《迭代法解方程》设计报告

孙雨欣

2020 年 12 月 3 日

目录

1 摘要

本项目根据《数值分析》课程相关知识，通过运用 C++，Qt 等工具，演示用五种迭代方法解方程及图示过程。

2 系统概述

本项目将制作一个小程序，用户可以输入其需要的目标函数（仅支持含 x 的加、减、乘、除、幂运算，和 e 的 x 次幂相关运算）、希望进行迭代的等价形式、进行迭代的初值 x0 及弦截法需要的 x1，选择五中迭代法其一，得到函数图像，并通过单击按钮控制迭代过程，画出迭代过程点及输出中间值。本项目意在形象理解课程内容，并提高工程能力。考虑到期待通过本实验学习知识：

- 形象理解《数值分析》第二章五种迭代方法及其原理
- 栈、队列等数据结构的实际应用
- 基于 C++ 的面向对象的思想和方法，类的继承和封装等
- QT 的 UI 设计，并通过信号和槽机制实现前后端的交互
- 其他工具的使用，包括 cmake，git，markdown 等

2.1 相关链接

源代码仓库
用户手册
代码规范
博客地址，持续更新本项目相关的帖子
演示视频

2.2 文件目录说明

见源代码仓库。

3 需求分析

3.1 功能需求

- 可通过前端可视化界面和用户交互
- 绘制出函数图像
- 绘制出每一步迭代点
- 在窗口中显示每一步结果和详细信息
- 用户可通过按钮控制开始迭代结束迭代

3.2 性能需求

- 支持跨 Windows、Linux 平台运行
- 可靠性高，由于用户输入等问题产生错误，可以及时作出异常处理。
- 易操作性，简单易懂，容易上手
- 模块化设计，易于以后的维护和扩展

3.3 开发环境需求

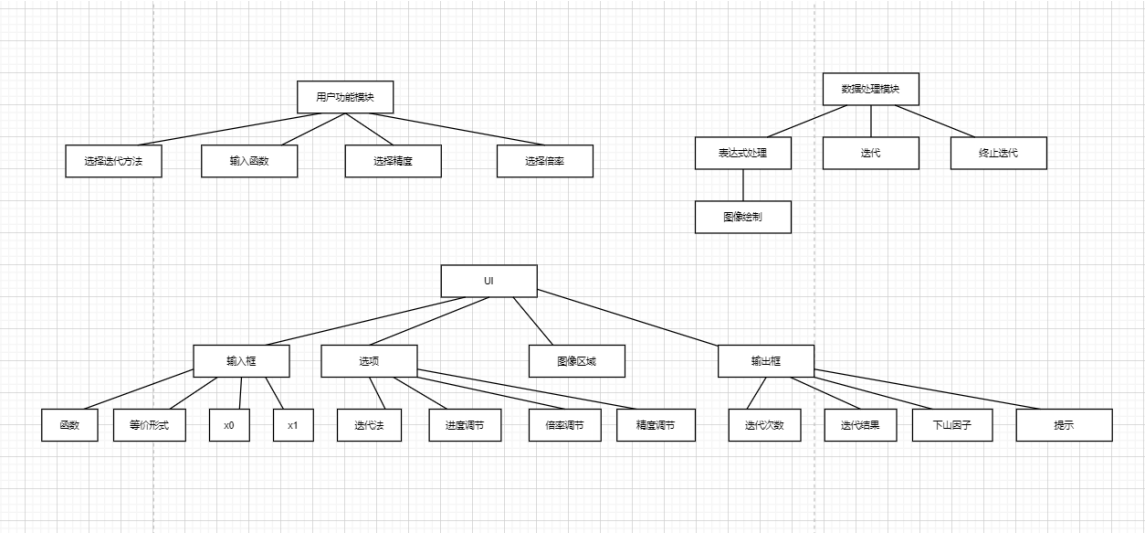
本实验采用：

- Linux ubuntu 18.04 操作系统
- vscode 编辑器
- cmake + make(Linux)
- cmake + vs(Windows)

