \quad 1 \quad 0] \cdot C_x$

并可用矩阵形式表示为：

$$\begin{pmatrix} p_0 \\ P_1 \\ R_0 \\ R_1 \end{pmatrix}_x = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \cdot C_x$$

将上式的两端分别乘以一个4×4矩阵的逆阵，可得：

$$C_x = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ R_0 \\ R_1 \end{pmatrix}_x$$

令：

$$M_h = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$ 为Hermite矩阵，常数

$G_h =$ 〔P0　P1　R0　R1〕^{T} 为Hermite几何矢量

则：　$C_x = M_h \cdot G_{hx}$

式　$x(t) = T \cdot C_x$　可改写为：　$x(t) = T \cdot M_h \cdot G_{hx}$

三次参数曲线的矢量形式可改写为：

$$\mathbf{P}(t) = T \cdot M_h \cdot G_h \qquad (0 \leq t \leq 1)$$

上式中的 $T \cdot M_h$ 称为调和函数。

若令其为 $F_h(t)$，则各分量可表示为：

$$F_{h1}(t) = 2t^3 - 3t^2 + 1$$

$$F_{h2}(t) = -2t^3 + 3t^2$$

$$F_{h3}(t) = t^3 - 2t^2 + t$$

$$F_{h4}(t) = t^3 - t^2$$

这样，当给定初始条件G_h后，Hermite曲线可表示成：

$$\mathbf{P}(t) = F_{h1}(t)\mathbf{P}_0 + F_{h2}(t)\mathbf{P}_1 + F_{h3}(t)\mathbf{R}_0 + F_{h4}(t)\mathbf{R}_1$$

式中P_0和P_1为曲线两端点的位置矢量，R_0和R_1为曲线两端点处的切线矢量。

利用上式，便可绘制出一段Hermite曲线。

四. 三次参数样条曲线

Hermite曲线要求给出端点处的切线矢量，给使用带来不便，但它是三次参数样条曲线的基础。

若有一组离散点列$P_1, P_2, \ldots, P_{i-1}, P_i, P_{i+1}, \ldots, P_n$,要求用一系列Hermite曲线段，通过这些点列，构成一条三次参数样条曲线。

n 个点可绘制出 n – 1 段Hermite曲线，其中第 i 段曲线的起点和终点分别为 P_i 和 P_{i+1}；第 i+1 段曲线为P_{i+1}和P_{i+2} 。

若要求两曲线连接处达到C^2, 通过推导，可得如下关系式：

$$P_i' + 4P_{i+1}' + P_{i+2}' = 3(P_{i+2} - P_i)$$

Hermite曲线：

$$P(t)=F_{h1}(t)P_0+F_{h2}(t)P_1+F_{h3}(t)P_0'+F_{h4}(t)P_1'$$

二阶求导：

$$P''(t)=F''_{h1}(t)P_0+F''_{h2}(t)P_1+F''_{h3}(t)P_0'+F''_{h4}(t)P_1'$$

其中：

$$F''_{h1}(t)=12t-6，\quad F''_{h2}(t)=-12t+6$$

$$F''_{h3}(t)=6t-4，\quad F''_{h4}(t)=6t-2$$

对于第i段曲线，起点P_i，终点P_{i+1}

t=0，$P''_i=-6P_i+6P_{i+1}-4P_i'-2P_{i+1}'$

t=1，$P''_{i+1}=6P_i-6P_{i+1}+2P_i'+4P_{i+1}'$

对于第i+1段曲线，起点P_{i+1}，终点P_{i+2}

t=0，$P''_{i+1}=-6P_{i+1}+6P_{i+2}-4P_{i+1}'-2P_{i+2}'$

t=1，$P''_{i+2}=6P_{i+1}-6P_{i+2}+2P_{i+1}'+4P_{i+2}'$

$\because$ C^2连续

$\therefore$ $6P_i - 6P_{i+1} + 2P_i' + 4P_{i+1}' = -6P_{i+1} + 6P_{i+2} - 4P_{i+1}' - 2P_{i+2}'$

可得：$P_i' + 4P_{i+1}' + P_{i+2}' = 3(P_{i+2} - P_i)$

同样，对于第 i +1和 i+2 段曲线，可得关系式：

$$P_{i+1}' + 4P_{i+2}' + P_{i+3}' = 3(P_{i+3} - P_{i+1})$$

依此类推，对于n 个点，可以得到 n – 2 个类似的方程。

$$\begin{cases} P_1' + 4P_2' + P_3' = 3(P_3 - P_1) \\ P_2' + 4P_3' + P_4' = 3(P_4 - P_2) \\ \qquad \cdots\cdots \\ P_{n-2}' + 4P_{n-1}' + P_n' = 3(P_n - P_{n-2}) \end{cases}$$

但这组联立方程中有 n 个未知数，为求解，必须再给出两个边界条件。常用的边界条件有自由端、夹持端和抛物端等。

以自由端为例，这种情况下，两端点处的二阶导数为零，即 $P_1'' = P_n'' = 0$

自由端三次参数样条曲线的矩阵表示式为：

$$
\begin{bmatrix}
2 & 1 & 0 & 0 & \ldots & & 0 \\
1 & 4 & 1 & 0 & \ldots & & 0 \\
0 & 1 & 4 & 1 & 0\ldots & & 0 \\
\multicolumn{7}{c}{\cdots\cdots\cdots\cdots\cdots\cdots} \\
\multicolumn{7}{c}{\cdots\cdots\cdots\cdots\cdots\cdots} \\
0 & 0 & \ldots & \ldots & 1 & 4 & 1 \\
0 & 0 & \ldots & \ldots & 0 & 1 & 2
\end{bmatrix}
\cdot
\begin{bmatrix}
P_1' \\ P_2' \\ P_3' \\ . \\ . \\ P_{n-1}' \\ P_n'
\end{bmatrix}
=
\begin{bmatrix}
3(P_2 - P_1) \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \cdots\cdots \\ \cdots\cdots \\ 3(P_n - P_{n-2}) \\ 3(P_n - P_{n-1})
\end{bmatrix}
$$

解方程组，可得各型值点处的切线向量 $P_i'\ (1 \le i \le n)$

五. Bezier曲线

Bezier曲线通过一组多边折线的各顶点唯一的定义出来。

在多边折线的各顶点中，只有第一点和最后一点在曲线上，其余的顶点则用来定义曲线的导数，阶次和形状。第一条边和最后一条边分别和曲线在起点和终点处相切，曲线的形状趋于多边折线的形状，改变多边折线的顶点位置和曲线形状的变化有着直观的联系。多边折线称为特征多边形，其顶点称为控制点。

Bezier曲线的参数方程：

$$Q(t)=\sum_{i=0}^{n} P_i B_{i,n}(t) \quad ,(0 \leqslant t \leqslant 1)$$

P_i(i = 0, 1, 2,…., n)是空间给定的 n+1个点的位置向量，也称控制点，它们构成了控制Bezier曲线形状的特征多边形。

$B_{i,n}(t)$为**Bernstain基函数**。

1. 二次Bezier曲线

当n＝2时，上式即为二次Bezier曲线表达式。二次Bezier曲线有三个控制点，它是一条经过P_0和P_2两个控制点的抛物线。

它的矩阵表示形式如下：

$$\mathbf{Q}(t)=[t^2 \quad t \quad 1]\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} \qquad (0\le t\le 1)$$

