《科学计算可视化》实验报告

实验题目： 采用网格序列法实现等值线生成算法

姓名 王雨朦

学号 3012216083

专业 计算机科学与技术

班级 3 班

天津大学计算机科学与技术学院

2013 年 10 月 28 日

一、 实验内容

采用网格序列法实现等值线生成算法；

1) 基本要求：

- 网格分辨率>20x20
- 网格结点数据为随机生成的[0,1]数据
- 将等值线用分别用不同颜色标示

2) 采用三角剖分法实现等值线生成
3) 采用曲线插值方法优化等值线生成

二、 主要技术及实现

实验采用 MFC 进行实现，主要技术和核心代码如下：

1. 生成一个 22*22 的网格。
2. 为网格顶点赋[0,1]区间内的随机值。
3. 每个顶点与等值线值进行比较，标正负符号(正用 1 表示，负用 0 表示)。
4. 等值线绘制:

✧ 网格序列法：

a) 对每个小网格进行遍历。分情况讨论不同顶点情况下的网格中等值线的形状，共分为以下几种情况：全正（无等值线不考虑）、全负（无等值线不考虑）、三正一负、三负一正、两正两负
b) 对于每种情况，找出等值线和网格的交点坐标，并连接
c) 二义性的处理。在两正两负且对角线顶点符号相同时,需进行二义性处理：算出中心结点的值 cent,与等值线值进行比较，标出正负。再将中心点符号和左上角的符号进行比较，如果相同是一种绘制情况，不相同则是另外一种情况

✧ 三角剖分法：

a) 对每个方格进行遍历时，把方格分为四个三角形用四个顶点的平均值表示中心点的值 midsign。每个三角形以中心点 mid 为底部，左、右顶点分别赋值，看成一个倒三角。
b) 对于每个方格里的每个倒三角，都有一正两负和一负两正的情况。找出等值线和三角形的交点，连接即可。

5. 优化等值线生成

用 Hermite 曲线优化等值线的生成：此处优化的是网格序列法，起点终点分别标记为（a,b）(c,d)，曲线分为 100 段，参数 t 取 0.01，范围为[] 坐标向量对于横线上的起点和终点取（10，0）（0，10）。对于每个参数 t，逐点进行连接，直到端点。