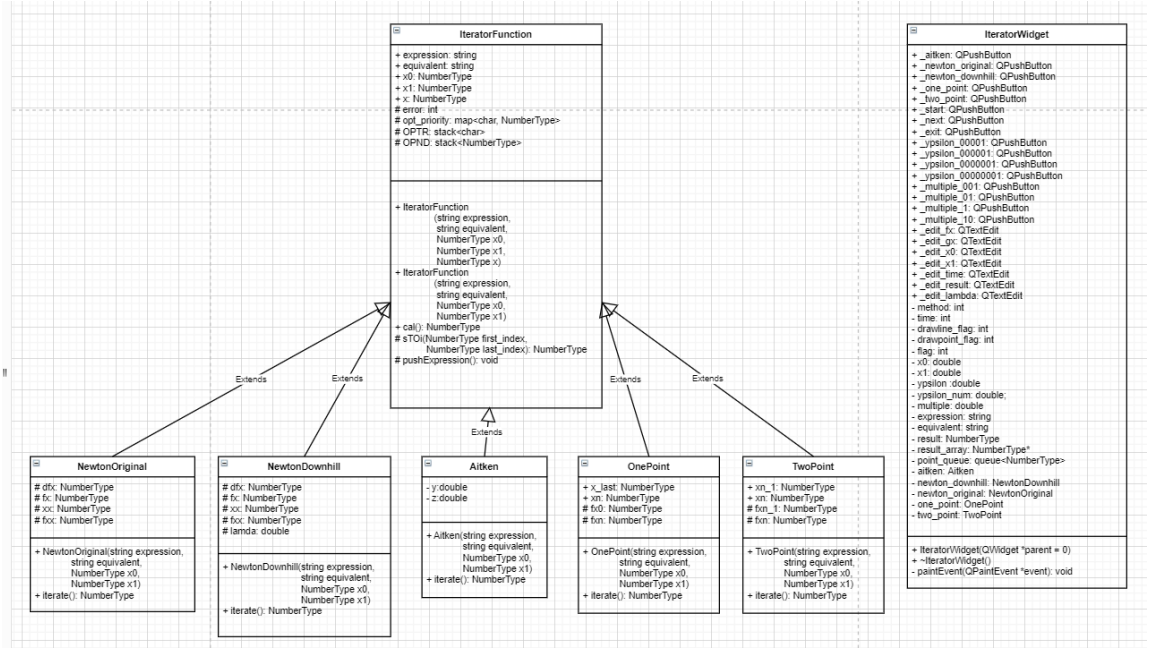
4 系统设计

4.1 系统总体模块图

4.1.1 模块划分



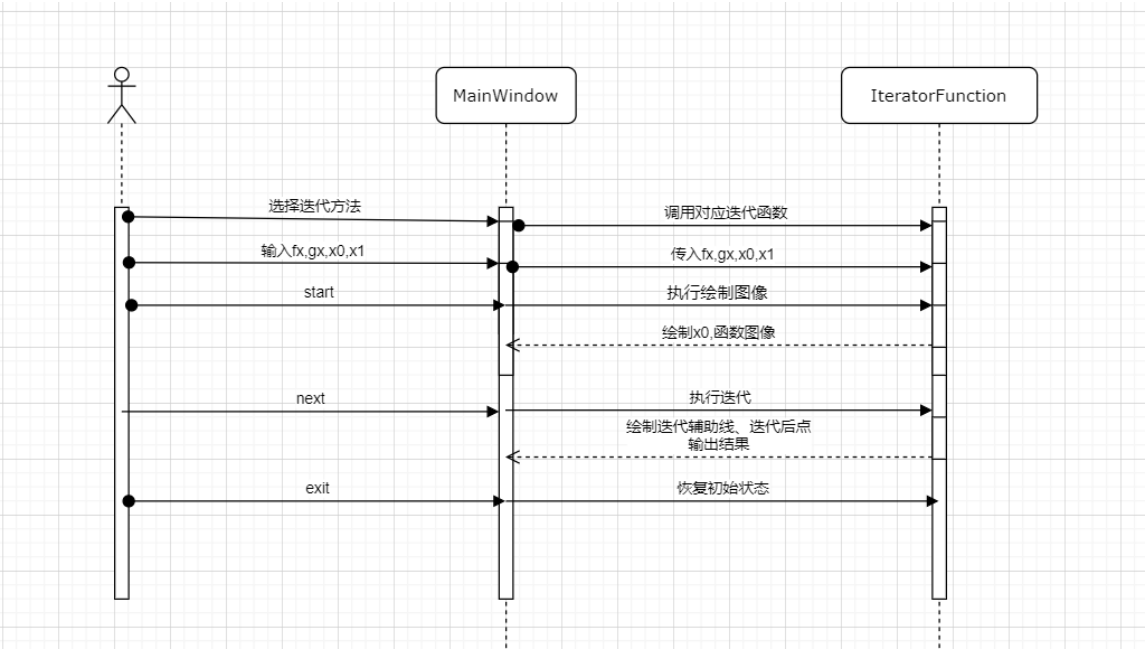
4.1.2 类设计



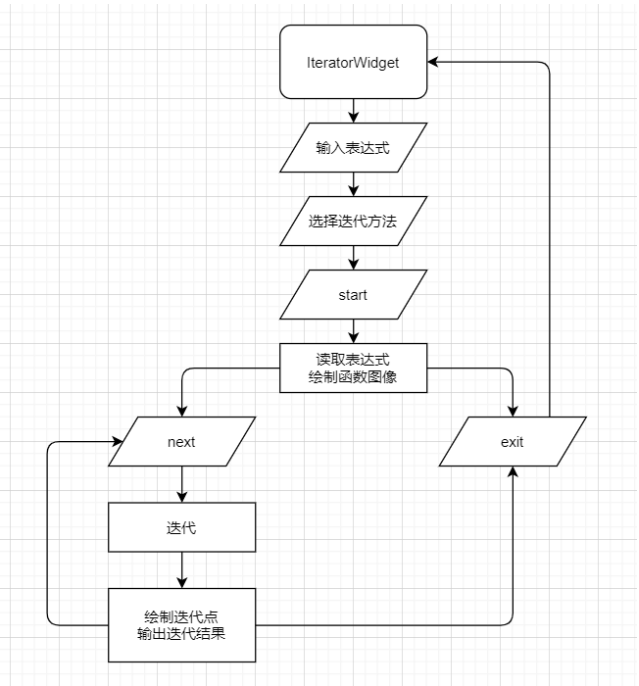
4.1.3 界面设计

4.2 软件动态模型设计

4.2.1 时序图



4.2.2 流程图



4.3 详细设计

4.3.1 读取表达式和初值

为不同的运算符号赋以不同的权值，根据算符优先算法确定进栈还是运算，即：

入栈操作 1、首先置操作数栈为空栈，表达式起始符“#”为运算符栈的栈底元素。

2、依次读入表达式中每个字符，若是操作数，则进数字栈；若是运算符，则与算符栈的栈顶运算符比较优先级后作相应操作。

直至整个表达式求值完毕（即算符栈的栈顶元素和当前读入的字符均为“#”）。

表达式处理 从左向右扫描表达式：

遇操作数——保存；

遇运算符号 aj——与前面的刚扫描过的运算符 ai 比较：

若 ai<aj 则保存 aj（因此已保存的运算符的优先关系为 a1<a2<a3<a4…）

若 ai>aj 则说明 ai 是已扫描的运算符中优先级最高者，可进行运算

若 ai=aj 则说明括号内的式子已计算完，需要消去括号

对于 x 和 e 的处理，我们可以直接将需要的 x 值代入。

4.3.2 绘制函数图像

将表达式计算结果进入队列，绘制图像时逐步出队并绘制。

为了能看到迭代过程，初值点应当放在图像正中间。那么为了清楚看到迭代过程，就画初值左右各一格的函数图像还是比较合适的。

另外，为了适应不同的图像，可以设置调节图像的放大倍率。

4.3.3 逐步迭代

每单击一次 next，调用一次对应的迭代函数，输出对应的点和结果。迭代满足精度要求时，输出“迭代成功!”

5 设计总结

5.1 收获

5.1.1 知识方面

- 对软件工程的大致流程有了较为清晰的认识
- 形象理解了《数值分析》第二章五种迭代方法及其原理，并有了一些有意思的发现
- 学会较灵活地应用栈
- 对类的继承和封装等面向对象原理有了更深刻的认识
- 从零开始学会了 Qt 库的使用
- 学会了 cmake 文档撰写，了解编译原理