若将上式中的向量 $\mathbf{P}_0$、$\mathbf{P}_1$、$\mathbf{P}_2$分解为二维平面上的 x 及 y 方向分量，就可得到二次 Bezier 曲线的参数式：

$$\begin{cases} x(t)=a_x t^2+b_x t+c_x \\ y(t)=a_y t^2+b_y t+c_y \end{cases} \qquad (0\le t\le 1)$$

式中系数分别为：

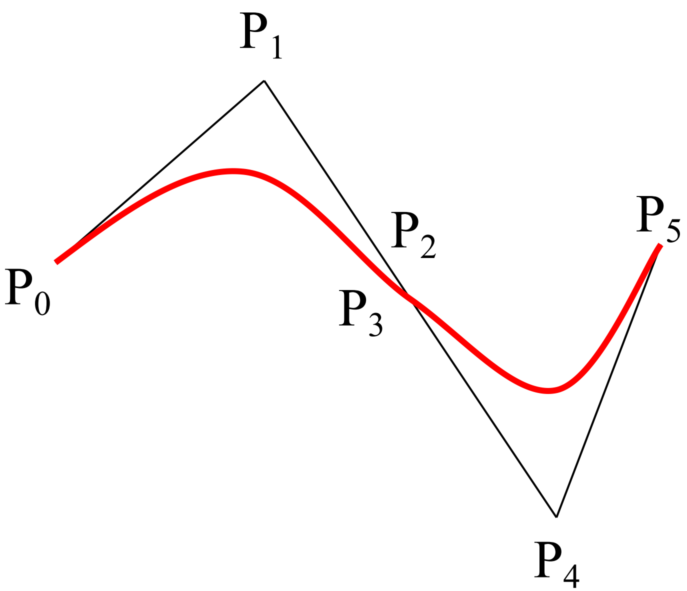
$$a_x=x_2-2x_1+x_0, \quad b_x=2(x_1-x_0), \quad c_x=x_0$$

$$a_y=y_2-2y_1+y_0, \quad b_y=2(y_1-y_0), \quad c_y=y_0$$

二段二次Bezier曲线在满足一定条件的情况下可以达到C^1连续：

$P_2 = P_3$，

P_4 应在 P_1P_2 的延长线上。



2. 三次Bezier曲线

三次Bezier曲线的矩阵表示形式如下：

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad (0 \le t \le 1)$$

分解后的参数式为：

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \end{cases} \quad (0 \le t \le 1)$$

式中系数分别为：

$a_x = -x_0 + 3x_1 - 3x_2 + x_3$ $\quad$ $a_y = -y_0 + 3y_1 - 3y_2 + y_3$

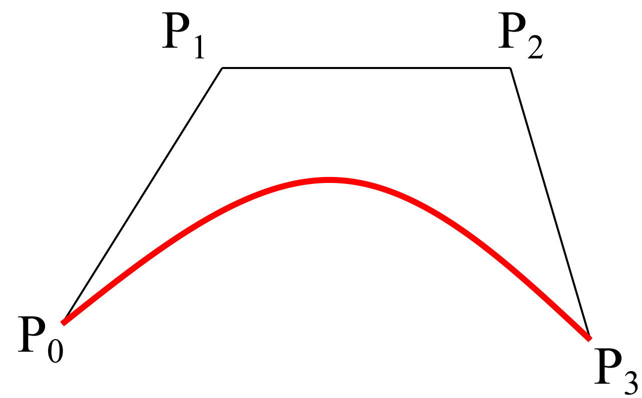
$b_x = 3x_0 - 6x_1 + 3x_2$ $\quad$ $b_y = 3y_0 - 6y_1 + 3y_2$

$c_x = -3x_0 + 3x_1$ $\quad$ $c_y = -3y_0 + 3y_1$

$d_x = x_0$ $\quad$ $d_y = y_0$

三次Bezier曲线的端点特性：

曲线经过首、末两个控制点，且与特征多边形的首、末两条边相切。



三次Bezier曲线段的连续性：

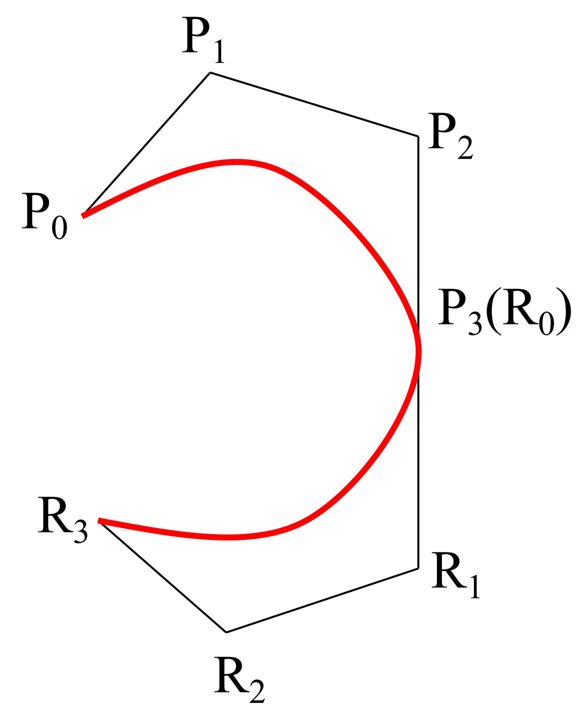
要使两段三次Bezier曲线达到C^1连续的充要条件是：

P_2 、 $P_3=R_0$ 、 R_1 三点共线

要使两段三次Bezier曲线达到C^2连续的充要条件是，要在C^1连续的前提下再增加两个条件：

1. 在连接处两曲线的密切平面重合。

2. 在连接处两曲线的曲率相等。



六. B样条曲线

B样条曲线是Bezier曲线的拓广，它是用B样条基函数代替了Bezier曲线表达式中的Bernstain基函数。

在空间给定n＋1个点的位置向量P_i（i=0, 1, 2, ……n, n>=k)，则称参数曲线

$$Q(t)=\sum_{i=0}^{n}P_iN_{i,k}(t) \qquad (0\le t\le 1)$$

为k阶(或k－1次)的B样条曲线。其中$N_{i,k}(t)$为**B样条基函数**。给定的n＋1个点为B样条曲线的控制顶点，由其构成的多边折线称**B特征多边形**。

1. 二次B样条曲线

二次B样条曲线的矩阵表示形式如下：

$$Q(t)=[t^2\ \ t\ \ 1]\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

分解后的参数式为：

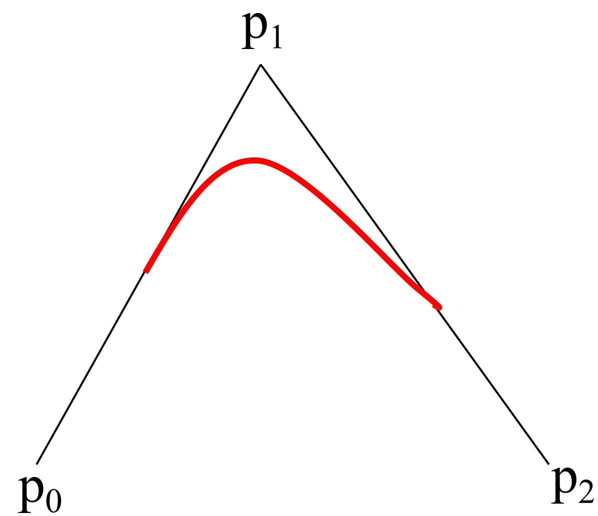
$$\begin{cases} x(t)= a_x t^2 + b_x t + c_x \\ y(t)= a_y t^2 + b_y t + c_y \end{cases}$$

