Python 结课作业项目报告

Vespera Zephyr

2025 年 5 月 5 日

项目选题：实用工具类.

目录

1 需求分析

- 支持常见微积分运算
- 直观的公式渲染能力
- 友好的图形界面和帮助选项
- 较快的计算速度

1.1 功能需求

核心功能	实现要求
符号计算	支持求导、积分、极限、化简等操作
输入验证	实时检测非法输入并提示
结果展示	双模式显示（文本 + 公式渲染）
历史保存	计算结果可保存为结构化文本
帮助系统	内置符号输入规范说明

2 设计思路

2.1 架构设计

采用 MVC 模式实现：

- **Model**: Sympy 符号计算引擎
- **View**: PyQt6 图形界面
- **Controller**: 事件驱动逻辑控制

引入第三方库：

```python
import sys
from datetime import datetime
from PIL import Image
from sympy import *
from sympy.parsing.sympy_parser import parse_expr
from PyQt6.QtWidgets import (
  QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
  QLineEdit, QPushButton, QLabel, QComboBox, QGridLayout,
  QTabWidget, QFrame, QScrollArea, QMessageBox, QFileDialog,
  QDialog, # 用于创建对话框
  QScrollArea # 滚动区域
)
from PyQt6.QtWebEngineWidgets import QWebEngineView
from PyQt6.QtCore import QThread, pyqtSignal, pyqtSlot
from PyQt6.QtGui import QAction
```

2.2 关键技术

- 多线程计算：QThread 防止界面冻结
- 动态界面：GridLayout 实现字段动态加载
- LaTeX 渲染：MathJax + QWebEngineView
- 输入验证：正则表达式 + 异常处理链

3 核心代码说明

3.1 计算线程（CalculationThread）

```
class CalculationThread(QThread):
  finished = pyqtSignal(str, str) # (full_latex, full_text)

  def __init__(self, operation, data):
    super().__init__()
    self.operation = operation
    self.data = data
    def run(self):
      try:
        x, y, z, t = symbols('x y z t')
        full_latex = ""
        full_text = ""

        ...（这里是多个用户操作，包括求导、极限、
        不定积分、定积分与反常积分、化简）

      except Exception as e:
        self.finished.emit("", f"计算错误：{str(e)}")
```

3.2 帮助系统（HelpDialog）

```
class HelpDialog(QDialog):
    def __init__(self):
      super().__init__()
```

```
        self.setWindowTitle("符号输入规则说明")
        self.setFixedSize(800, 600)

        layout = QVBoxLayout()

        # 使用WebEngineView显示帮助内容
        self.web_view = QWebEngineView()
        self.web_view.setHtml(self.help_content())

        # 添加滚动区域
        scroll = QScrollArea()
        scroll.setWidget(self.web_view)
        scroll.setWidgetResizable(True)

        layout.addWidget(scroll)
        self.setLayout(layout)

        # 关闭按钮
        close_btn = QPushButton("关闭")
        close_btn.clicked.connect(self.accept)
        layout.addWidget(close_btn)

        self.setLayout(layout)

    def help_content(self):
        return f"""
            <html>
            <!-- MathJax配置 -->
            <script>MathJax = {{...}};</script>
            <!-- 排版样式 -->
            <style>body {{font-family: 华文楷体;}}</style>
            <!-- 帮助内容 -->
            <h1>符号输入规则说明</h1>
            ...
            </html>
            """
```

3.3 主窗口程序（QMainWindow）

为节省篇幅，这里省略一些代码细节设置，以函数名称等为主，省略部分有些会注释.

```
class MathAssistantPro(QMainWindow):
  def __init__(self):
    super().__init__()
    self.setup_ui()
    self.setup_style()
    self.setWindowTitle("一元微积分计算器")
    self.setGeometry(100, 100, 800, 600)
    self.current_operation = ""
    self.extra_fields = {}
    self.setup_help_button()
    self.setup_save_button() # 初始化保存按钮
    self.current_result = ("", "") # 保存最新结果（latex, text）
    def setup_help_button(self):
    """在顶部工具栏添加帮助按钮"""
    def setup_save_button(self):
      """添加保存按钮到工具栏"""

    def show_help(self):
      dialog = HelpDialog()
      dialog.exec()
    def setup_ui(self):
      main_widget = QWidget()
      self.setCentralWidget(main_widget)
      main_layout = QVBoxLayout()
      main_widget.setLayout(main_layout)
      # 操作选择区
      self.operation_combo = QComboBox()
      operations = ["求导", "不定积分", "定积分与反常积分", "极限",
          "化简"]
      self.operation_combo.addItems(operations)
      self.operation_combo.currentTextChanged.connect(self.update_input_fields)
      main_layout.addWidget(self.operation_combo)