- 手写代码：1200 行左右，提高了代码能力和 debug 能力

5.1.2 能力方面

尽管这是一个很简单的实现，最终我的收获却远比想象中的要大。通过本项目，我不仅学习和巩固了在系统概述中提到的基础知识和技能，还提高了配置环境、遇到问题并独立解决问题的能力。

- qt 库函数使用原理详解
- cmake 编译原理和过程
- 跨编辑器产生的字符编码问题
- 环境变量的调整和外来库的引入
- 撰写源码遇到的种种问题

以上问题中，部分是有明确报错、可以通过个人知识和百度搜索改正的，部分是报错内容不清晰且百度搜不到、甚至没有报错编译通过、运行调试中才出现问题的。对于后者，我首先借助已有知识和错误出现位置确定错误的大致位置，然后通过增删语句，cout 输出反复调试最终解决问题。在此过程中，我感觉我的潜力被充分激发，遇到 bug 时许多平时用不到的知识被充分调动，产生的直觉帮助我解决了许多问题。

由此可以看出知识储备的重要性。我们学到的基础知识不一定能切实的用到代码里，更多的体现在在遇到问题时发现并解决问题的能力。

5.2 反思

异常处理做的不够好。 由于五种迭代法被我直接封装成类，且与窗口类没有直接关系，所以窗口类使用的数据只有迭代函数的返回值，其中的 error 变量无法访问。因此比如表达式读取出现错误不会在窗口弹出提示。这是程序需要优化的一个重要部分。

软件效率不够高。 后期反思时，发现了许多代码中效率较低的地方。比如，读取表达式部分可以采用逆波兰表达式，五种迭代法进行初始化出现了部分的重复代码，窗口中重复实现画图功能等。

文档写的还是不够细致，没有勾勒出从头至尾的整个过程。 文档中，由于时间紧张，写代码过于着急，除了图是设计前画的，其余设计内容大多是后期反思的内容，难以还原我设计时的心路历程。

输入格式有局限性。 输入表达式如果含有小数，比如“0.6x2”,“0.6”作为三个字符，并且带小数点，不太好识别。鉴于暂时还不急于优化，暂且只允许以分数格式输入小数，以后有空再处理。本程序也无法处理 sinx,cosx,lnx 等问题。

界面没有美化 时间紧张，没有做 ui，所有内容都是在 IteratorWidget 类手写，一点一点计算并调整位置。因而界面不仅没有 Ui 且不够美观，这也是日后设计软件需要注意的一点。