式中系数分别为：

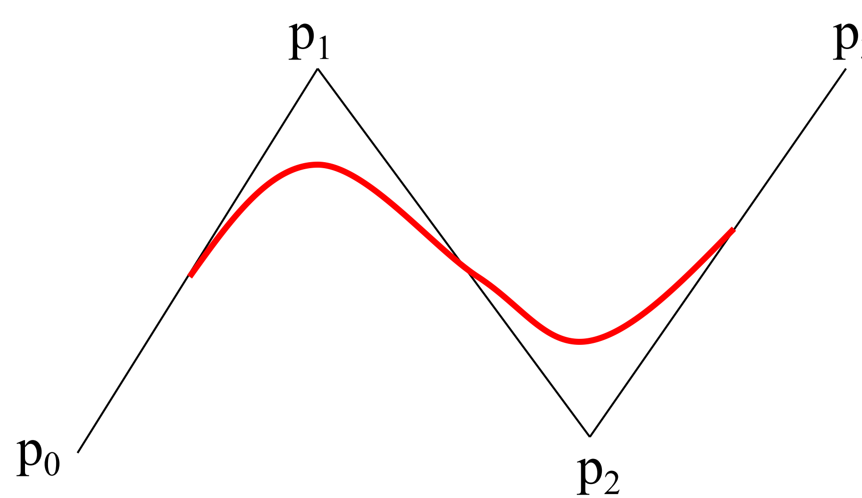
$a_x=(x_0-2x_1+x_2)/2$ $b_x=x_1-x_0$ $c_x=(x_0+x_1)/2$

$a_y=(y_0-2y_1+y_2)/2$ $b_y=y_1-y_0$ $c_y=(y_0+y_1)/2$

二次B样条曲线的端点特性与Bezier曲线不同，它是以二次B特征多边形的二边上的中点为其起点和终点，并在端点处与二边相切。



由三个控制顶点(P_0、P_1、P_2)确定的一条二次B样条曲线是一条抛物线，如果再增加一个控制顶点P_3，就可由P_1、P_2、P_3三个控制顶点生成第二条二次B样条曲线。由于第一条二次B样条曲线的终点就是第二条二次B样条曲线的起点，而且它们有一条公共的切线P_1P_2，所以二条二次B样条曲线在切点衔接处达到C^1连续。



2. 三次B样条曲线

三次B样条曲线的矩阵表示形式如下：

$$Q(t)=[t^3\ \ t^2\ \ t\ \ 1]\frac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix} \qquad (0\le t\le 1)$$

分解后的参数式为：

$$\begin{cases} x(t)=a_xt^3+b_xt^2+c_xt+d_x \\ y(t)=a_yt^3+b_yt^2+c_yt+d_y \end{cases} \qquad (0\le t\le 1)$$

式中系数分别为：

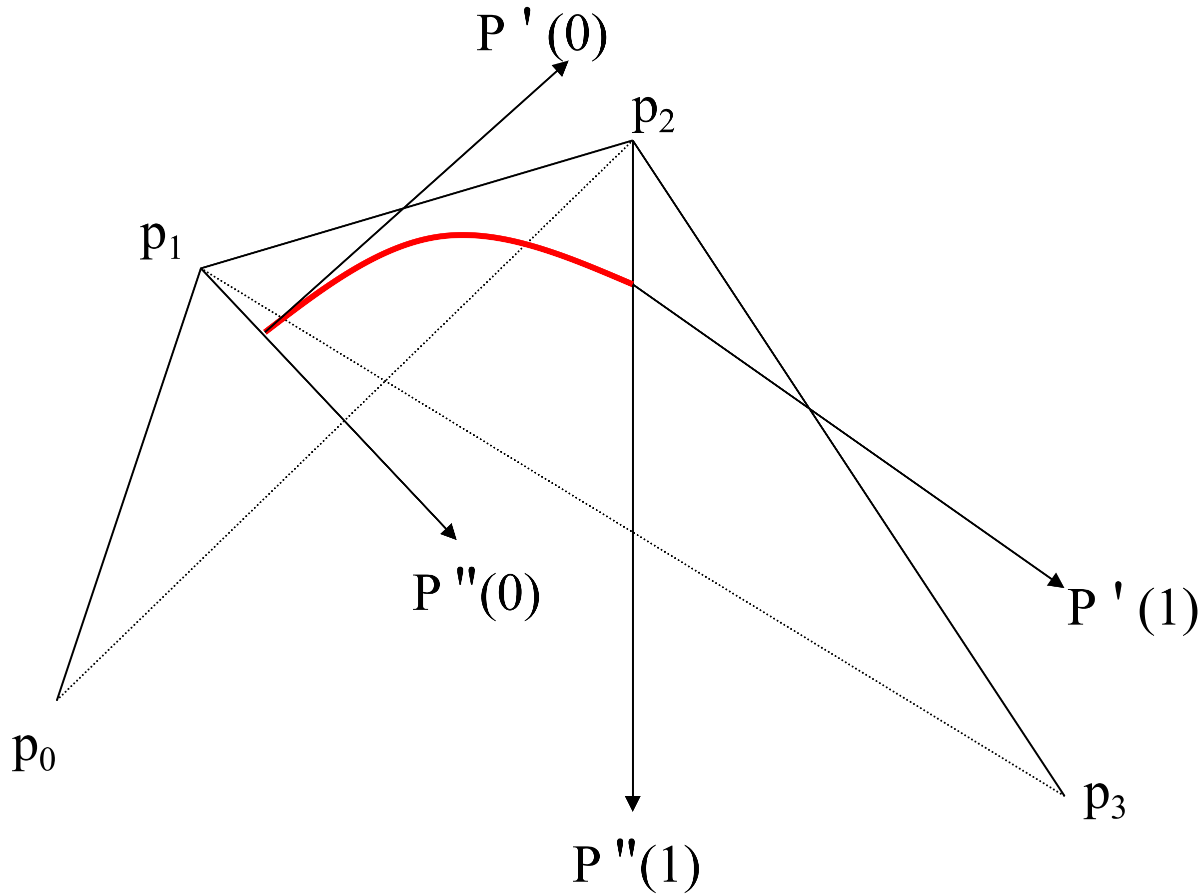
$a_x=-(x_0-3x_1+3x_2-x_3)/6$　　$a_y=-(y_0-3y_1+3y_2-y_3)/6$

