《迭代法解方程》设计报告

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。

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^x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

相关链接

- [源代码仓库]https://github.com/fighterkaka22/iterator_for_equation
- [用户手册]https://github.com/fighterkaka22/iterator_for_equation/blob/dev/documents/md/%E7%94%A8%E6%88%B7%E6%89%8B%E5%86%8C.md
- [代码规范]https://github.com/fighterkaka22/iterator_for_equation/blob/dev/documents/md/%E4%BB%A3%E7%A0%81%E8%A7%84%E8%8C%83.md
- [博客地址，持续更新本项目相关的帖子]https://www.cnblogs.com/fighterkaka22/category/1881452.html

目录说明

```
.
├── build //Linux和Windows下的编译中间文件和可执行程序
│   ├── linux //除了Iterator外，其余均是编译链接过程产生文件，可以忽略
│   │   └── Iterator //Linux下的可执行文件，可在装有Qt5.9.2以上版本的Linux系统下运行
│   └── windows //除了Debug文件夹中内容，其余均是编译链接过程产生文件，可以忽略
│       └── Debug //添加了Windows下运行该程序需要的Qt库和平台文件，因而可以在无Qt的Windows系统运行
│           ├── Iterator.exe //Windows下的可执行文件，可以直接运行
│           ├── Iterator.ilk
│           ├── Iterator.pdb
│           ├── platforms
│           ├── Qt5Cored.dll
│           ├── Qt5Guid.dll
│           └── Qt5Widgetsd.dll
├── documents //工程相关文档，分为.markdown和.pdf格式，便于阅读
│   ├── md
│   │   ├── 代码规范.md
│   │   ├── 用户手册.md
│   │   └── 设计报告.md
│   └── pdf
│       ├── 代码规范.pdf
│       ├── 用户手册.pdf
│       └── 设计报告.pdf
├── library
│   ├── CMakeLists.txt //CMake文档，用于生成Windows下的.sln工程和Linux下的makefile
│   ├── IteratorFunction //迭代函数类，内涵读取表达式并计算代码及五种迭代法底层代码
│   │   ├── Aitken.cpp
│   │   ├── Aitken.h
│   │   ├── IteratorFunction.cpp
│   │   ├── IteratorFunction.h
│   │   ├── NewtonDownhill.cpp
│   │   ├── NewtonDownhill.h
│   │   ├── NewtonOriginal.cpp
```

```
│   │   ├── NewtonOriginal.h
│   │   ├── OnePoint.cpp
│   │   ├── OnePoint.h
│   │   ├── TwoPoint.cpp
│   │   └── TwoPoint.h
│   ├── IteratorWidget //窗口类，实现前后端交互
│   │   ├── IteratorWidget.cpp
│   │   └── IteratorWidget.h
│   └── main.cpp //主函数
├── pictures //工程相关图示，分为.drawio格式和.png格式
│   ├── drawio
│   │   ├── 时序图.drawio
│   │   ├── 模块划分.drawio
│   │   ├── 流程图.drawio
│   │   ├── 界面设计.drawio
│   │   └── 类图.drawio
│   └── png
│   ├── 时序图.png
│   ├── 模块划分.png
│   ├── 流程图.png
│   ├── 界面设计.png
│   └── 类图.png
├── README.md //仓库说明，实际上是设计报告，便于在github上阅读
├── 设计报告.pdf //设计报告，放在最外层便于用户直接阅读
└── 演示视频.mp4 //演示视频
```