三、 实验结果

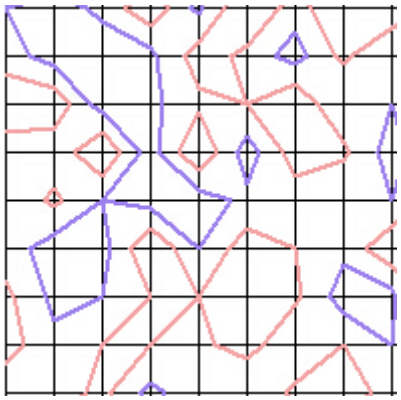

图 1：网格序列法绘制等值线的值分别为 0.2 和 0.6 的等值线图

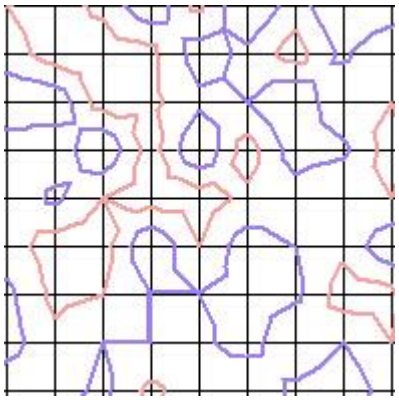

图 2：三角剖分法绘制等值线值分别为 0.2 和 0.6 的等值线图

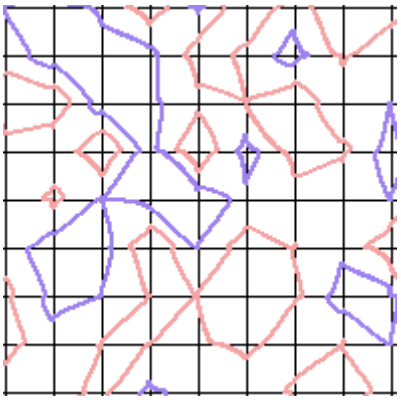

图 3：Herbite 插值法优化等值线值分别为 0.2 和 0.6 的等值线

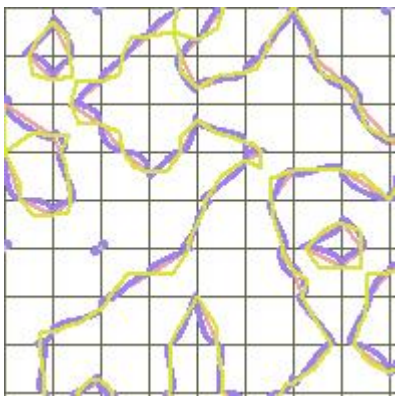

图 4：网格序列（粉色）、三角剖分（黄色）、Herbinite（蓝色）插值绘制等值线值为 0.6 的等值线

四、实验分析和总结

1. 实验分析

从实验结果来看，网格序列法绘制等值线实现情况较好，但是网格绘制的比较生硬，而且比较不精确；

三角剖分法绘制等值线实现情况比较好，等值线绘制比网格法更加精确，但是由于是由直线段拼接而成，所以还是显得生硬了一些；

Hermite 插值优化算法，优化后的曲线拼接成的等值线本应该比直线平滑细腻很多，但是笔者在这里实现的不够好。生成的曲线比较平滑，但是曲线段之间的连接不够自然，导致等值线的视觉效果不理想。

Hermite 插值算法的不理想结果，应该是由于笔者在这里，对于每一条方格中的等值线段，其起点和终点的切向量直接简单粗暴的取为点所在方格线的方向向量。一种优化改进的思想是，对于每一个等值线和方格线的交点，把其切向量取为入边的方向向量与出边方向向量的和，就能使交点处曲线的连接更加的平稳光滑。

2. 实验总结

这次实验的网格序列和基于网格的三角剖分算法实现的比较理想，实验结果较明晰。而 Hermite 插值算法的实验结果不够理想，应该根据实验分析中的优化思想进一步的改进优化实验。

通过这次实验，不仅学到使用 MFC 基于单文档，在 ONDRAW 函数里绘图的功能，更重要的是深刻理解和掌握了网格序列算法、三角剖分算法和 Hermite 插值算法。