$b_x=(x_0-2x_1+x_2)/2$　　$b_y=(y_0-2y_1+y_2)/2$

$c_x=-(x_0-x_2)/2$　　$c_y=-(y_0-y_2)/2$

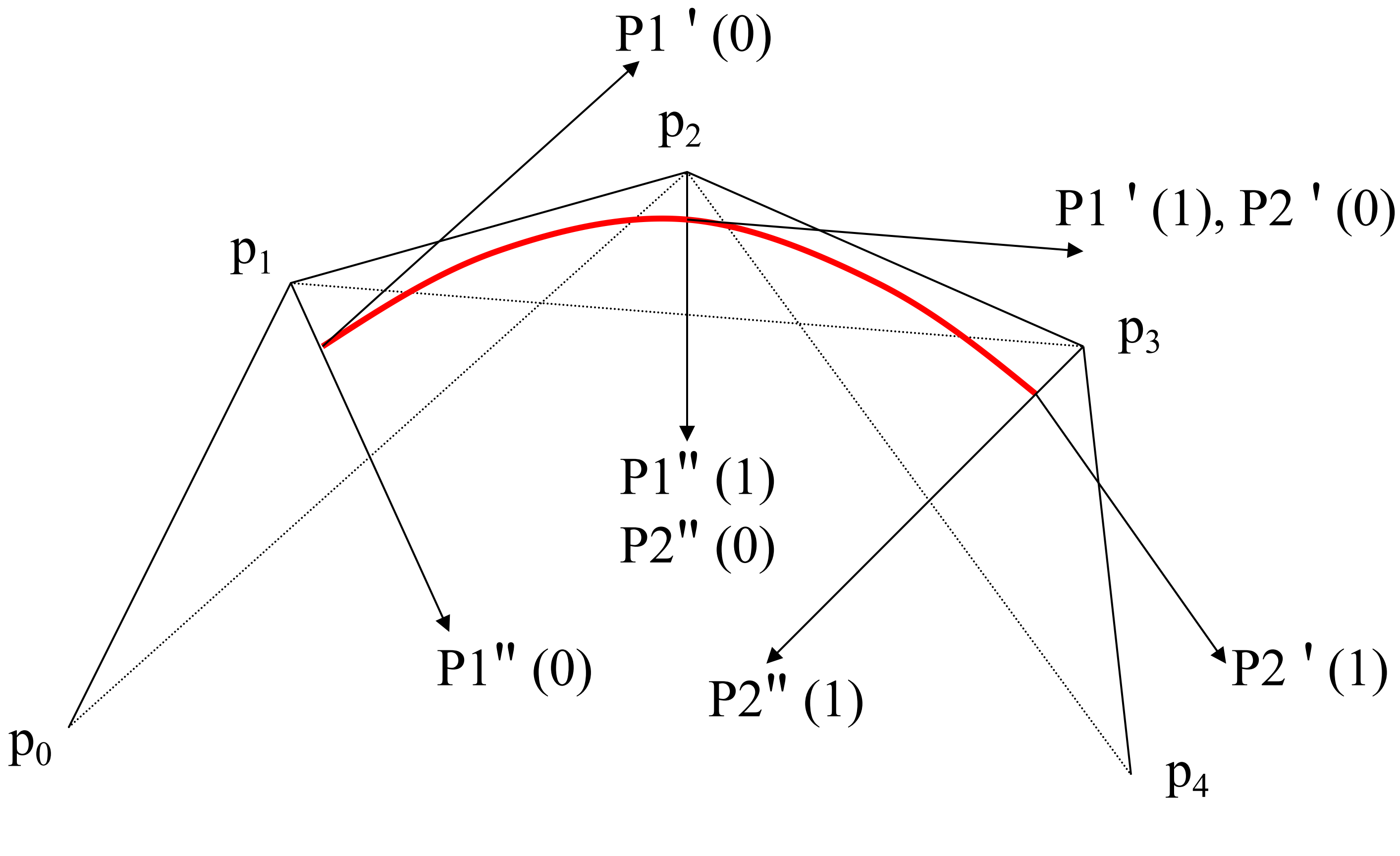
$d_x=(x_0+4x_1+x_2)/6$　　$d_y=(y_0+4y_1+y_2)/6$

三次B样条曲线的端点特性：



三次B样条曲线的连续性：

在已有的三次B样条曲线的基础上，增加一个控制点，就可相应地增加一段B样条曲线，并自然地达到 C^2 连续。



3. 三次B样条曲线的绘制技巧

在三次B样条曲线的设计中，常会遇到以下几种情况：

* 要在某处使曲线段与特征多边形相切
* 要在某处使曲线形状出现一个尖点或通过某一个角点
* 要在某处使曲线出现一个拐点(制图中所谓的反向弧切接)
* 要在某处使曲线形状中切接入一段直线 ……