```

```
33      # 动态输入区
34      self.dynamic_input_frame = QFrame()
35      self.dynamic_layout = QGridLayout()
36      self.dynamic_input_frame.setLayout(self.dynamic_layout)
37      main_layout.addWidget(self.dynamic_input_frame)
38
39      # 公共输入区
40      self.expression_input = QLineEdit()
41      self.expression_input.setPlaceholderText("输入数学表达式,
42                                               例如: exp(x)*sin(x)")
43      main_layout.addWidget(self.expression_input)
44
45      # 按钮组
46      btn_layout = QHBoxLayout()
47      self.calc_btn = QPushButton("开始计算")
48      self.calc_btn.clicked.connect(self.validate_inputs)
49      self.render_btn = QPushButton("渲染表达式")
50      self.render_btn.clicked.connect(self.render_expression)
51      btn_layout.addWidget(self.calc_btn)
52      btn_layout.addWidget(self.render_btn)
53      main_layout.addLayout(btn_layout)
54
55      # 结果展示区
56      result_tabs = QTabWidget()
57      self.text_result = QLabel()
58      self.text_result.setWordWrap(True)
59      self.web_view = QWebEngineView()
60      self.web_view.setHtml(self.base_html(""))
61      result_tabs.addTab(self.text_result, "文本结果")
62      result_tabs.addTab(self.web_view, "公式渲染")
63      main_layout.addWidget(result_tabs)
64
65      # 初始化动态字段
66      self.init_dynamic_fields()
67
68  def init_dynamic_fields(self):
69      """创建所有可能的额外输入组件"""
```

```
70
71    def update_input_fields(self, operation):
72        """根据操作类型显示对应输入字段"""
73        self.clear_dynamic_layout()
74
75        row = 0
76        if operation in ["求导", "不定积分"]:
77        self.add_dynamic_row(row, 'variable')
78        row += 1
79
80        ...其余省略
81
82    def add_dynamic_row(self, row, field_key):
83        """添加一行动态输入组件"""
84        label = getattr(self, f"{field_key}_label")
85        field = getattr(self, f"{field_key}_field")
86        self.dynamic_layout.addWidget(label, row, 0)
87        self.dynamic_layout.addWidget(field, row, 1)
88
89    def clear_dynamic_layout(self):
90        """清空动态布局"""
91
92    def validate_inputs(self):
93        """收集并验证输入参数"""
94        operation = self.operation_combo.currentText()
95        data = {'expression': self.expression_input.text().strip()}
96
97        if not data['expression']:
98            QMessageBox.critical(
99            self,
100           "输入错误",
101           "必须输入数学表达式！",
102           QMessageBox.StandardButton.Ok
103           )
104           return # 阻止后续执行
105
106       # 收集额外参数
```

```
107     if operation in ["求导", "不定积分", "定积分与反常积分", "极限"]:
108       data['variable'] = getattr(self,
              'variable_field').text().strip() or 'x'
109
110     if operation == "定积分与反常积分":
111       data['lower'] = getattr(self, 'lower_field').text().strip() or
              '0'
112       data['upper'] = getattr(self, 'upper_field').text().strip() or
              '1'
113
114     if operation == "极限":
115       data['point'] = getattr(self, 'point_field').text().strip() or
              '0'
116
117   # 启动计算线程
118     self.start_calculation(operation, data)
119
120   @pyqtSlot(str, str)
121   def handle_result(self, full_latex, full_text):
122     self.calc_btn.setEnabled(True)
123     self.calc_btn.setText("开始计算")
124     self.current_result = (full_latex, full_text) # 存储最新结果
125     if full_latex:
126       self.web_view.setHtml(self.base_html(full_latex))
127       self.text_result.setText(f"计算结果: \n{full_text}")
128     else:
129       self.text_result.setText(full_text)
130       self.web_view.setHtml(self.base_html(""))
131
132   def start_calculation(self, operation, data):
133     """启动计算线程"""
134     if not data['expression']:
135       self.show_result("", "请输入数学表达式! ")
136       return
137
138     self.calc_btn.setText("计算中...")
139     self.calc_btn.setEnabled(False)
```

```
140
141     self.thread = CalculationThread(operation, data)
142     self.thread.finished.connect(self.handle_result)
143     self.thread.start()
144
145   def render_expression(self):
146     """仅渲染表达式"""
147     expr = self.expression_input.text().strip()
148     if not expr:
149       return
150
151     try:
152       parsed = parse_expr(expr)
153       self.web_view.setHtml(self.base_html(latex(parsed)))
154       self.text_result.setText(f"原始表达式: {str(parsed).replace('**',
          '^')}")
155     except Exception as e:
156       self.text_result.setText(f"渲染错误: {str(e)}")
157
158     def base_html(self, latex_code):
159       return f"""
160       <html>
161       <head>
162         添加MathJax支持
163       </script>
164       </head>
165       <body>
166       <div style="font-size: 20px; padding: 20px; background:
          #C1FFC1;">
167       $${latex_code}$$
168       </div>
169       <p>不定积分默认省略常数\\(C\\)</p>
170       </body>
171       </html>
172       """
173
174   def setup_style(self):
```

```
175     self.setStyleSheet(...) #这里是样式设置
176   def save_to_txt(self):
177     """保存结果到文本文件"""
178     if not self.current_result[0] and not self.current_result[1]:
179       QMessageBox.critical(self, "保存错误", "没有可保存的结果！")
180       return
181
182     # 获取保存路径
183     file_path, _ = QFileDialog.getSaveFileName(
184     self,
185     "保存计算结果",
186     "",
187     "文本文件 (*.txt)"
188     )
189
190     if not file_path: # 用户取消选择
191       return
192
193     try:
194       # 组织保存内容
195
196       with open(file_path, "w", encoding="utf-8") as f:
197         f.write(content)
198
199         QMessageBox.information(self, "保存成功",
              f"结果已保存至：\n{file_path}")
200
201     except Exception as e:
202       QMessageBox.critical(
203       self,
204       "保存失败",
205       f"文件保存失败：\n{str(e)}"
206       )
```

