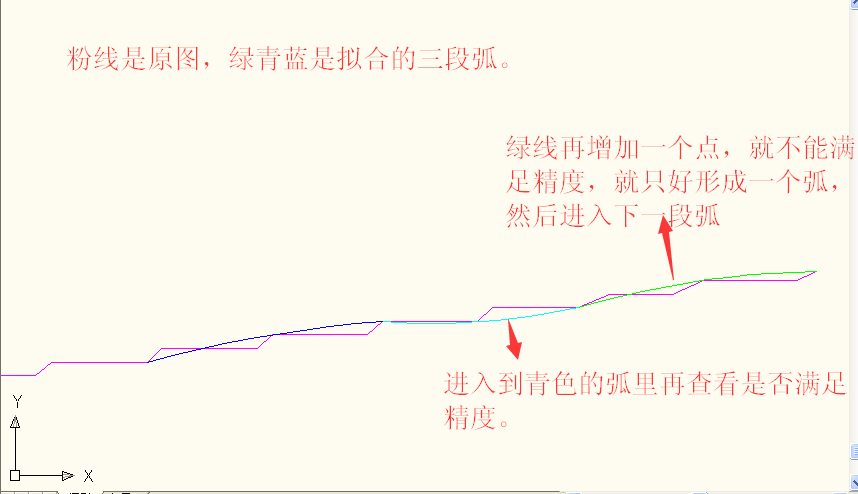
一．图形简化算法思想

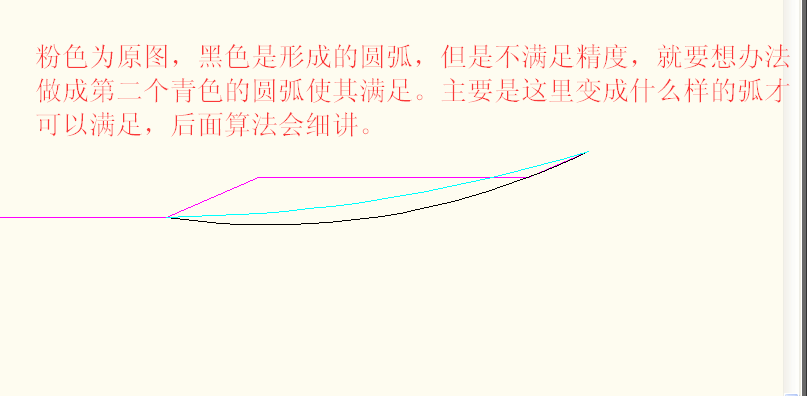
1. 先去掉图形上相邻点斜率一样的点：从头扫描点，每增加一个点后，看开始点与增加点之间的点是否在这两个点的直线上。如果是就删除掉，不是就从新选择下一个开始点。
2. 把整理后的点拟合成圆弧：

如下图：粉色的是原图，绿青蓝色为测试的圆弧。每增加一个点取起点，中间点，终点做圆弧，如果满足精度就增加一个点再形成圆弧看是否满足精度。如果不满足精度改变圆弧的凸度后再查看是否满足精度，如果还不满足精度就保存之前可以满足的圆弧，形成第一段圆弧，然后进入下一段圆弧的形成。