运用**角点重叠**和**角点共线**的技巧。

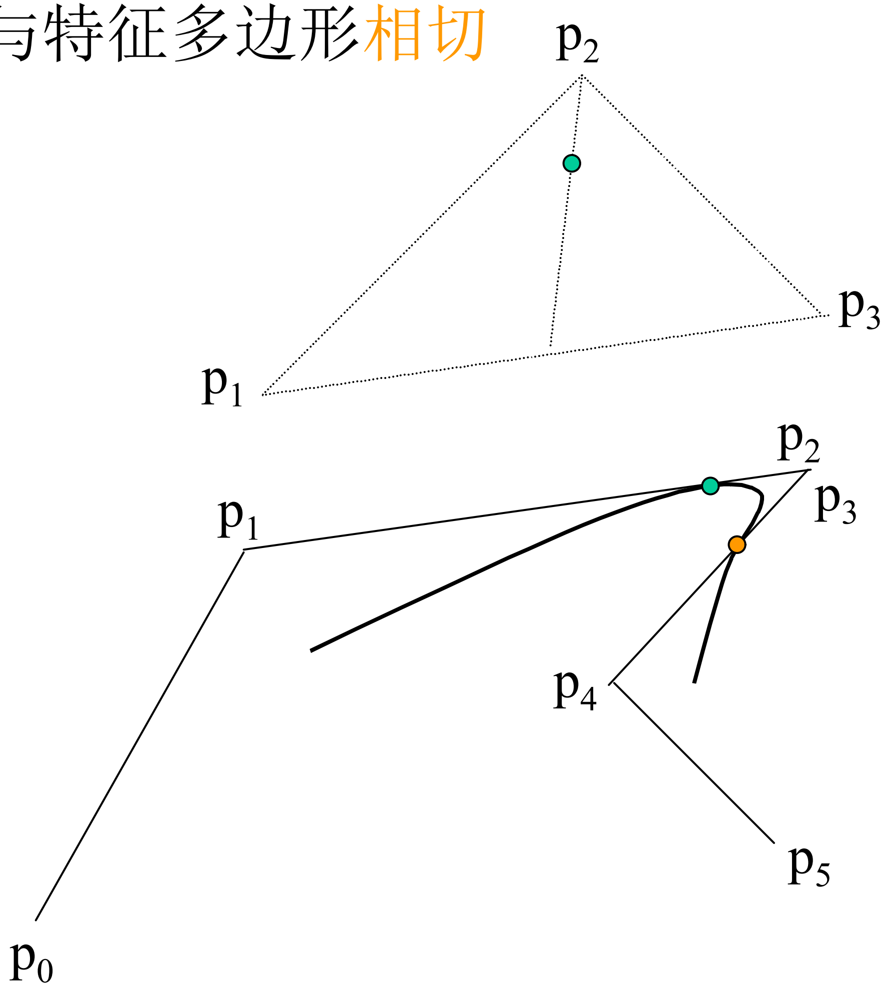
例如：

二重角点　三角点共线

三重角点　四角点共线

二重角点

曲线段与特征多边形相切

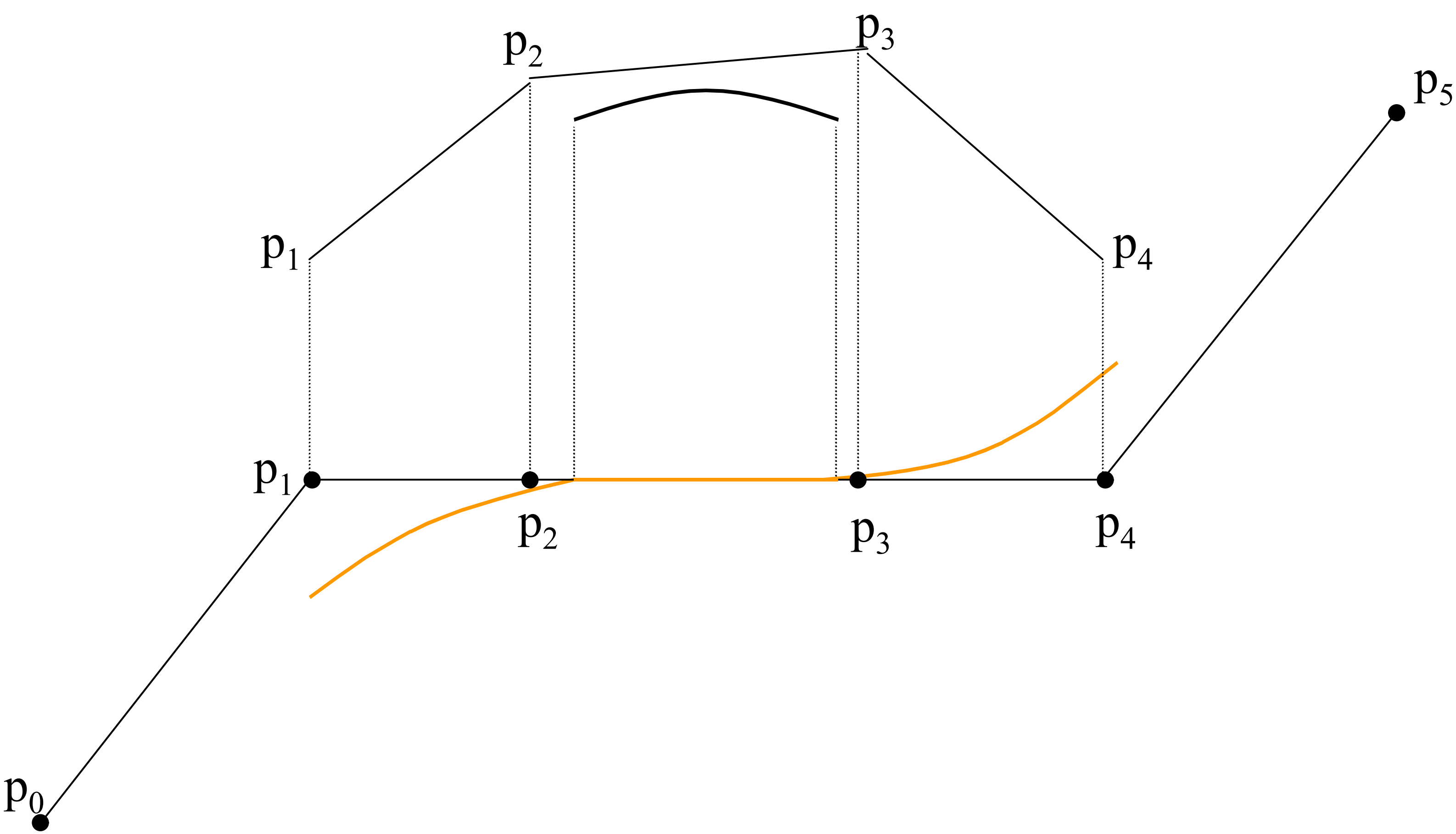


三角点共线

曲线出现一个拐点

四角点共线

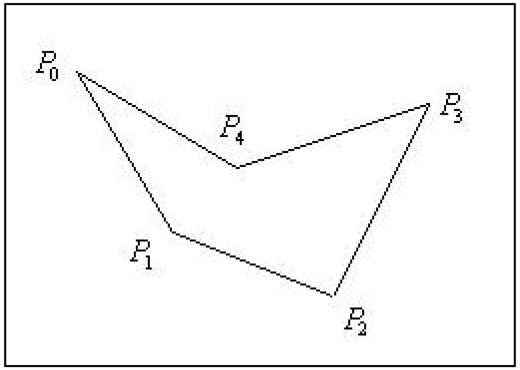
在曲线中切接入一段直线

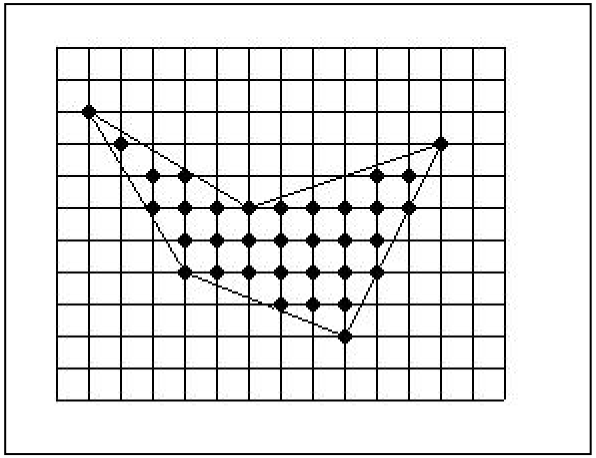


3.4 多边形的扫描转换与区域填充

一. 多边形的扫描转换

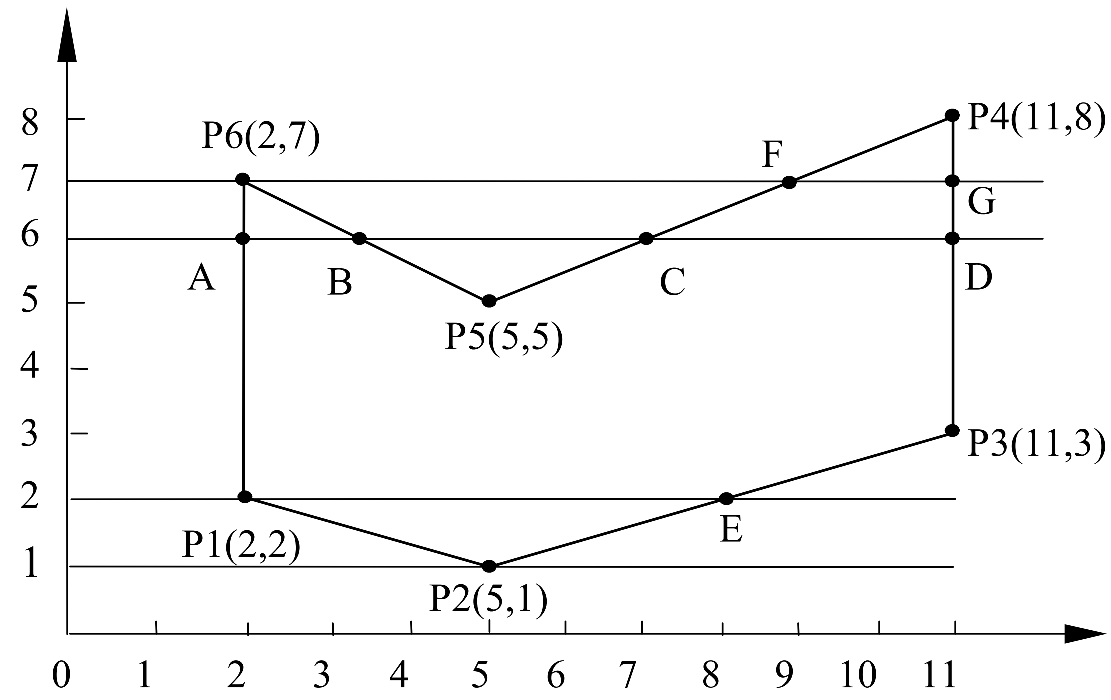
- 多边形有两种重要的表示方法：顶点表示和点阵表示。
- 多边形的扫描转换:把多边形的顶点表示转换为点阵表示。





- 扫描线算法
  - 基本思想：

    按扫描线顺序，计算扫描线与多边形的相交区间，再用要求的颜色显示这些区间的象素，即完成填充工作。
  - 对于一条扫描线填充过程可以分为四个步骤：
    - 求交
    - 排序
    - 配对
    - 填色