4 运行截图

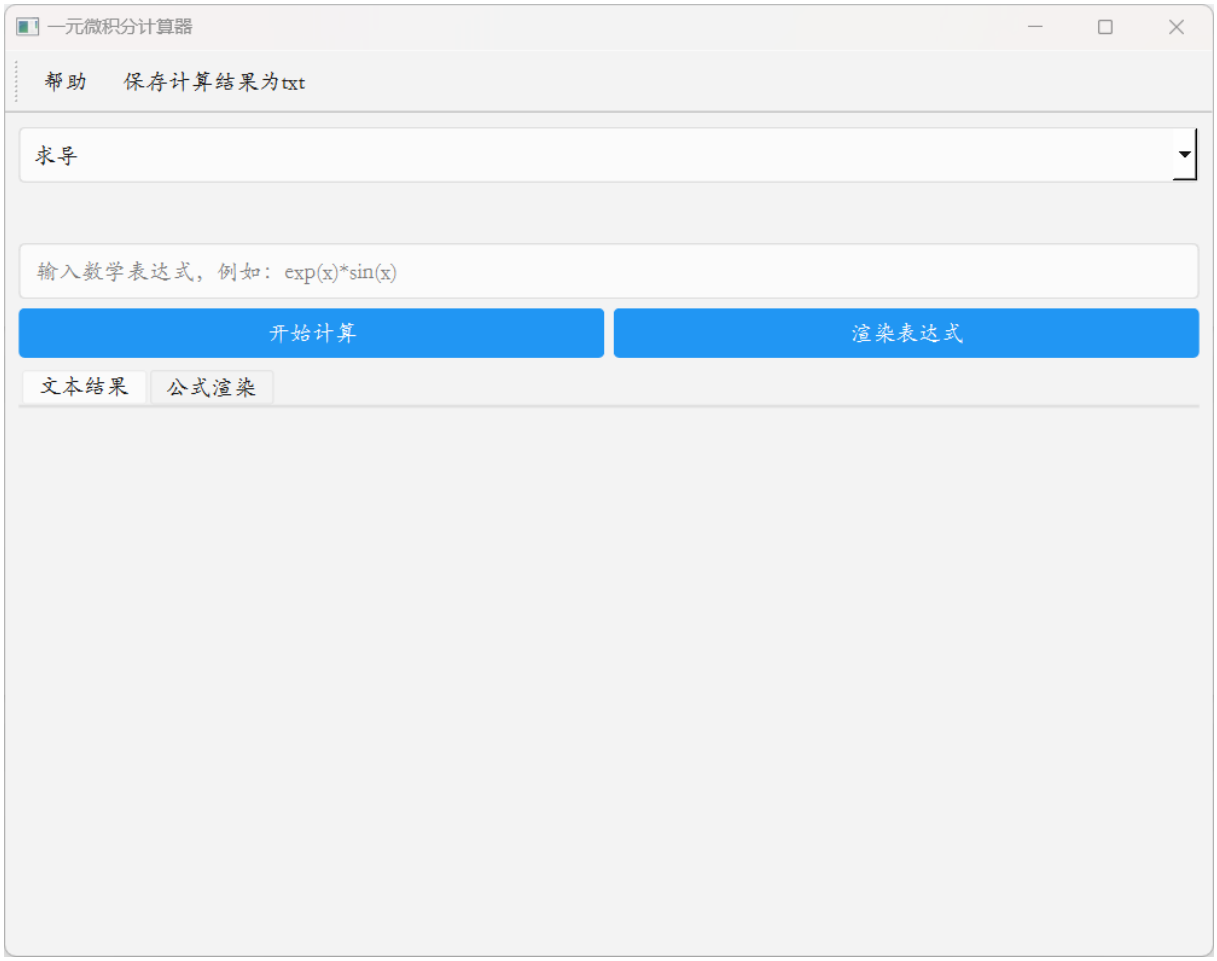

图 1: 主界面

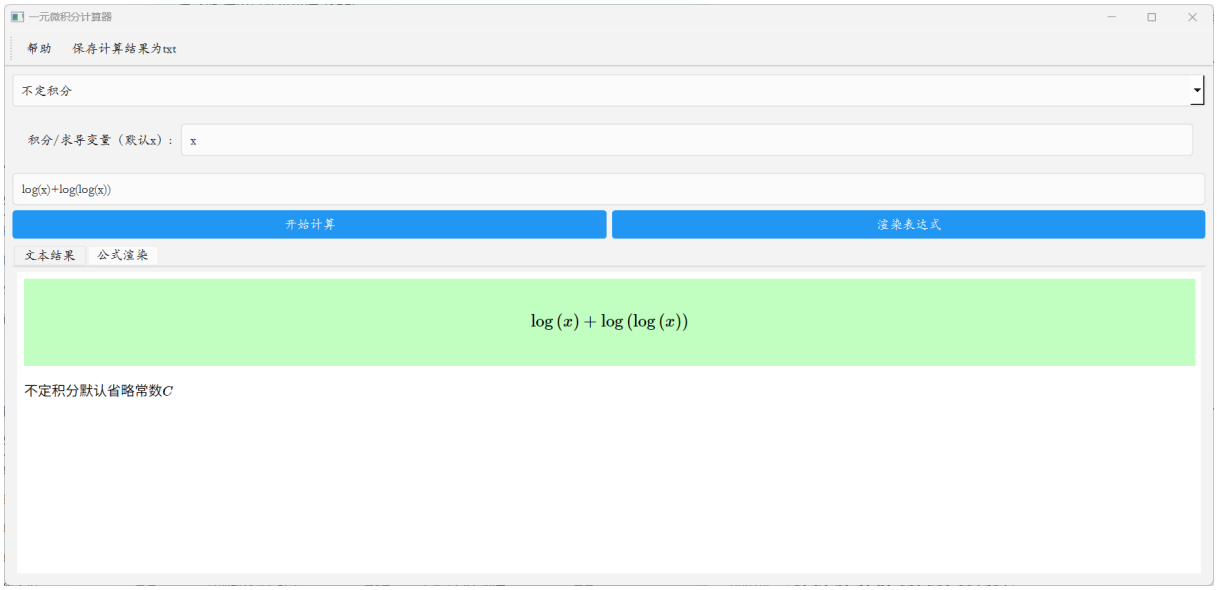

图 2: 公式渲染

图 3: 求导计算：文字结果

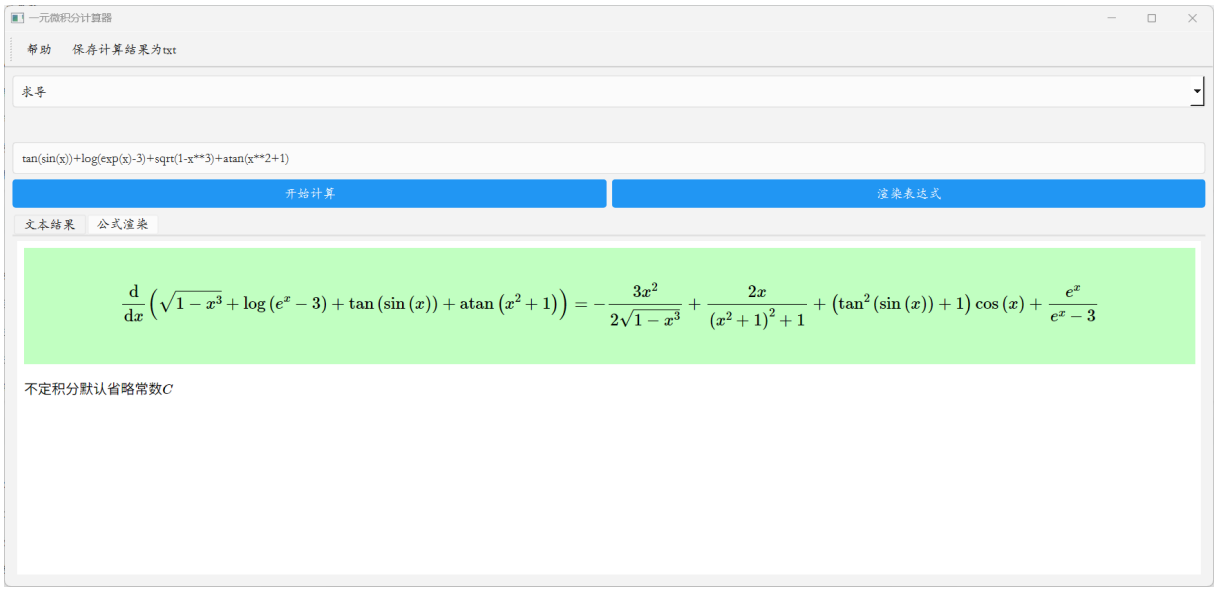

图 4: 求导计算：公式渲染

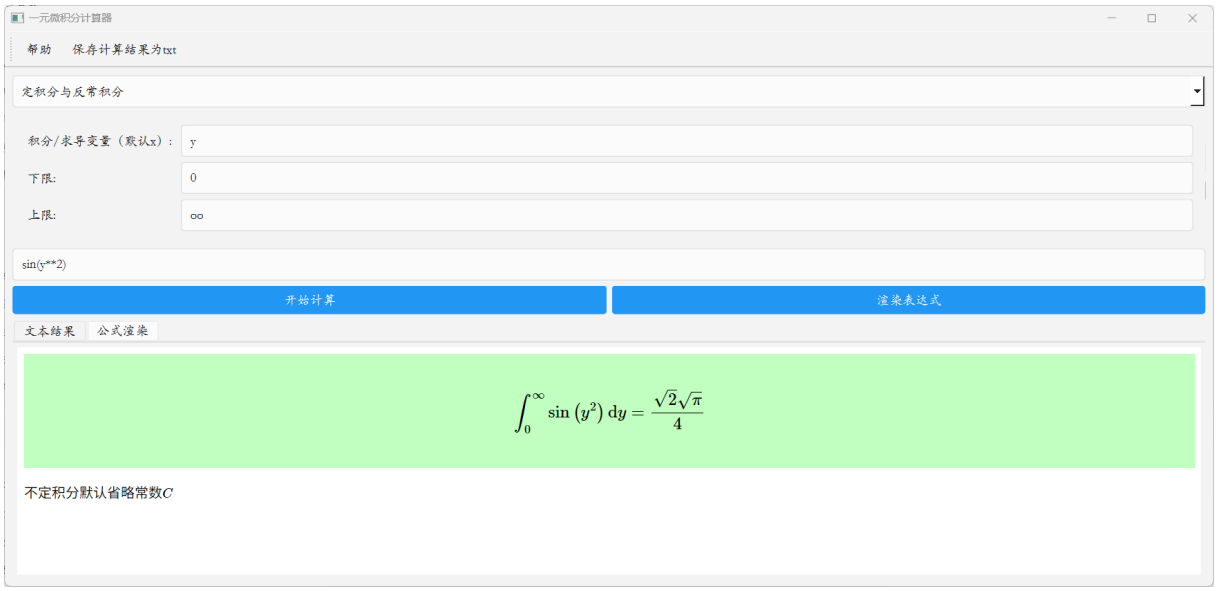

