

babylonian算法求平方根 - 代码使用说明

1. 问题描述

本项目旨在通过**巴比伦算法 (Babylonian Method)**，又称海伦法 (Heron's Method) 或牛顿迭代法 (Newton's Method)，数值计算一个正数 y 的平方根。

题目需要解决以下核心问题：

- 算法实现**：编写函数实现迭代计算，直到误差满足给定的精度 ϵ ，返回近似解、迭代步数、真实误差等。
- 收敛性分析**：探究误差 ϵ_n 与迭代步数 n 之间的关系。
- 初值敏感性**：分析不同的初始猜测值 x_0 对收敛速度的影响。
- 工程化改进**：实现基于相邻项误差 (Cauchy收敛准则) 的停止条件，以模拟真实场景 (即不知道真实值的情况)。

2. 算法原理与解答思路

2.1 迭代公式

求解 \sqrt{y} 等价于求解方程 $x^2 - y = 0$ 的正根。应用牛顿迭代法，其递推公式为：

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{y}{x_n} \right)$$

其中：

- y ：目标数值 (被开方数)。
- x_n ：第 n 步的近似解。
- x_0 ：初始猜测值 (通常取 1 或其他正数)。

2.2 误差定义

代码中使用了两种误差定义方式：

- 真实误差 (True Error)**： $|x_n - \sqrt{y}|$ 。用于评估算法相对于真实值的准确度 (需预知真实值)。
- 相邻误差 (Adjacent Error)**： $|x_{n+1} - x_n|$ 。用于在未知真实值的情况下判断算法是否收敛。

3. 代码功能模块详解

代码分为四个主要任务模块 (Task)，使用了 `matplotlib` 进行可视化分析。

Task 1: 基础算法实现

定义函数 `babylonian(y, epsilon, x0)`。

- 输入**：目标值 y ，目标精度 ϵ ，初始值 x_0 。
- 逻辑**：循环执行迭代公式，直到真实误差小于 ϵ 或达到最大迭代次数。
- 输出**：最终近似解 x_n ，消耗步数 n ，最终误差。

Task 2: 精度与步数关系分析

批量测试从 10^{-1} 到 10^{-15} 的不同精度要求。

- **可视化**: 绘制 **迭代步数 (n) vs 真实误差 (ϵ_n)** 的半对数曲线。
- **结论**: 该算法具有**二阶收敛**特性, 误差随步数增加呈指数级下降 (收敛速度极快)。

Task 3: 初始值 x_0 的影响

设定目标值 $y=10$, 测试不同初始值 $x_0 \in [0.1, 0.5, 1, 5, 10, 50, 100]$ 。

- **可视化**: 在同一张图上绘制不同 x_0 下的误差收敛曲线。
- **结论**:
 - 算法对初值不敏感, 无论 x_0 远大于还是小于真实值, 最终都能收敛。
 - x_0 越接近真实值, 前期的“纠正”步骤越少。

Task 4: 基于相邻误差的改进

定义函数 `babylonian2(y, epsilon, x0)`。

- **改进点**: 停止条件改为 $|x_{n+1} - x_n| < \epsilon$ 。
- **输出**: 返回迭代过程中的完整数组 `{xn}` 和 `{epsilon_n}`。
- **可视化**: 绘制相邻误差随迭代步数的变化趋势。

4. 快速开始 (Usage Guide)

4.1 环境依赖

运行此代码需要安装 Python 3 及以下科学计算库:

- `numpy`
- `matplotlib`

安装命令:

```
pip install numpy matplotlib
```

4.2 运行代码

步骤 1: 导入库

```
import math
import matplotlib.pyplot as plt
import numpy as np

# 设置绘图风格 (可选)
plt.style.use('seaborn-v0_8-whitegrid')
# 解决中文显示问题
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode_minus'] = False
```

步骤 2: 调用基础计算 (Task 1)

```
# 示例: 计算根号 10, 精度要求 1e-7  
sol, steps, err = babylonian(y=10, epsilon=1e-7, x0=1)  
print(f"近似解: {sol}, 步数: {steps}")
```

步骤 3: 运行分析图表

直接运行 Notebook 中的 Task 2、Task 3 和 Task 4 代码块, 即可生成分析图表。

5. 注意事项

1. **除零保护**: 虽然代码逻辑健壮, 但如果 x_0 设为 0 会导致除零错误。建议保持 $x_0 > 0$ 。
2. **精度极限**: 计算机浮点数精度通常在 10^{-16} 左右 (双精度)。如果 `epsilon` 设置过小 (如 $1e-20$), 算法可能会因无法达到精度而进入死循环 (代码中已设置 `max_iter=1000` 防止此类情况)。
3. **绘图乱码**: 代码中包含了设置中文字体 (SimHei) 的配置, 如果在非 Windows 系统或未安装该字体的环境中运行出现乱码, 请注释掉 `plt.rcParams['font.sans-serif']...` 相关行。