一个多边形与若干扫描线

- 求交

设多边形某条边所在的直线方程为：$ax+by+c=0$

计算扫描线$y = y_i$与该边的交点：

纵坐标：y_i，横坐标：$x_i = -(by_i +c)/a$

增量计算法：

计算扫描线$y = y_{i+1}$与该边的交点：

纵坐标：y_{i+1}

横坐标：$x_{i+1} = -(by_{i+1} +c)/a = -[b(y_i +1)+c]/a$

$$= -(by_i +c)/a-b/a$$

$$= x_i - b/a$$

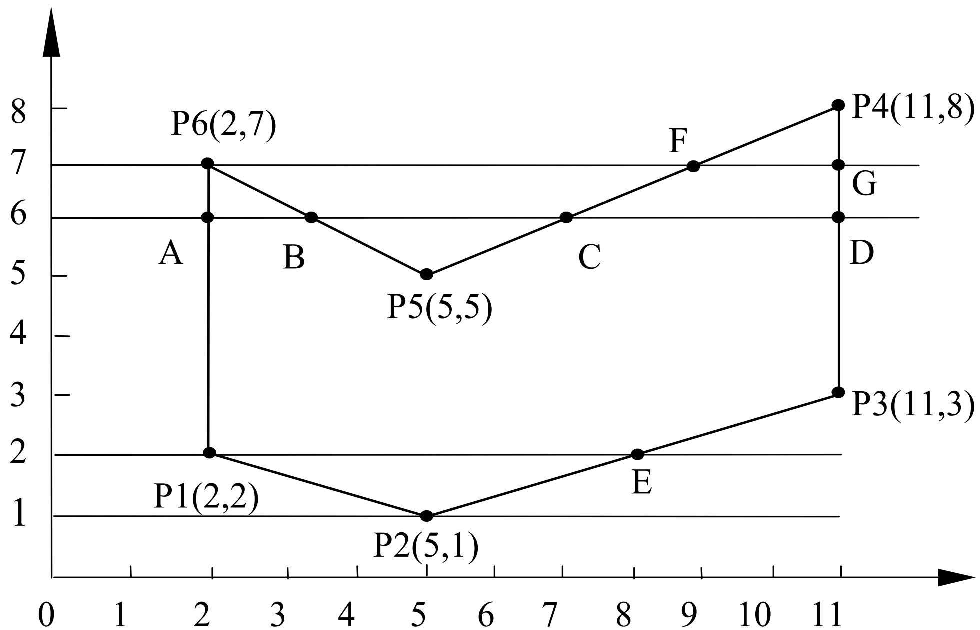
$$= x_i + \Delta x$$

- 排序

  将所有交点按照x坐标由小到大进行排序。

- 配对

  相邻交点构成一个区间，每个交点只配对一次。

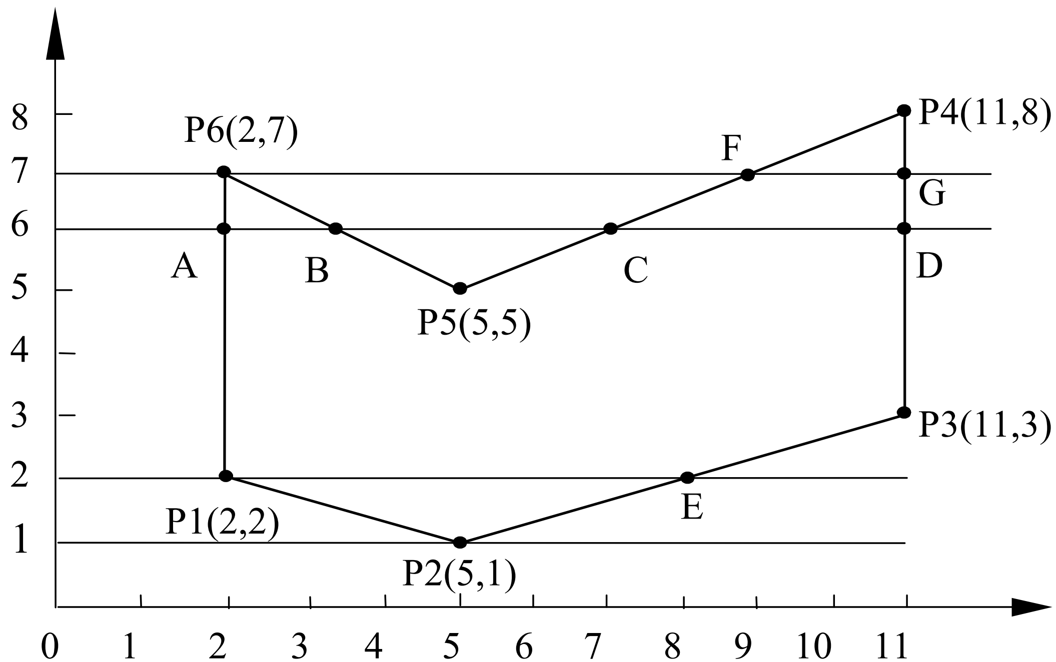


- 填色

  把每个区间内的象素用指定颜色显示。

- 交点的取舍

问题：扫描线与多边形的顶点相交时，交点如何计数？



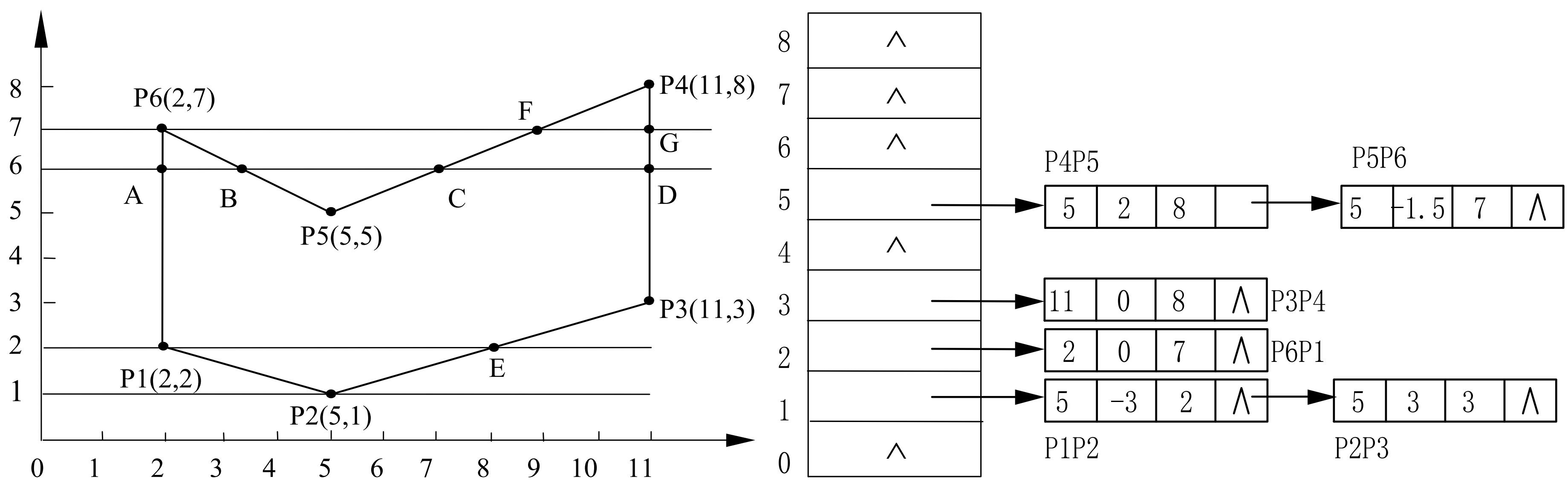
交点取舍方法：检查顶点所在两条边的另外两个端点，若位于同侧，则顶点作为两个交点，否则顶点作为一个交点。