图 5: 定积分与反常积分计算：公式渲染

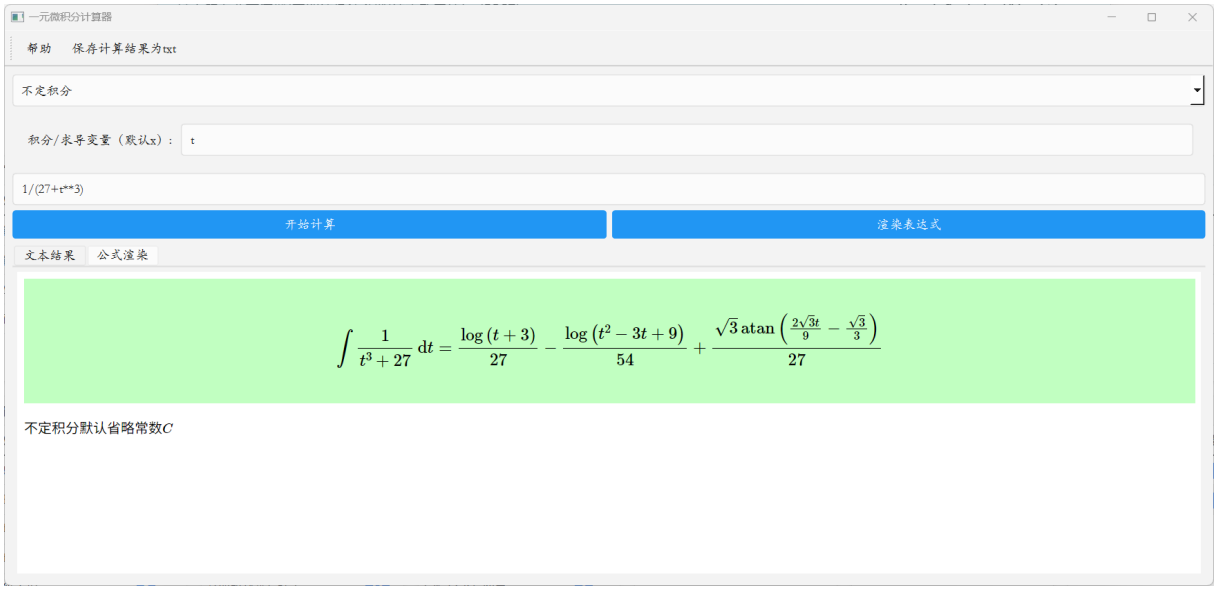

图 6: 不定积分计算：公式渲染

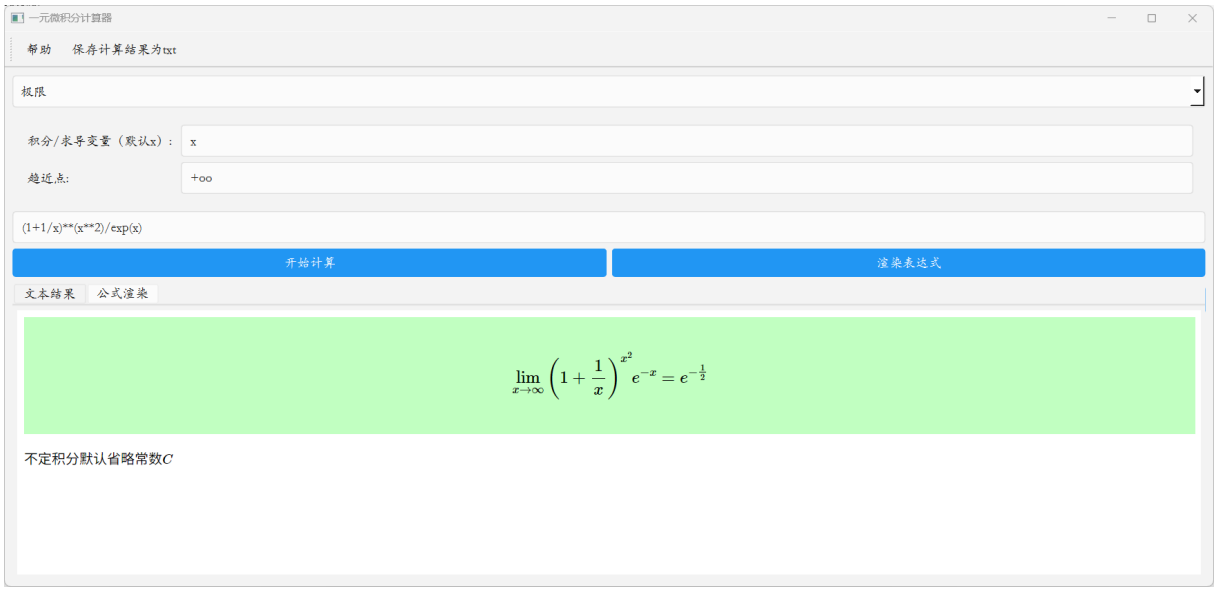

图 7: 极限计算：公式渲染

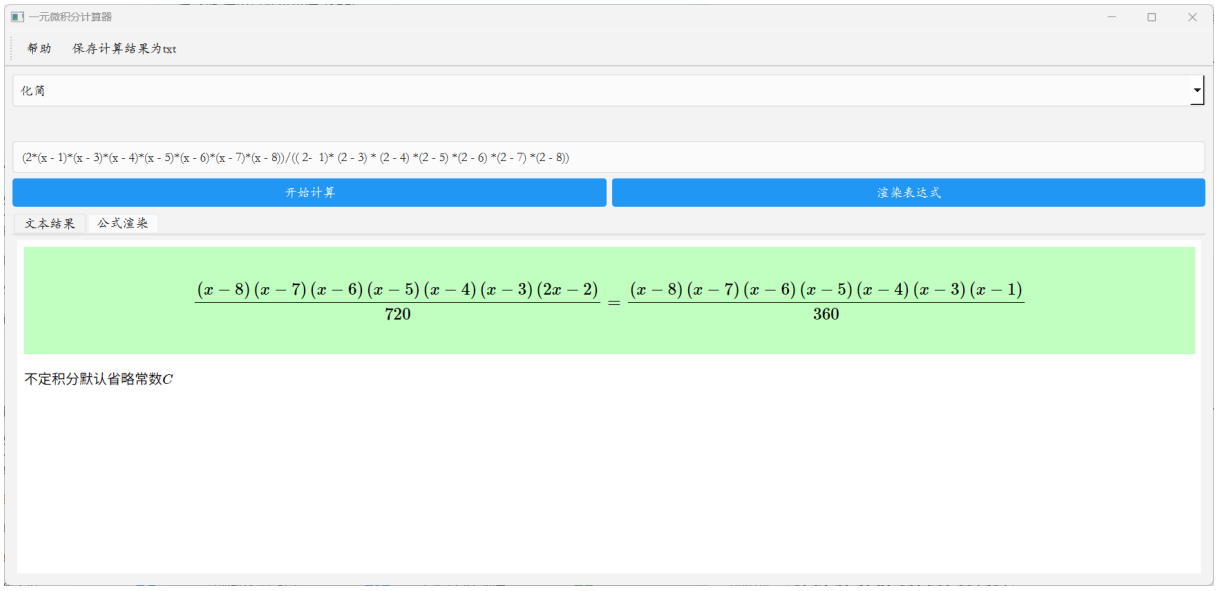

图 8: 化简

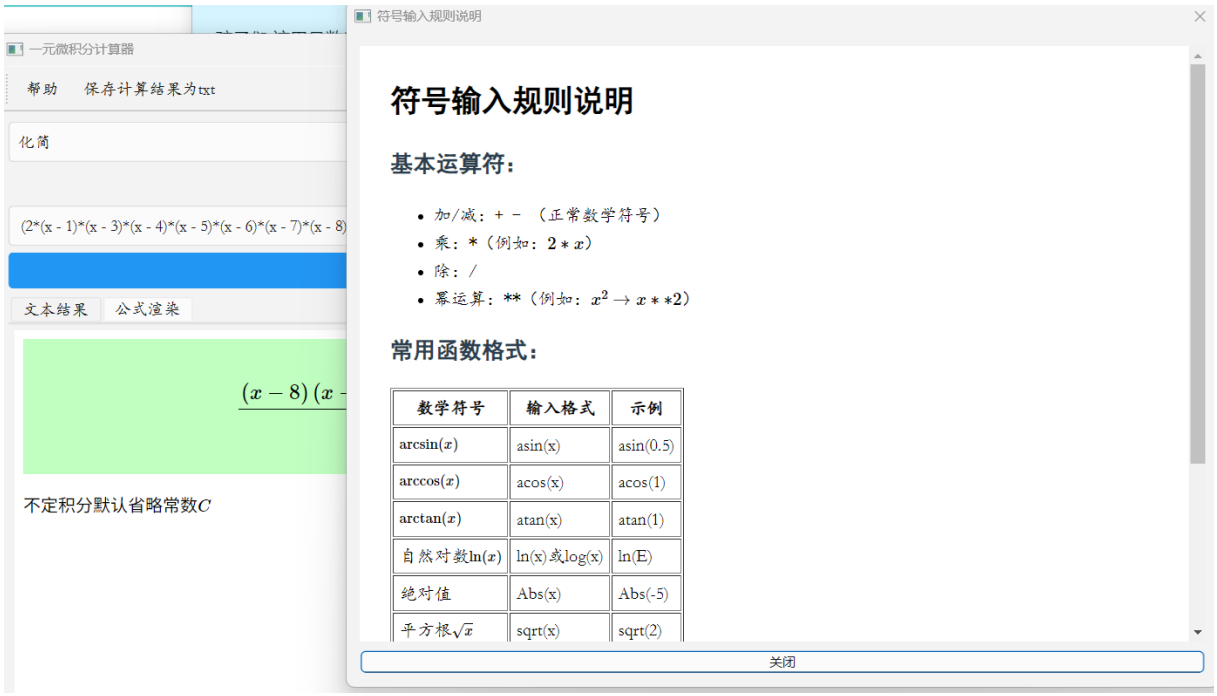

