目录

1.问题描述······ · · · ·· · · · · ·· · · · · · · · · · · · · · · · · · ·· · · · ··· ·P01

2.基本求解思路······ · · ·· · · ·· · · · · · · · · · · · · · · · · · ·· · · · ··· ·P01

3.算法设计······ · · · ·· · · · · ·· · · · · · · · · · · · · · · · · · ·· · · · ··· ·P01

4.界面设计······ · · · ·· · · · · ·· · · · · · · · · · · · · · · · · · ·· · · · ··· ·P05

5.结果展示······ · · · ·· · · · · ·· · · · · · · · · · · · · · · · · · ·· · · · ··· ·P06

6. 期间遇到的问题，心得，感想以及收获· · · · · · · · ·· · · · ········· · ·P08

# 问题描述

1.绘制一个能显示直角坐标系的窗口，且需在x y轴相应位置注释上该点的值；

2.写一个能够将输入的字符串处理成算式的函数处理输入的字符串；

3.以一定的精确度在[-π，π]区间内不断改变x的值以得到相应的y值；

4.将对应的x，y经过一定比例缩放绘制到直角坐标系中形成连续的曲线。

# 基本求解思路

通过UI界面绘制的辅助将窗口绘制完成，在TextEdit输入算式后复制给string类型变量，从左到右反复扫描该字符串，确立优先级关系从最内层括号里的内容开始求解，算出的值将填充掉原来括号里的算式和括号的位置，简单来说就是从左到右逐个求出每个括号内带入x应有的值，然后层层向外算，求解最后返回相应的y值，在绘图窗口以（x，y）点为圆心绘制一个半径为2像素的圆，x只需精确到0.01便基本可在窗口绘制一条连续的曲线，该曲线即为所求函数图像。

# 算法设计

**对于该算法，我的核心思维就是找到最内层括号，带入x求出它的值，将值带回原式相应位置，继续找下一个最内层括号，直到求出整个式子，具体流程如图1所示。**

|  |
| --- |
| 算法3.1 |
|  |
| 图1.算法基本流程图 |

C++核心代码

核心求解函数有四个：

分别为

double MainWindow::**changetonumber**(string a)

string MainWindow::**changetostring**(double a)

double MainWindow::**result**(string str,double x)

void MainWindow::**number**(int start,int end,int condition,double x)

表2 算法1核心代码

|  |
| --- |
| 函数：double MainWindow::**changetonumber**(string a) |
| 功能描述：该函数的主要功能是将传入的string a解析为double类型的数字  其解析的基本规则为： |
| 1. 从a[0]开始遍历该字符串，且若扫描到’-’则会将最终结果\*-1再返回，若在为扫描到小数点的情况下扫描到数字则进行result=result\*10+a[i]-‘0’的运算（result初值为0） 2. 若在为扫描到小数点的情况下扫描到数字则进行result += double((a[i]-'0'))/pow(10,dot)的运算此处的dot为扫描到小数点后a[i]的i++的次数+1 3. 返回该值 |

|  |
| --- |
| 函数：string MainWindow::**changetostring**(double a) |
| 功能描述：该函数的主要功能是将传入的double a解析为string类型的数字  其解析的基本规则为： |
| 1.当a=“0”直接返回0 |
| 2.当a<0时，先给要返回结果s变量前添加一个‘-’号，且a\*=-1  3.取出a的整数部分，取出a的小数部分分别存放再两个变量中 |
| 1. 整数部分每次提取一位到s中，然后除10直至该储存a整数部分的变量值为0，此时表明原a的整数部分已全部写入s中 2. 向s写入’.’ 3. 储存a的小数部分的变量循环\*10每次将\*10后的整数部分写入s中，因为精度需求这里循环了四次，后面的值太小可忽略 4. 返回s |
|  |