–数据结构

1. 边表（ET）——桶（向量＋链表）

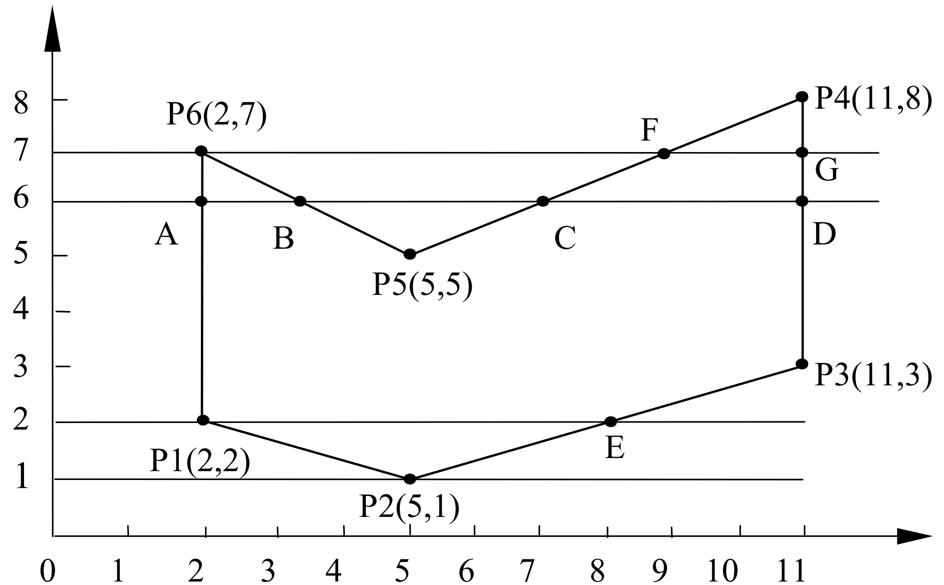
向量：存放扫描线信息。

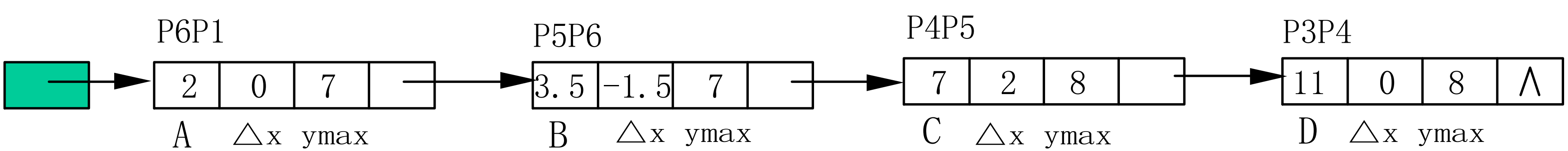
链表：存放从某条扫描线开始的多边形的边信息，多边形的边根据其端点中最小的y坐标值放入相应的桶内，结点信息包括该边y值较小的端点的x坐标、Δx、端点中较大的y坐标。



2. 活化边表（有效边表，AET）

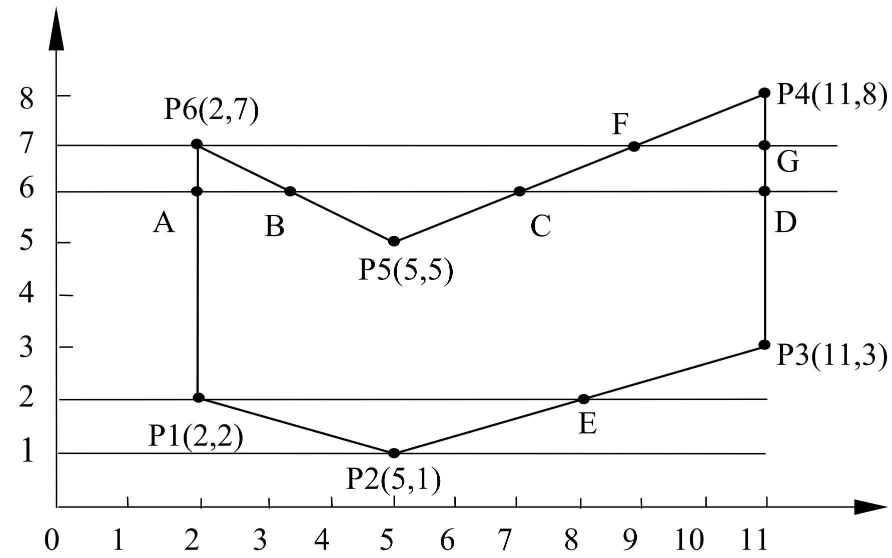
活化边：与当前处理的扫描线相交的边。按与当前扫描线交点x坐标递增的顺序存放在一个链表中，结点内容包括：当前扫描线与边的交点的x坐标、△x、该边端点中较大的y坐标。





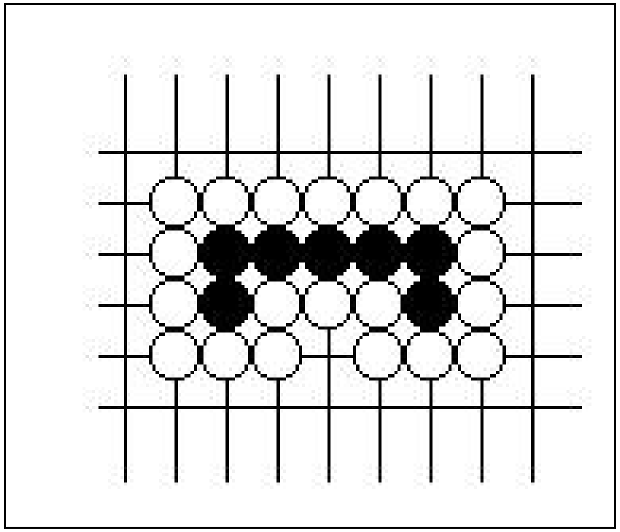
– 算法

```
void polyfill (polygon, color)
{
    for (各条扫描线i )
        把y_min = i 的边放进边表ET[i];
    y = 最低扫描线号;
    初始化活化边表AET为空;
    for (各条扫描线i )
    {
        把边表ET [i] 中的边结点用插入排序法插入AET表，使之
        按x坐标递增顺序排列;
        遍历AET表，把配对交点区间上的象素(x, y)，用指定颜
        色color显示;
        遍历AET表，把y_max= i 的结点从AET表中删除，并把y_max
        大于i 的结点的x值递增Δx;
    }
}
```



二. 区域填充

- 区域：点阵表示的图形，像素集合
- 表示方法：内点表示、边界表示
- 内点表示
  - 枚举出区域内部的所有像素
  - 内部的所有像素着同一个颜色
  - 边界像素着与内部像素不同的颜色
- 边界表示
  - 枚举出边界上所有的像素
  - 边界上的所有像素着同一颜色
  - 内部像素着与边界像素不同的颜色