下图为圆弧的增加：



下图为改变圆弧凸度，使其满足精度。



二.流程图

否

是

扫描点集里的点，增加一个点取起点，中间点，终点形成圆弧，看是否满足精度。

去掉图形上相邻点斜率一样的点，也可以说去掉一条直线上有多个点。

开始

把之前的圆弧保存

否

记录最大距离（图形在圆弧里）和最小距离（图形在圆弧外）。最大最小距离是否都超出精度。

是

只有最大或最小超出精度

是

改变圆弧的凸度，使其在最大或最小那点在满足精度，再查看新的圆弧是否满足期间的点。

否

扫描完所有点结束

三．用到的算法

1，已知三个点。求一个圆：

数学问题：就是解三元二次方程组！

圆方程：(x-a)^2+(y-b)^2=r^2。

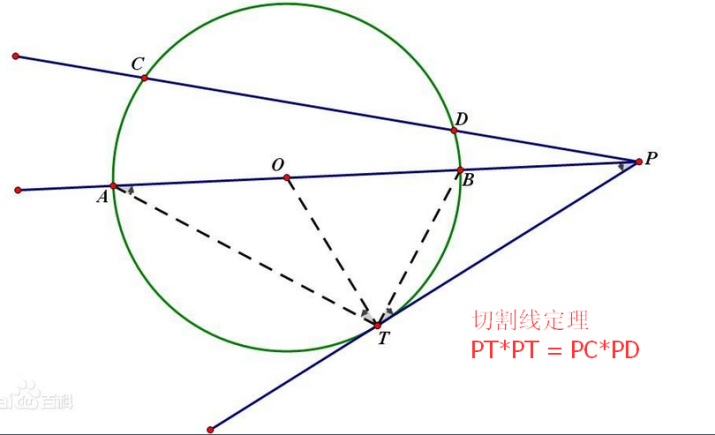
把已知三个点的坐标代入圆方程，解方程组即可

(x1-X)²-(y1-Y)²=R²      (1)式  
(x2-X)²-(y2-Y)²=R²      (2)式  
(x3-X)²-(y3-Y)²=R²      (3)式

C++代码：如下

a=2\*(x2-x1);  
b=2\*(y2-y1);  
c=x2\*x2+y2\*y2-x1\*x1-y1\*y1;  
d=2\*(x3-x2);  
e=2\*(y3-y2);  
f=x3\*x3+y3\*y3-x2\*x2-y2\*y2;  
圆心：x=(b\*f-e\*c)/(b\*d-e\*a);  
圆心：y=(d\*c-a\*f)/(b\*d-e\*a);  
半径：r=sqrt((x-x1)\*(x-x1)+(y-y1)\*(y-y1));

2，已知圆上两个点和圆的切线。求圆



根据两条直线交于一点求得点P,然后根据切割线定理求点T。再根据三点C,D,T求圆。

解一元二次方程组:

直线PT斜率： k

直线PT长度： l

直线PT与Y轴交点： b

P点坐标（x1,y1）

以下是关于圆心坐标x的一元二次方程一般式的A,B,C :

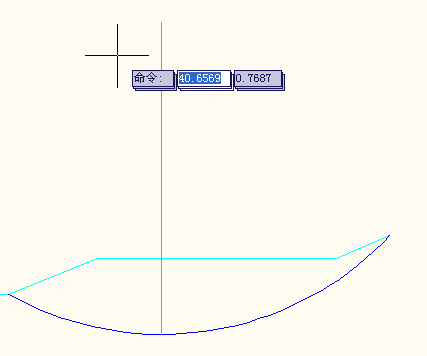
A = 1+k\*k;

B = -2\* x1 -2\*( y1-b)\*k;

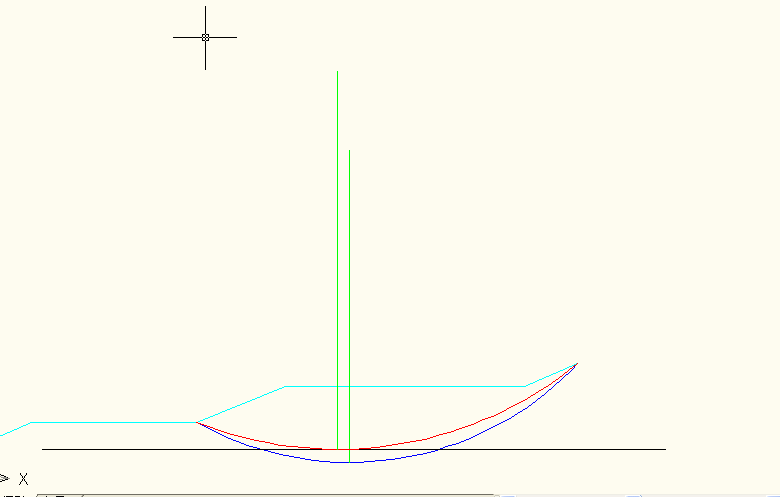
C = x1\* x1+( y1-b)\*( y1-b)-l;