|  |
| --- |
| 函数： double MainWindow::**result**(string str,double x) |
| 功能描述：该函数主要功能有两个  **功能一、**将用户输入的函数表达式字符串进行预处理，然后切分为一组单词存入string类型vector temp中  操作过程如下：   1. 遍历用户输入的函数字符串 若发现“sin”，删去‘ i ’，‘ n’ ，发现“cos” ，删去‘o’，‘s’（便于后期操作） 2. 划分：若当前字符为‘+’，‘-’，‘\*’，‘/’，‘s’，‘c’，‘（’，‘）’则将该符号切分为一个单词 3. 若遇到数字或小数点则将结果写入一个暂时用来保存的string类型变量中，直至发现2.中所述字符，则将该字符前面那一部分划分为一个单词 存入temp中 4. 在temp首尾分别添加“（”和“）” |
| **功能二、**从temp[1]开始扫描单词（temp[0]必为“（”无需查看），当发现某个单词满足如下条件：   1. 该单词自身不为“（”或“）” 2. 比该单词index小的单词中存在单词“（”，且这两个单词之间不存在“）” 3. 比该单词index大的单词中存在单词“）”，且这两个单词之间不存在“（” |
| 则记录下两个括号单词的下标，同时检查“（”前一项是否为单词“s”或“c”（若有则删除“s”或“c”并将int类型变量condition值变为1或2（默认为0，即不需要进行三角函数运算）），从vector temp中删去这两个括号单词 |
| 将两括号下标，condition，x的值传给number函数  遍历temp查看里面是否有括号单词，若有则循环功能二，若无则通过changetonumber函数将temp[0]转为double类型后返回，此值即为传入x所对应该函数的y值 |
|  |

|  |
| --- |
| 函数：void MainWindow::**number**(int start,int end,int condition,double x) |
| 功能描述：  1.从temp的第start项开始到end项结束，遍历temp  2．若在第i项发现“\*”或“/”，通过changetonumber函数将第i-1和i+1项变为double类型后进行\*或/操作（若某一项为x，则直接带入传入的x的值即可）  3.将结果changetostring写入到temp[i-1]后删除temp[i],temp[i+1],end-=2,继续向后遍历直至没有\*或/单词  4.重复第2步和第3步但此时搜寻目标改为“+”或“-”，其它操作类似 |
| 5.上最终结果将保存在temp[start]里，此时根据condition的值对temp[start]进行sin，cos操作或者不操作  6.返回result函数继续运算 |
|  |

# 界面设计

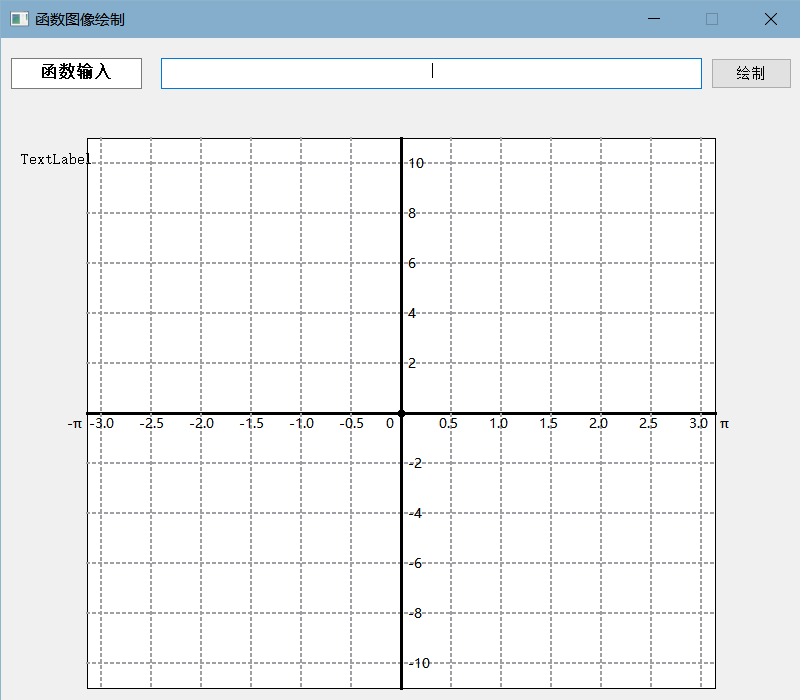
1. 在ui界面通过拖拽设置好了TextEdit和开始绘制的pushbutton；
2. 在界面下方绘制一个以白色为填充色的矩形作为坐标系的背景；
3. Define好横纵向单位像素长度以便于后期绘图时函数值与像素点之间的转换；
4. 通过for循环绘制横线竖线形成坐标系并在绘制时相应的在相应位置表面该线对应x=某个值或y=某个值，**最终绘制效果如图2所示；**
5. 将绘制pushbutton转到槽设置信号为clicked，当用户完成输入时按下绘制按钮会将textEdit内容赋值给str变量以供后期计算以及函数图像绘制。
6. 其它细节已在代码内注释；
7. 绘图既是取足够多的在所给函数图像上的点将它们在坐标系中标出，当点密集程度够大时，便会呈现出连续的曲线。
8. 