需求分析

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

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

开发环境需求

本实验采用：

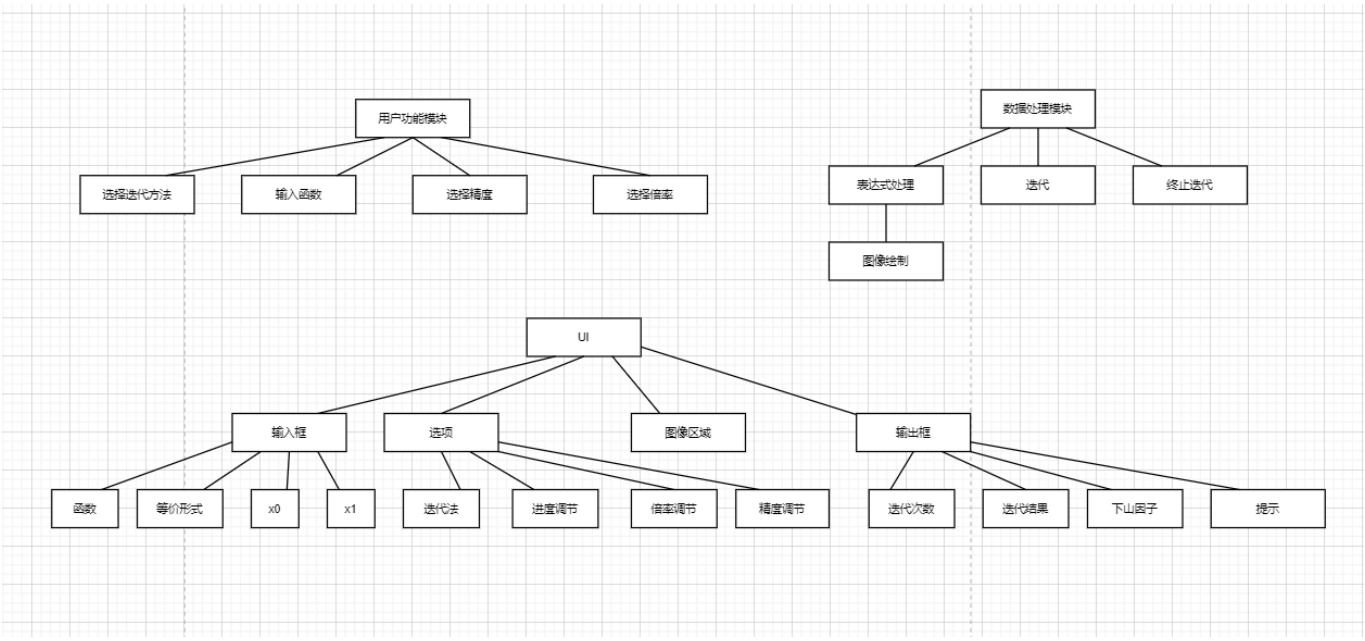
- Linux ubuntu 18.04 操作系统

- vscode编辑器
- cmake + make(Linux)
- cmake + vs(Windows)