五、核心源代码

```
//以等值线为0.6为例

    #define GRIDNUM 22 //网格数
    #define GRIDLEN 24 //网格长度
    #define VAL 0.6 //等值线值
    #define VECTOR 10 //插值算法中的向量参数
    BOOL flag = TRUE;  //解决赋随机值时ONDRAW函数刷新问题
    double point[GRIDNUM][GRIDNUM];

//插值函数
    void Lint(int sta, int end, double a, double b, double c, double d, CDC* pDC)
    {
    CPen myPen1(PS_SOLID, 3, RGB(158, 130, 238));//插值等值线用蓝色的笔
    CPen* oldPen1 = pDC->SelectObject(&myPen1);
//判断横交点所在的方格线为竖线还是横线，并给坐标向量赋值
    double dpx, dpy, dpx1, dpy1;//每一条等值线段起点和终点坐标向量
    if (sta == 0 || sta == 2){ dpx = VECTOR; dpy = 0; }
    else { dpx = 0; dpy = VECTOR; }
    if (end == 0 || end == 2){ dpx1 = VECTOR; dpy1 = 0; }
    else { dpx1 = 0; dpy1 = VECTOR; }
//插值
    double px = a, py = b, px1, py1;//线上的每一个先后点
    for (double t = 0.01; t < 1.00; t = t + 0.01)
    {
        //Hermite公式
        double t1 = 2 * t*t*t - 3 * t*t + 1,
            t2 = -2 * t*t*t + 3 * t*t,
            t3 = t*t*t - 2 * t*t + t,
            t4 = t*t*t - t*t;
        px1 = a*t1 + c*t2 + dpx*t3 + dpx1*t4;
        py1 = b*t1 + d*t2 + dpy*t3 + dpy1*t4;
        pDC->MoveTo(px, py);// 连接先后点
        pDC->LineTo(px1, py1);
        px = px1; py = py1;
    }
    pDC->MoveTo(px1, py1);// 连接终点
    pDC->LineTo(c, d);
}
//设置画笔，颜色，并且选择到设备里
    CClientDC dc(this);
    CPen myPen, *Oldpen;
```

```
    myPen.CreatePen(PS_SOLID, 0.5, RGB(103, 105, 83));
    Oldpen = dc.SelectObject(&myPen);
// 画网格 22*22
    int i;
    int n = (GRIDNUM - 1)* GRIDLEN;
    for (i = 0; i <= n; i = i + GRIDLEN)
    {
        dc.MoveTo(0, i);
        dc.LineTo(n, i);
        dc.MoveTo(i, 0);
        dc.LineTo(i, n);
    }
//为网格点赋随机值
    for (int i = 0; i < GRIDNUM; i++)
    {
        for (int j = 0; j < GRIDNUM; j++)
        {
            if(flag == TRUE)
            {
                point[i][j] = ((rand() % 11) / 10.00); //0~10用%11
            }
        }
    }
    flag = FALSE; //解决赋随机值时 ONDRAW 函数刷新问题
//为网格标正负号
    int sign[GRIDNUM][GRIDNUM];
    for (int i = 0; i < GRIDNUM; i++)
    {
        for (int j = 0; j < GRIDNUM; j++)
        {
            if (point[i][j] < VAL)
            {
                sign[i][j] = 0; //0代表负号
            }
            else sign[i][j] = 1; //1代表正号
        }
    }
//网格序列法。判断四个顶点正负类型
    CClientDC dc1(this);
    CPen myPen1(PS_SOLID, 2, RGB(244, 164, 164));//等值线用红色的笔
    CPen* oldpen1 = dc1.SelectObject(&myPen1);
    for (int i = 0; i < GRIDNUM - 1; i++) //
    {
        for (int j = 0; j < GRIDNUM - 1; j++)
```

```
        {
            x0 = i * GRIDLEN;
            y0 = j * GRIDLEN;
            x1 = (i + 1) * GRIDLEN;
            y1 = (j + 1) * GRIDLEN;
        //一正三负
            if ((sign[i][j] + sign[i + 1][j] + sign[i][j + 1] + sign[i + 1][j + 1]) == 1)
            {
                if (sign[i][j] == 1)
                {
                    a = x0;
                    b = ((point[i][j] - VAL)*y1 + (VAL - point[i][j + 1])*y0) /
      (point[i][j]- point[i][j + 1]);
                    c = ((point[i][j] - VAL)*x1 + (VAL - point[i + 1][j])*x0) /
       (point[i][j]-point[i + 1][j]);
                    d = y0;
                    sta = 1; end = 0;
                }
                if (sign[i + 1][j] == 1)
                {
                    a = ((point[i + 1][j] - VAL)*x0 + (VAL - point[i][j])*x1) / (point[i
+ 1][j] - point[i][j]);
                    b = y0;
                    c = x1;
                    d = ((point[i + 1][j] - VAL)*y1 + (VAL - point[i + 1][j + 1])*y0) /
(point[i + 1][j] - point[i + 1][j + 1]);
                    sta = 0; end = 3;
                }
                if (sign[i][j + 1] == 1)
                {
                    a = ((point[i][j + 1] - VAL)*x1 + (VAL - point[i + 1][j + 1])*x0) /
(point[i][j + 1] - point[i + 1][j + 1]);
                    b = y1;
                    c = x0;
                    d = ((point[i][j + 1] - VAL)*y0 + (VAL - point[i][j])*y1) / (point[i][j
+ 1] - point[i][j]);
                    sta = 2; end = 1;
                }
                if (sign[i + 1][j + 1] == 1)
                {
                    a = x1;
                    b = ((point[i + 1][j + 1] - VAL)*y0 + (VAL - point[i + 1][j])*y1) /
(point[i + 1][j + 1] - point[i + 1][j]);