图 2 软件界面设计

表3 界面设计核心代码

|  |
| --- |
| 算法3.函数 void MainWindow::***paintEvent***(QPaintEvent \*event) |
| 功能描述：该函数的主要功能是绘制窗口界面，该函数的运行过程是：先判断int类型变量drawp的值是0还是1，若drawp值为0，则表示用户目前未输入函数，只需绘制坐标轴，坐标轴的绘制方法为先在适当位置绘制一个以白色为填充色的628\*550像素的矩形作为背景，再设定像素点QPoint(400,375)，设定单位长度为50像素，即每个正方形格子的边长为50像素  **For（i=-6;i<7;i++）**  **{**  每隔五十像素画条长为550像素的竖直直线。默认画笔为灰色虚线，当i=0；即绘制y轴时将画笔转为黑色实线，且每隔50像素再相应位置标上该竖线对应x的值。  **}**  **水平坐标线绘制原理同上**  当用户有对TextEdit进行输入并按下绘制按钮时，int 类型变量drawp的值会被  void MainWindow::**on\_pushButton\_clicked**()函数由0改为1  此时用户有函数输入，则该绘图函数会在经历上述绘制坐标轴的基础上，调用求解函数求出从-3.14开始到3.14结束x每次增加0.001的每个x对应的y值，x乘单位长度，y\*0.5个单位长度（y轴上每格两格之间的值相差2）后，绘图函数会在坐标轴相应像素点处标记该点，只要点的数量足够密集便可呈现出连续的曲线。（对于y值过大的点（即超出坐标轴范围的点，绘图函数会忽略） |
|  |