以下是应用的地方：

假如设定精度为1。青色的是原图形，蓝色为第一次形成的弧，绿色为过垂直青线的半径，弧与青线最远距离为1.2。不满足精度。

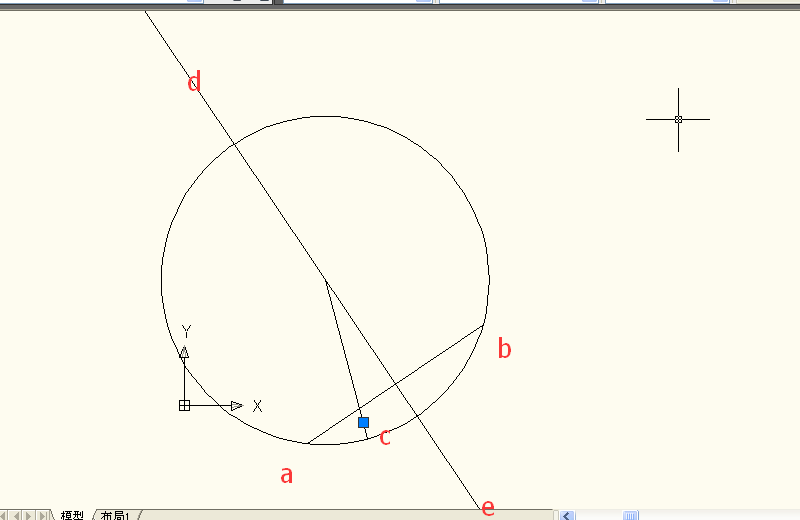


然后做一条黑色线与青色线平行，距离为0.9。再做一个弧红色。这样原图与红色的弧最远距离为0.9。满足精度要求。



1. 已知圆上两个点和一点到圆的距离。求圆

先求圆上两点的连线的垂线de。然后根据圆心到点c的距离加上点到圆的距离 = 半径列关于圆心坐标x的一元二次方程。



解一元二次方程组:

设：点a: posRound1, 点b: posRound2, 点c: posInRound

直线de的斜率： k

直线de与Y轴的交距：b

e = 2\*k\*posInRound.y + 2\*posInRound.x -2\*k\*posRound1.y - 2\*posRound1.x;

f = posRound1.y\*posRound1.y +posRound1.x\*posRound1.x + 2\*b\*posInRound.y - 2\*b\*posRound1.y - posInRound.y\*posInRound.y -posInRound.x\*posInRound.x -dDisToRound\*dDisToRound;

以下是关于圆心坐标x的一元二次方程一般式的A,B,C :

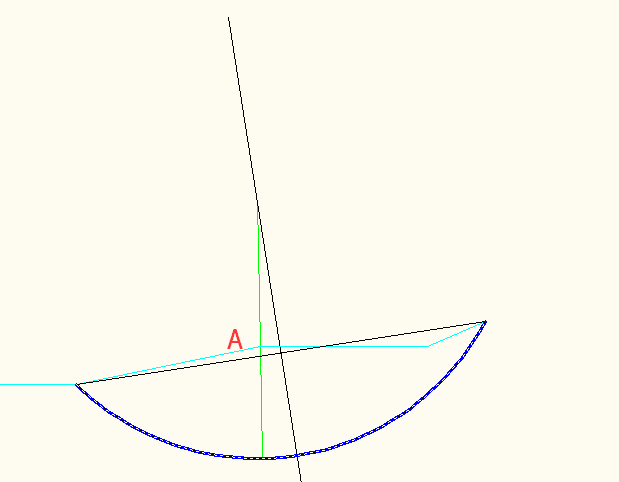
A = 4\*dDisToRound\*dDisToRound\*(1+k\*k) - e\*e;

B = 4\*dDisToRound\*dDisToRound\*(2\*k\*(b-posInRound.y) - 2\*posInRound.x) - 2\*e\*f;

C = 4\*dDisToRound\*dDisToRound\*((b-posInRound.y)\*(b-posInRound.y)+ posInRound.x\*posInRound.x) - f\*f;

以下是应用的地方：

假如要求精度为1。青色是原图，蓝色为第一次形成的弧，原图离弧最远的A点并不是直线（因为过圆心与直线的垂足并不在线段内）。假如点A到弧的距离为1.2。不满足精度要求。



然后设定一个距离到点A的距离为0.9。通过上面的算法就可以得到红色的弧，为满足条件。