                    c = ((point[i + 1][j + 1] - VAL)*x0 + (VAL - point[i][j + 1])*x1) /
```

```
(point[i + 1][j + 1] - point[i][j + 1]);
                    d = y1;
                    sta = 3; end = 2;
                }
            }//end一正三负
    //一负三正
            if ((sign[i][j] + sign[i + 1][j] + sign[i][j + 1] + sign[i + 1][j + 1]) == 3)
            {
                if (sign[i][j] == 0)
                {
                    c = x0;
                    d = (abs(point[i][j] - VAL)* y1 + abs(VAL - point[i][j + 1])*y0) /
abs(point[i][j] - point[i][j + 1]);
                    a = (abs(point[i][j] - VAL)* x1 + abs(VAL - point[i + 1][j])*x0) /
abs(point[i][j] - point[i + 1][j]);
                    b = y0;
                    sta = 0; end = 1;
                }
                if (sign[i + 1][j] == 0)
                {
                    c = (abs(point[i + 1][j] - VAL)*x0 + abs(VAL - point[i][j])*x1) /
abs(point[i + 1][j] - point[i][j]);
                    d = y0;
                    a = x1;
                    b = (abs(point[i + 1][j] - VAL)*y1 + abs(VAL - point[i + 1][j + 1])*y0)
/ abs(point[i + 1][j] - point[i + 1][j + 1]);
                    sta = 3; end = 0;
                }
                if (sign[i][j + 1] == 0)
                {
                    c = (abs(point[i][j + 1] - VAL)*x1 + abs(VAL - point[i + 1][j + 1])*x0)
/ abs(point[i][j + 1] - point[i + 1][j + 1]);
                    d = y1;
                    a = x0;
                    b = (abs(point[i][j + 1] - VAL)*y0 + abs(VAL - point[i][j])*y1) /
abs(point[i][j + 1] - point[i][j]);
                    sta = 1; end = 2;
                }
                if (sign[i + 1][j + 1] == 0)
                {
                    c = x1;
                    d = (abs(point[i + 1][j + 1] - VAL)*y0 + abs(VAL - point[i + 1][j])*y1)
/ abs(point[i + 1][j + 1] - point[i + 1][j]);
                    a = (abs(point[i + 1][j + 1] - VAL)*x0 + abs(VAL - point[i][j + 1])*x1)
```

```
/ abs(point[i + 1][j + 1] - point[i][j + 1]);
                    b = y1;
                    sta = 2; end = 3;
                }
            }
            dc1.MoveTo(a, b);//方格
            dc1.LineTo(c, d);
            Lint(sta, end, a, b, c, d, pDC);//插值
//两个正点
            if ((sign[i][j] + sign[i + 1][j] + sign[i][j + 1] + sign[i + 1][j + 1]) == 2)
            {
        //两个正点相邻
                if (sign[i][j] + sign[i + 1][j + 1] == 1) //异号
                {
                    if (sign[i][j] == 0 && sign[i + 1][j] == 0)
                    {
                        c = x0;
                        d = ((point[i][j + 1] - VAL)*y0 + (VAL - point[i][j])*y1) /
(point[i][j + 1] - point[i][j]);
                        a = x1;
                        b = ((point[i + 1][j + 1] - VAL)*y0 + (VAL - point[i + 1][j])*y1)
/ (point[i + 1][j + 1] - point[i + 1][j]);
                        sta = 3; end = 1;
                    }
                    if (sign[i][j + 1] == 0 && sign[i + 1][j + 1] == 0)
                    {
                        a = x0;
                        b = ((point[i][j + 1] - VAL)*y0 + (VAL - point[i][j])*y1) /
(point[i][j + 1] - point[i][j]);
                        c = x1;
                        d = ((point[i + 1][j + 1] - VAL)*y0 + (VAL - point[i + 1][j])*y1)
/ (point[i + 1][j + 1] - point[i + 1][j]);
                        sta = 1; end = 3;
                    }
                    if (sign[i][j] == 0 && sign[i][j + 1] == 0)
                    {
                        b = y0;
                        a = ((point[i + 1][j] - VAL)*x0 + (VAL - point[i][j])*x1) /
(point[i + 1][j] - point[i][j]);
                        d = y1;
                        c = ((point[i + 1][j + 1] - VAL)*x0 + (VAL - point[i][j + 1])*x1)
/ (point[i + 1][j + 1] - point[i][j + 1]);
                        sta = 0; end = 2;
                    }
```

```
                    if (sign[i + 1][j] == 0 && sign[i + 1][j + 1] == 0)
                    {
                        d = y0;
                        c = ((point[i + 1][j] - VAL)*x0 + (VAL - point[i][j])*x1) /
(point[i + 1][j] - point[i][j]);
                        b = y1;
                        a = ((point[i + 1][j + 1] - VAL)*x0 + (VAL - point[i][j + 1])*x1)