# 结果展示

对于本程序，我输入了如下函数进行测试：

**1.2\*sin(x)+3\*cos(x)+4.5\*x,测试结果如图3所示**

**2. 3.2\*x+sin(3\*x)/(2\*x) ,测试结果如图4所示**

**3.sin(x) ,测试结果如图5所示**

**4.cos(x) ,测试结果如图6所示**

**5.x\*x\*x,测试结果如图7所示**

**6.2\*x+6,测试结果如图8所示**

**7.x\*x\*+3\*x+6,测试结果如图9所示**

**8.sin(3\*x)+6,测试结果如图10所示**

**9.1/x,测试结果如图11所示**

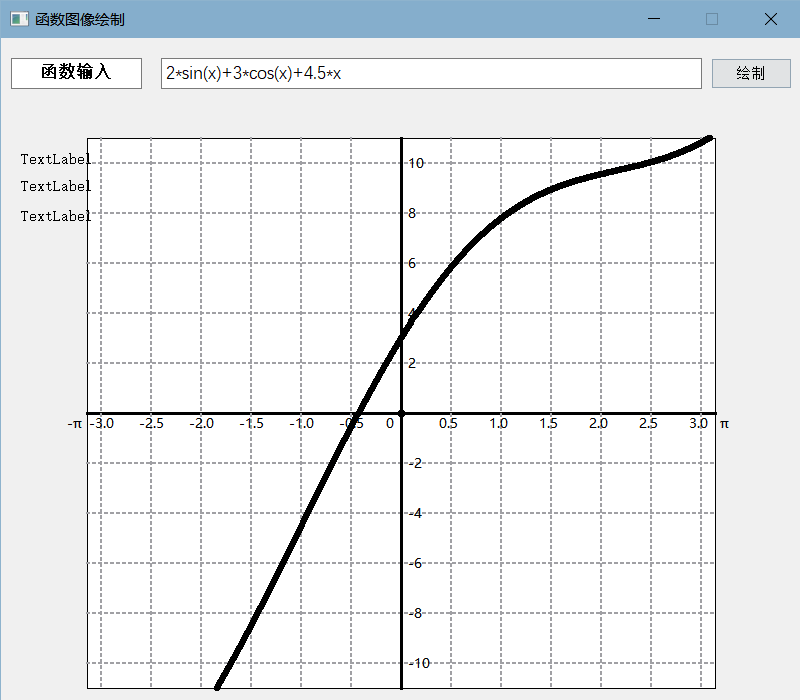
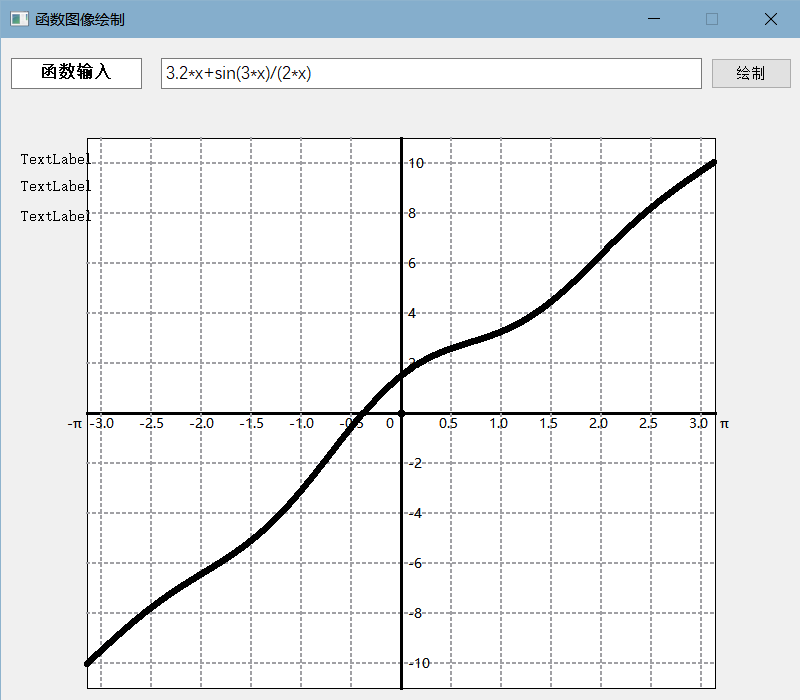
 

图 3 标准测试算例1：2\*sin(x)+3\*cos(x)+4.5\*x 图 4标准测试算例2：3.2\*x+sin(3\*x)/(2\*x)

其它测试：

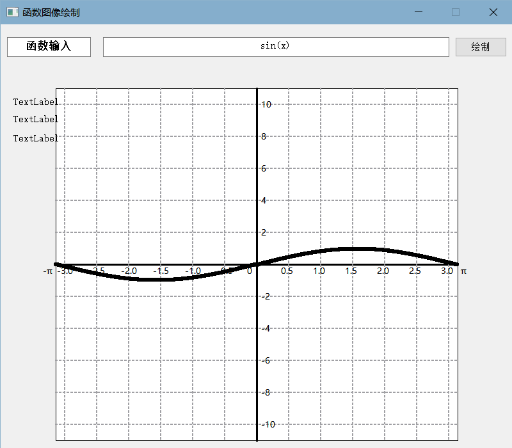
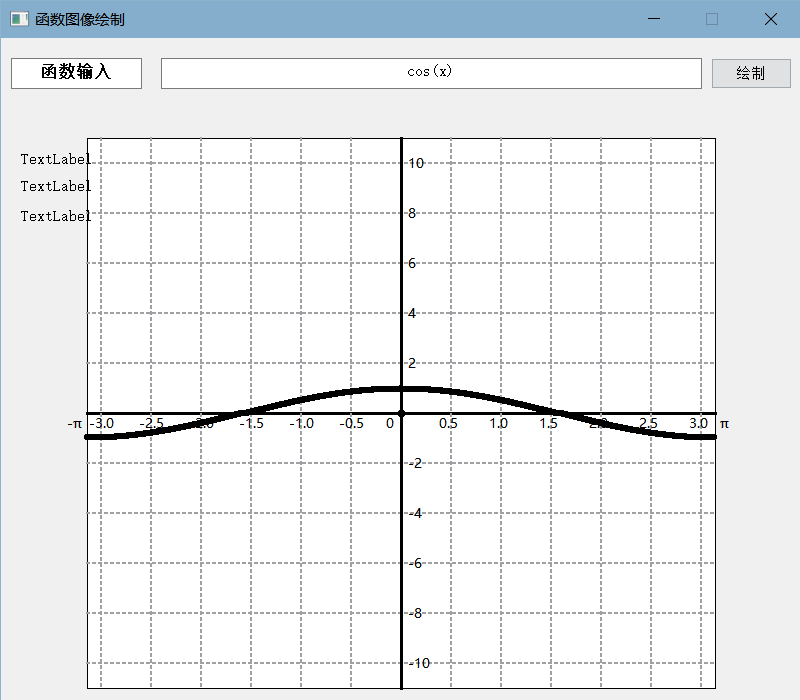
 