图 9: 帮助界面

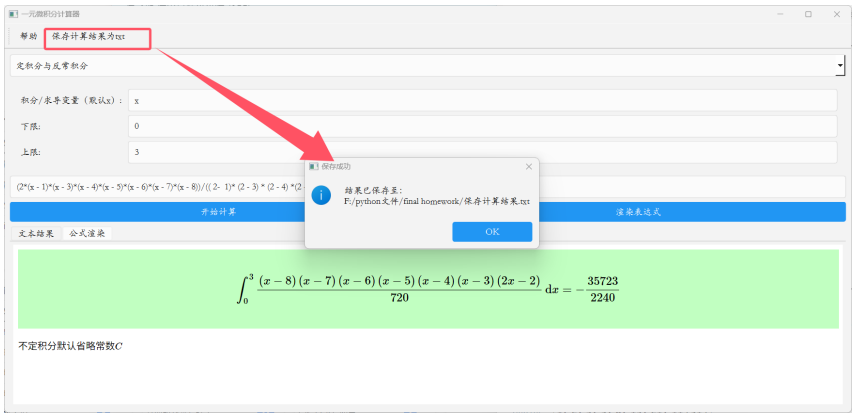

图 10: 保存计算结果为 txt 文件

5 总结与改进方向

5.1 项目成果

- 完整实现 5 类微积分运算（一元函数的极限、求导、不定积分、定积分与反常积分）与符号化简
- 开发响应式 GUI 界面
- 建立符号输入规范体系

5.2 缺点与改进方向

1. 有些较为复杂的积分计算无法算出，如