/ (point[i + 1][j + 1] - point[i][j + 1]);
                        sta = 2; end = 0;
                    }
                    dc1.MoveTo(a, b);//方格
                    dc1.LineTo(c, d);
                    Lint(sta, end, a, b, c, d, pDC);//插值
                }
//两个正点相对    else
                {
        //消除二义性
                    double cent = (point[i][j] + point[i + 1][j] + point[i][j + 1] +
point[i + 1][j + 1]) / 4;
                    int centsign;
                    if (cent < VAL) centsign = 0;
                    else centsign = 1;
                    if ((sign[i][j] == centsign)) //左上角和中心点相等
                    {
                        //两条线
                        a = (abs(point[i + 1][j] - VAL)*x0 + abs(VAL - point[i][j])*x1)
/ abs(point[i + 1][j] - point[i][j]);
                        b = y0;
                        c = x1;
                        d = (abs(point[i + 1][j] - VAL)*y1 + abs(VAL - point[i + 1][j
+ 1])*y0) / abs(point[i + 1][j] - point[i + 1][j + 1]);
                        sta = 0; end = 3;
                        Lint(sta, end, a, b, c, d, pDC);
                        dc1.MoveTo(a, b);
                        dc1.LineTo(c, d);

                        a = (abs(point[i][j + 1] - VAL)*x1 + abs(VAL - point[i + 1][j
+ 1])*x0) / abs(point[i][j + 1] - point[i + 1][j + 1]);
                        b = y1;
                        c = x0;
                        d = (abs(point[i][j + 1] - VAL)*y0 + abs(VAL - point[i][j])*y1)
/ abs(point[i][j + 1] - point[i][j]);
                        dc1.MoveTo(a, b);
                        dc1.LineTo(c, d);
```

```
                        sta = 2; end = 1;
                        Lint(sta, end, a, b, c, d, pDC);
                    }
                    else //cent>=val
                    {
                        c = x0;
                        d = (abs(point[i][j] - VAL)* y1 + abs(VAL - point[i][j + 1])*y0)
/ abs(point[i][j] - point[i][j + 1]);
                        a = (abs(point[i][j] - VAL)* x1 + abs(VAL - point[i + 1][j])*x0)
/ abs(point[i][j] - point[i + 1][j]);
                        b = y0;
                        dc1.MoveTo(a, b);
                        dc1.LineTo(c, d);
                        sta = 0; end = 1;
                        Lint(sta, end, a, b, c, d, pDC);

                        c = x1;
                        d = (abs(point[i + 1][j + 1] - VAL)*y0 + abs(VAL - point[i +
1][j])*y1) / abs(point[i + 1][j + 1] - point[i + 1][j]);
                        a = (abs(point[i + 1][j + 1] - VAL)*x0 + abs(VAL - point[i][j
+ 1])*x1) / abs(point[i + 1][j + 1] - point[i][j + 1]);
                        b = y1;
                        dc1.MoveTo(a, b);
                        dc1.LineTo(c, d);
                        sta = 3; end = 2;
                        Lint(sta, end, a, b, c, d, pDC);
                    }
                }//else
            }//if
        }//for
    }//for

/*三角剖分法*/
    CClientDC dc2(this);
    CPen myPen2(PS_SOLID, 2, RGB(216, 228, 64));//三角剖分等值线用黄色的笔
    CPen* oldpen2 = dc2.SelectObject(&myPen2);

    for (int i = 0; i < GRIDNUM - 1; i++)
    {
        for (int j = 0; j < GRIDNUM - 1; j++)
        {
            //方格的顶点坐标
            x0 = i * GRIDLEN;
            y0 = j * GRIDLEN;
```

```
            x1 = (i + 1) * GRIDLEN;
            y1 = (j + 1) * GRIDLEN;
            //计算中点值并赋符号
            double midpoint = (point[i][j] + point[i + 1][j] + point[i][j + 1] + point[i
+ 1][j + 1]) / 4;
            double midx = (x0 + x1) / 2, midy = (y0 + y1) / 2; //mid的坐标
            //mid的正负标志
            int midsign;
            if (midpoint < VAL) midsign = 0;
            else midsign = 1;
            //以三角剖分后的mid为三角形的底部点，左边和右边的坐标,符号和随机值
            double lx, ly, rx, ry;
            double lpoint, rpoint;
            int lsign, rsign;

            for (int k = 0; k < 4; k++)
            {
                if (k == 0)
                {
                    //上三角
                    lsign = sign[i][j], lpoint = point[i][j],
                        rsign = sign[i + 1][j], rpoint = point[i + 1][j],
                        lx = x0, ly = y0, rx = x1, ry = y0;