图 5 函数sin(x)的仿真结果 图 6 函数cos(x)的仿真结果

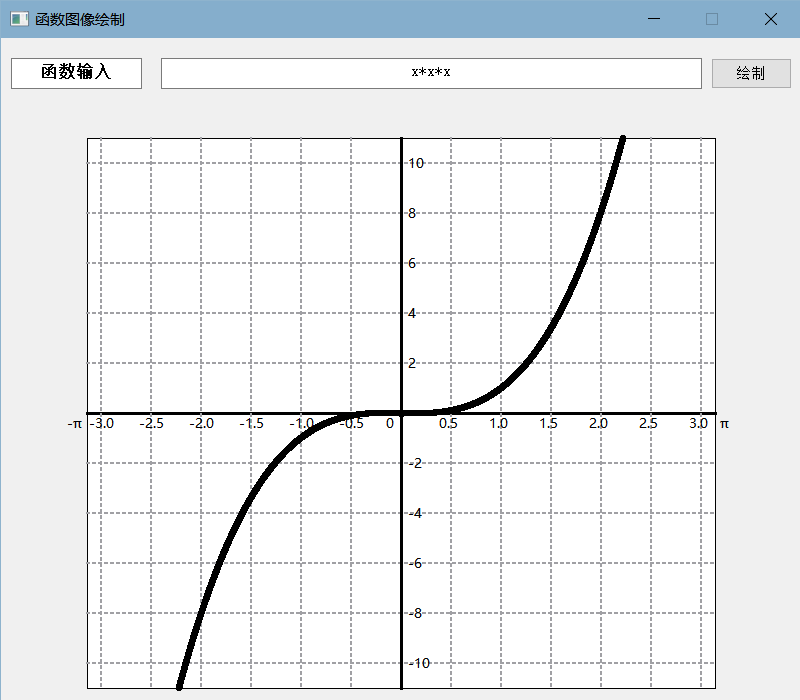
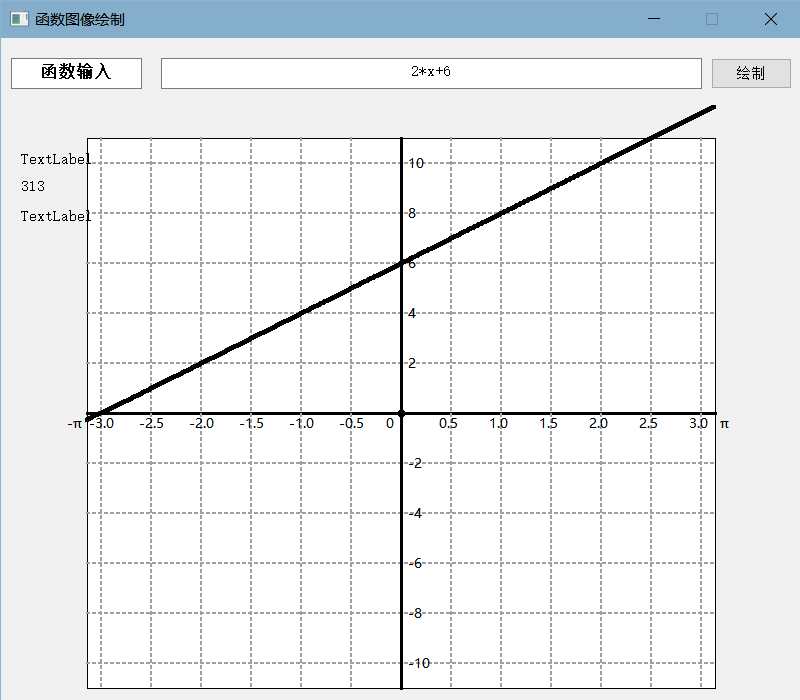
 