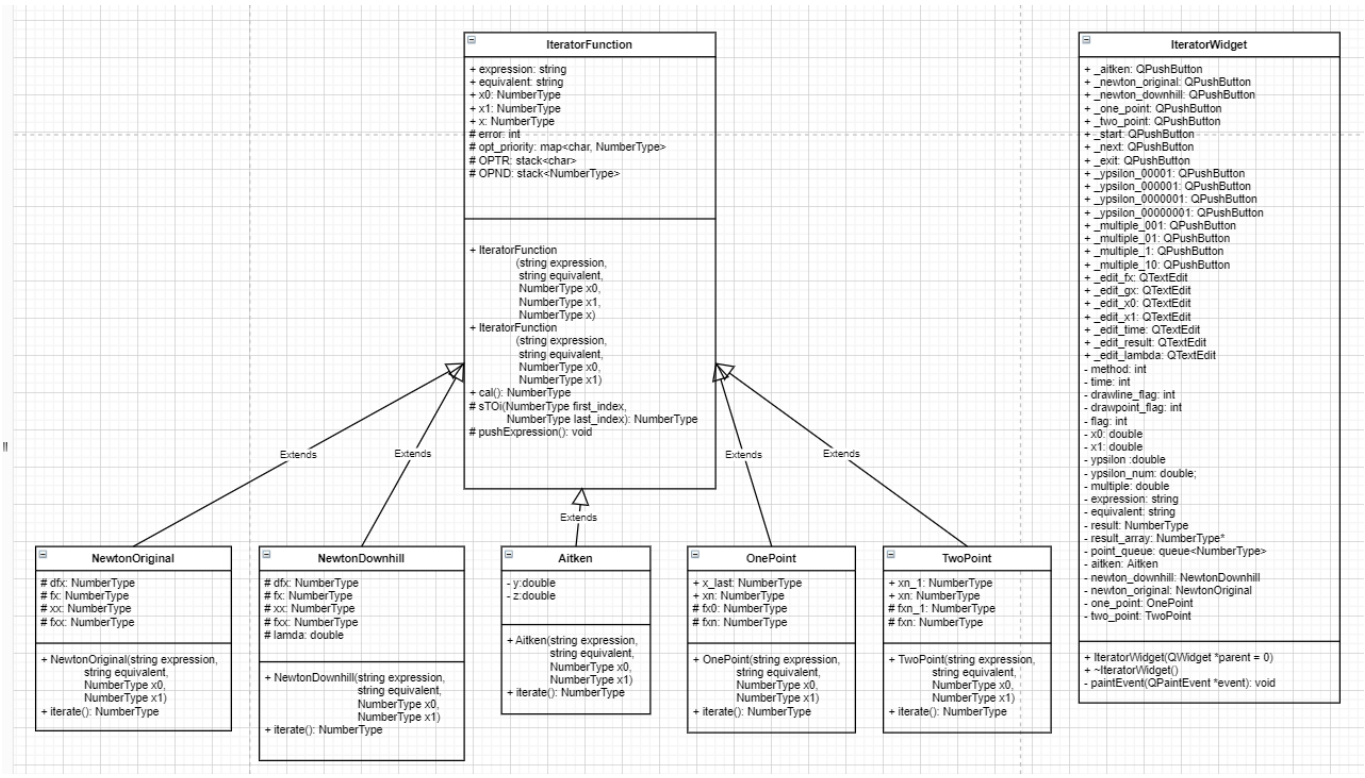
系统设计

系统总体模块图

模块划分



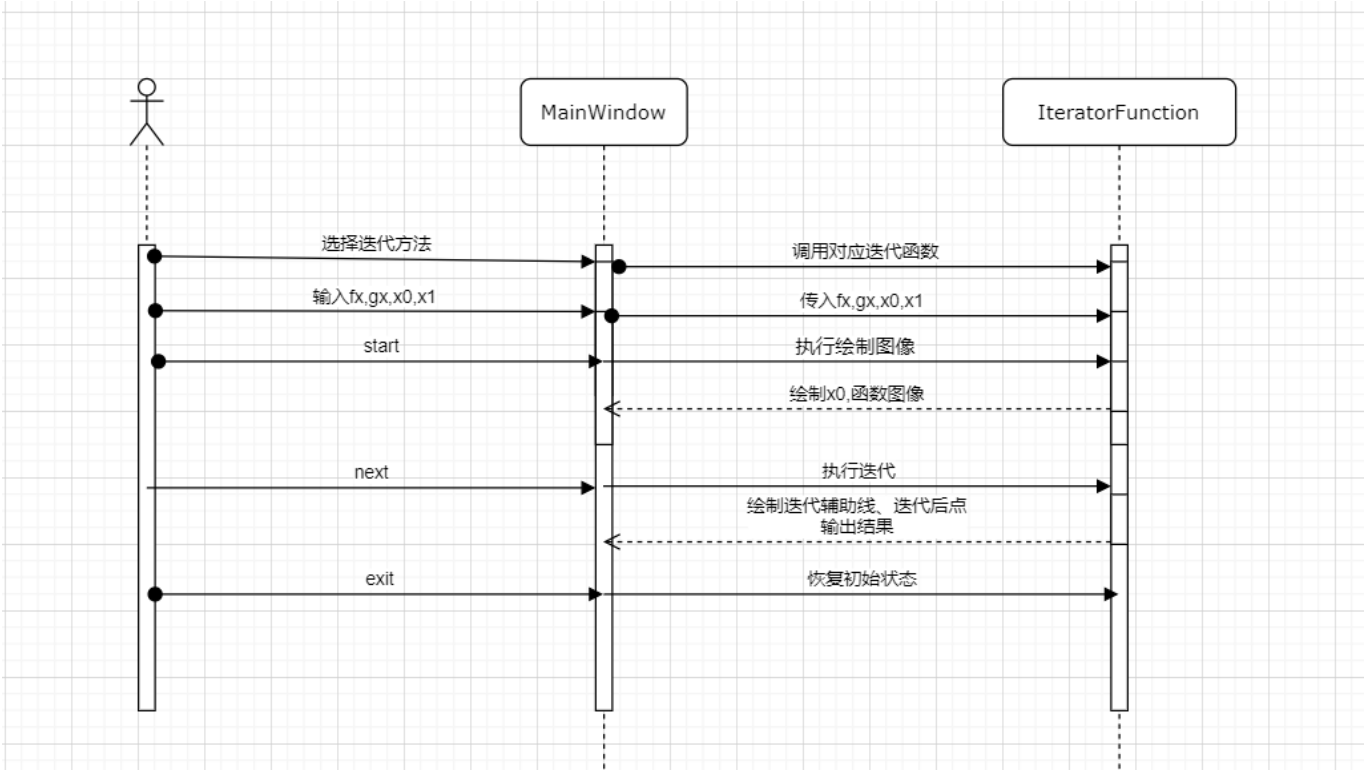
类设计



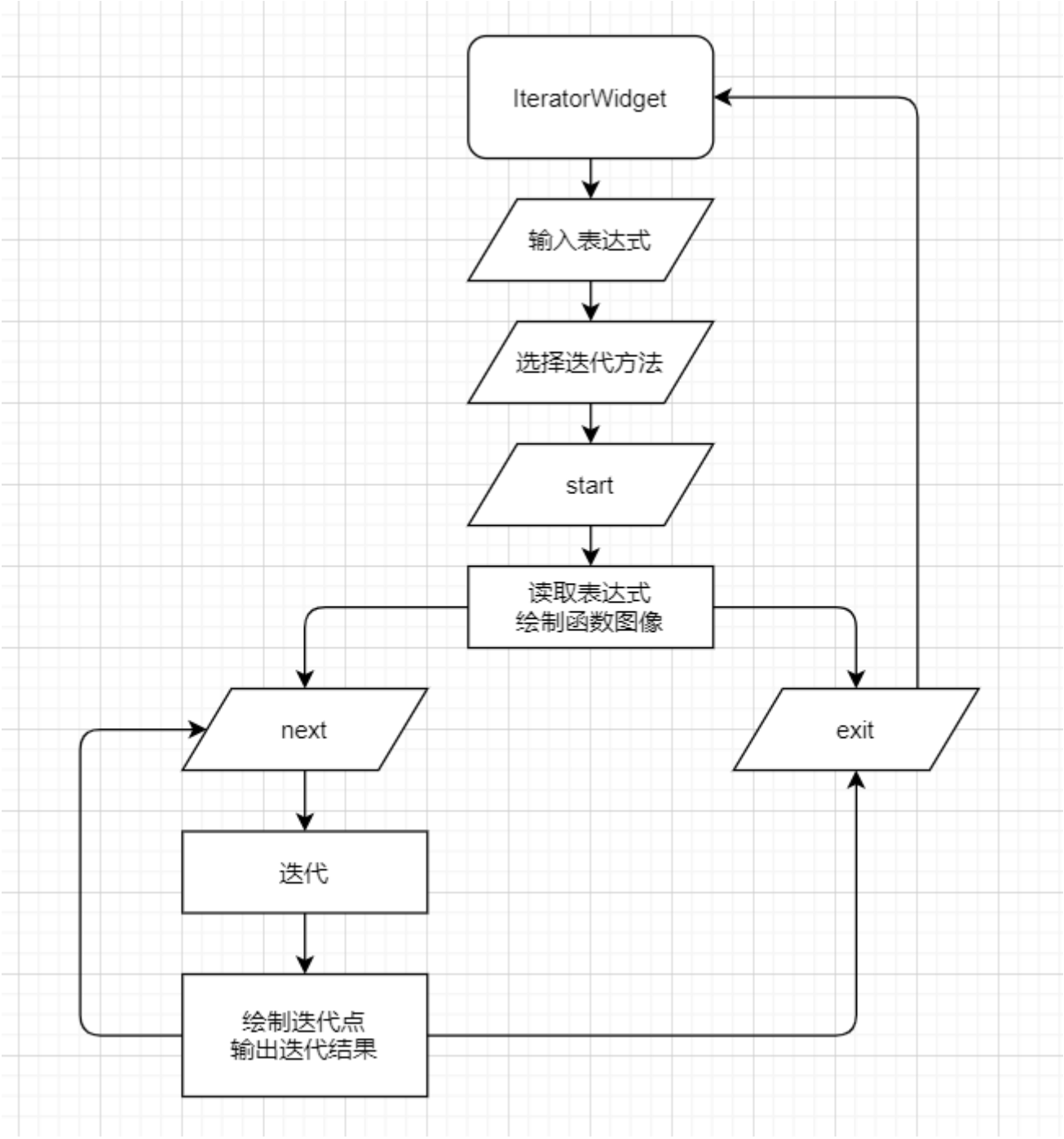
界面设计

软件动态模型设计

时序图



流程图



详细设计

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。

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。

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！”

设计总结

收获

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

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。
设计过程中遇到并解决的问题：

- 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。

反思

- 最大的缺陷：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。
  但由于暂时还没有学最优化相关的内容，这里暂且不着急。
- 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“0.6x^2”,“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。