$$\int_0^{\frac{\pi}{2}} \ln\sin x \,\mathrm{d}x = -\frac{\pi \ln 2}{2}$$

无法计算，

$$\int_0^1 \frac{\arctan t}{t} \,\mathrm{d}t = \boldsymbol{G}$$

也无法计算，这里的 $\boldsymbol{G}$ 是 Catalan 常数.

2. 有些计算结果不是最简形式，如

$$\int_0^{\pi/4} \tan x \,\mathrm{d}x = \frac{\ln 2}{2},$$

输出结果为 $-\log\left(\frac{\sqrt{2}}{2}\right)$.

3. 不定积分计算没有加常数 C. 关于一些化简计算不会原封不动地输出原始输入.

4. 有些函数的输入不太常用，如 $\arctan(x)$ 也写成 $\mathrm{atan}(x)$

5. 所有自然对数的输出都是 $\log x$ 而不是常用的 $\ln x$.

6. 仅支持 x, y, z, t 作为变量；

7. **功能扩展**：支持的功能还不够丰富，但是没时间制作了，还可以完善如下内容

   (a) 函数在某点的 Taylor 级数展开；

   (b) 数值计算；

   (c) 判断表达式的真假；

   (d) 解方程，例如求解多项式的根和某些微分方程；

   (e) 线性代数功能，例如线性方程组的求解，矩阵的加法、数乘、乘法、转置、逆，判断矩阵的秩，计算方阵的行列式和特征值等.

8. **性能优化**：引入计算缓存机制

9. **可视化增强**：支持 2D/3D 图形绘制