图 7 函数x\*x\*x的仿真结果 图 8 函数2\*x+6的仿真结果

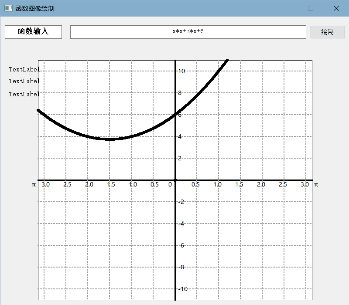
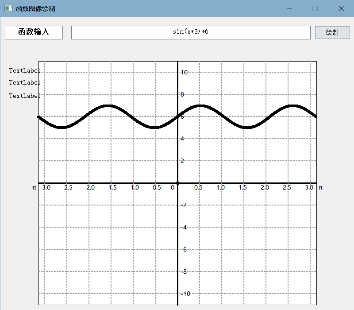
 

图 9 函数x\*x+3\*x+6的仿真结果 图 10 函数sin(3\*x)+6的仿真结果

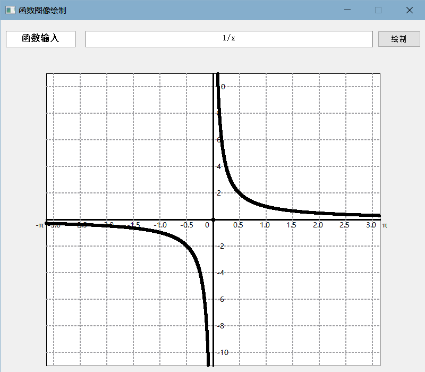


图 11 函数1/x的仿真结果

左侧三个textlabel是调试时用的，后期已删除。

1. **期间遇到的问题，心得，感想以及收获：**

这是首次使用qt进行较复杂的数据处理，期间遇到了一些问题也学到了一些新东西。

问题1：之前我的函数进行sin和cos运算总是结果与直接算相差较大，求解出来的点分布完全不连续，最后发现sin，cos内的数默认应该是弧度制，而且之前在for循环改变x的值时自动补全了int为改成double通过该次编程也是有温故而知新，并且在以后的学习中也会更加的细致。

问题2：运算优先级问题，这应该是该题的难处所在，经过一段时间的反复尝试，在寻找求解过程中我反复思考，人在看到这样一个式子后是如何处理得到答案的，在进行细致的分析，将抽象的东西具象化，于是想出了这种寻找最内层括号法，该方法正如我们人做题一样，先算出最里面括号的值，带回原式，继续算出当前最里面括号的值，层层求解，我认为在以后的编程中，这种站在计算机的角度分析人思考的方式也会是是一种好方法。

程序的制作过程中，我重写了很多遍程序，尝试了许多思路，从一开始的杂乱无章，到最后整齐的结构。在这个摸索过程中，我熟悉了软件工程中软件框架搭建的步骤，更明白了他的重要性。我也进一步的熟悉了通过qt进行编程的操作，也体会到了其进行图形界面编程的方便性。还有很重要一点就是，在开始一个很大的工程前，要先规划好软件框架，画出流程图，更要读清楚需求文档。这样可以减少重构的次数，一遍完成。