                }
                if (k == 1)
                {
                    //左三角
                    lsign = sign[i][j + 1], lpoint = point[i][j + 1],
                        rsign = sign[i][j], rpoint = point[i][j],
                        lx = x0, ly = y1, rx = x0, ry = y0;
                }
                if (k == 2)
                {
                    //右三角
                    lsign = sign[i + 1][j], lpoint = point[i + 1][j],
                        rsign = sign[i + 1][j + 1], rpoint = point[i + 1][j + 1],
                        lx = x1, ly = y0, rx = x1, ry = y1;
                }
                if (k == 3)
                {
                    //下三角
                    lsign = sign[i + 1][j + 1], lpoint = point[i + 1][j + 1],
                        rsign = sign[i][j + 1], rpoint = point[i][j + 1],
                        lx = x1, ly = y1, rx = x0, ry = y1;
```

```
                }

                if (lsign + rsign + midsign == 1)
                {
                    if (lsign == 1)
                    {
                        a = (abs(lpoint - VAL)*rx + abs(VAL - rpoint)*lx) / abs(lpoint
- rpoint);
                        b = (abs(lpoint - VAL)*ry + abs(VAL - rpoint)*ly) / abs(lpoint
- rpoint);
                        c = (abs(lpoint - VAL)*midx + abs(VAL - midpoint)*lx) /
abs(lpoint - midpoint);
                        d = (abs(lpoint - VAL)*midy + abs(VAL - midpoint)*ly) /
abs(lpoint - midpoint);
                    }
                    if (rsign == 1)
                    {
                        a = (abs(rpoint - VAL)*lx + abs(VAL - lpoint)*rx) / abs(rpoint
- lpoint);
                        b = (abs(rpoint - VAL)*ly + abs(VAL - lpoint)*ry) / abs(rpoint
- lpoint);
                        c = (abs(rpoint - VAL)*midx + abs(VAL - midpoint)*rx) /
abs(rpoint - midpoint);
                        d = (abs(rpoint - VAL)*midy + abs(VAL - midpoint)*ry) /
abs(rpoint - midpoint);
                    }
                    if (midsign == 1)
                    {
                        a = (abs(lpoint - VAL)*midx + abs(VAL - midpoint)*lx) /
abs(lpoint - midpoint);
                        b = (abs(lpoint - VAL)*midy + abs(VAL - midpoint)*ly) /
abs(lpoint - midpoint);
                        c = (abs(rpoint - VAL)*midx + abs(VAL - midpoint)*rx) /
abs(rpoint - midpoint);
                        d = (abs(rpoint - VAL)*midy + abs(VAL - midpoint)*ry) /
abs(rpoint - midpoint);
                    }
                }//if
                //两正一负
                if (lsign + rsign + midsign == 2)
                {
                    if (lsign == 0)
                    {
                        a = (abs(lpoint - VAL)*rx + abs(VAL - rpoint)*lx) / abs(lpoint
```

```
- rpoint);
                        b = (abs(lpoint - VAL)*ry + abs(VAL - rpoint)*ly) / abs(lpoint
- rpoint);
                        c = (abs(lpoint - VAL)*midx + abs(VAL - midpoint)*lx) /
abs(lpoint - midpoint);
                        d = (abs(lpoint - VAL)*midy + abs(VAL - midpoint)*ly) /
abs(lpoint - midpoint);
                    }
                    if (rsign == 0)
                    {
                        a = (abs(rpoint - VAL)*lx + abs(VAL - lpoint)*rx) / abs(rpoint
- lpoint);
                        b = (abs(rpoint - VAL)*ly + abs(VAL - lpoint)*ry) / abs(rpoint
- lpoint);
                        c = (abs(rpoint - VAL)*midx + abs(VAL - midpoint)*rx) /
abs(rpoint - midpoint);
                        d = (abs(rpoint - VAL)*midy + abs(VAL - midpoint)*ry) /
abs(rpoint - midpoint);
                    }
                    if (midsign == 0)
                    {
                        a = (abs(lpoint - VAL)*midx + abs(VAL - midpoint)*lx) /
abs(lpoint - midpoint);
                        b = (abs(lpoint - VAL)*midy + abs(VAL - midpoint)*ly) /
abs(lpoint - midpoint);
                        c = (abs(rpoint - VAL)*midx + abs(VAL - midpoint)*rx) /
abs(rpoint - midpoint);
                        d = (abs(rpoint - VAL)*midy + abs(VAL - midpoint)*ry) /
abs(rpoint - midpoint);
                    }
                }
                dc2.MoveTo(a, b);
                dc2.LineTo(c, d);
            }//for4
        }//for
    }//for
```