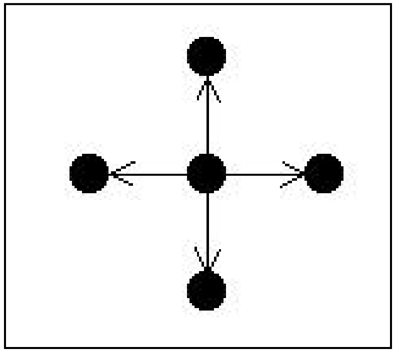
区域填充： 对区域重新着色的过程

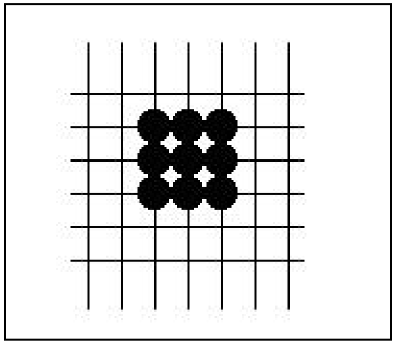
– 将指定的颜色从种子点扩展到整个区域的过程
– 区域填充算法要求区域是连通的

- 连通性

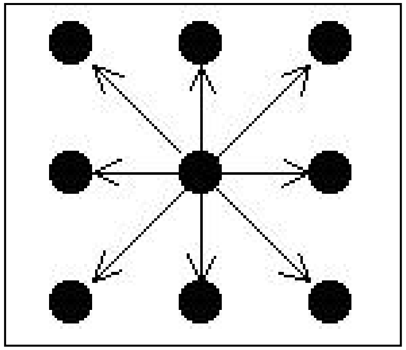
  4连通、8连通

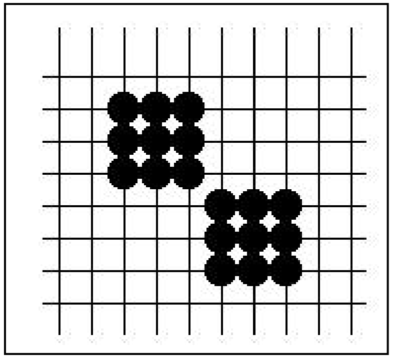
- 4连通



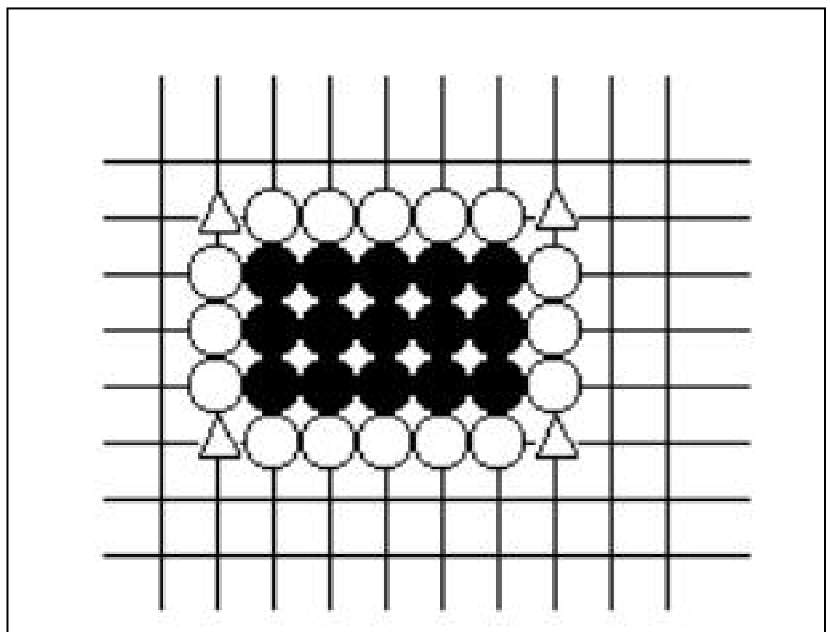


- 8连通

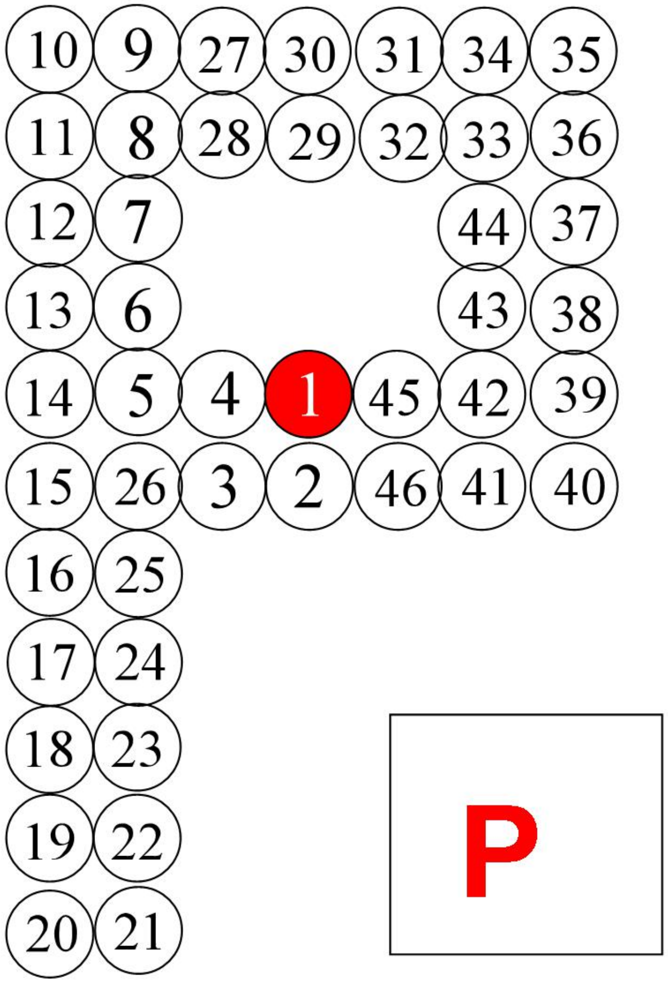




- 4连通与8连通区域的区别
  - 连通性： 4连通可看作8连通区域，但对边界有要求
  - 对边界的要求



4连通区域递归填充算法：




```
Void  Fill(int x,int y,int bcolor,
                int ncolor)
{ int  color;
  color = GetPixel(x,y);
  if ((color!=bcolor)&&(color!=ncolor))
   { PutPixel( x, y, ncolor);
     Fill(x, y+1, bcolor, ncolor);
     Fill(x, y -1, bcolor, ncolor);
     Fill(x -1, y, bcolor, ncolor);
     Fill(x +1, y, bcolor, ncolor);    }
  }
```

利用堆栈实现4连通区域的填充算法：

```
初始化：将算法设置的堆栈置为空；
设置种子象素（x，y）及填充颜色（ncolor）；
PutPixel(x,y,ncolor);
PushStack(x,y);
While(栈为非空）
{  访问栈顶元素；
   if（在填充区域内，栈顶元素存在未被填充的邻接象素）
   { 取其中一个邻接象素（x，y）；
     PutPixel（x，y，ncolor）；
     PushStack（x，y）}
   else
     PopStack（栈顶元素）
 }
```

本 章 小 结

1. 光栅扫描显示器上的图形都是由象素点组成的。

2. 直线和圆弧是最基本的图形元素。在光栅扫描显示器上绘制直线和圆弧，实际上是通过算法找到最靠近直线或圆弧的象素点，并将其点亮。因此，不可避免地会出现走样现象。

3. 其余的二次曲线可采用参数插值的方法绘制。插值点之间用小直线段连接。

4. 自由曲线采用分段拟合的方法绘制。分段后用来拟合的曲线，绘制时的方法同上。

5. 多边形扫描转换利用与扫描线的交点构成边对，通过对边对之间象素点的处理完成多边形绘制。

6. 区域填充采用种子填充算法通过设置内部点颜色完成整个区域的填充。