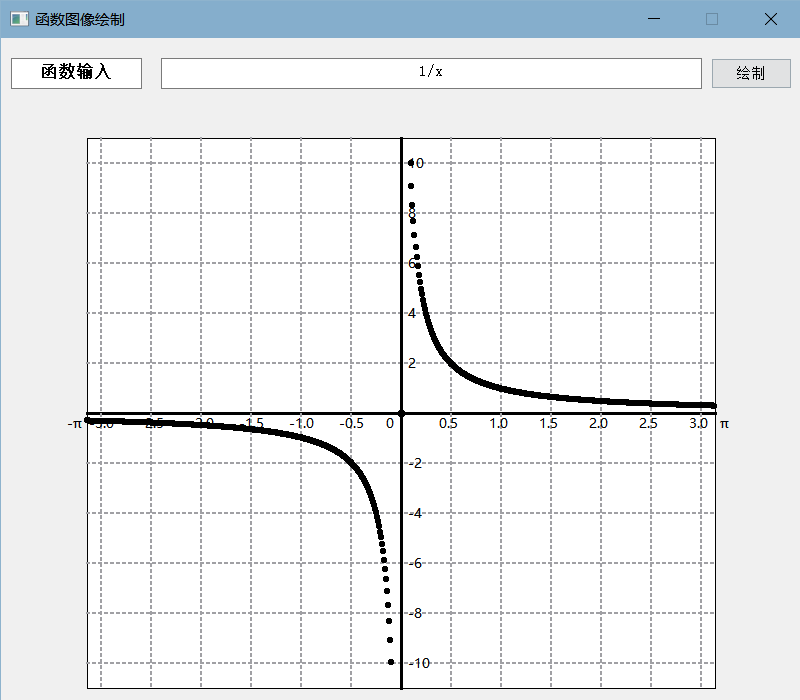
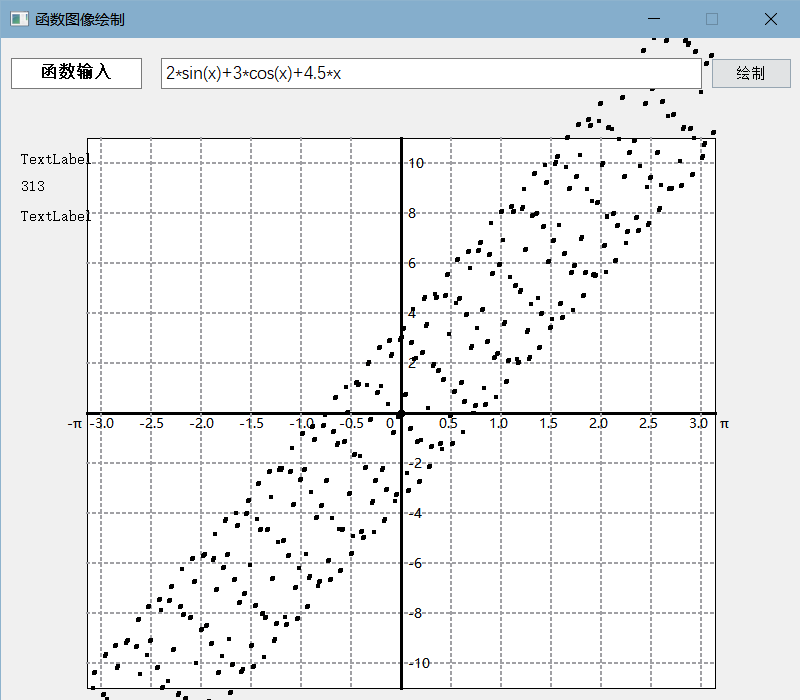
除此之外，还有一些小细节。在进行函数图像绘制等对数据精确度要求较高的程序编写时，在不会对效率有很大影响的前提下，最好尽可能的使用double类型，而不是int类型变量，并且需要往小数点后较多位进行精确，这样程序才能体现出其该有的效果，若精确度不够，**便出现了如下图 图12错误示例，图13错误示例所示的问题**。 

图 12 错误示例 图 13 错误示例