计算物理作业 2

杨远青 22300190015

2024 年 9 月 20 日

1 题目 1：三次方程根的求解与精确化

1.1 题目描述

Sketch the function $\boldsymbol{x^3 - 5x + 3 = 0}$

(i) Determine the two positive roots to 4 decimal places using the bisection method.
Note: You first need to bracket each of the roots.

(ii) Take the two roots that you found in the previous question (accurate to 4 decimal places) and "polish them up" to 14 decimal places using the Newton-Raphson method.

(iii) Determine the two positive roots to 14 decimal places using the hybrid method.

1.2 程序描述

标准的求根问题，顺便练习一下 C++. 先使用 Mathematica® 画出函数草图如下：

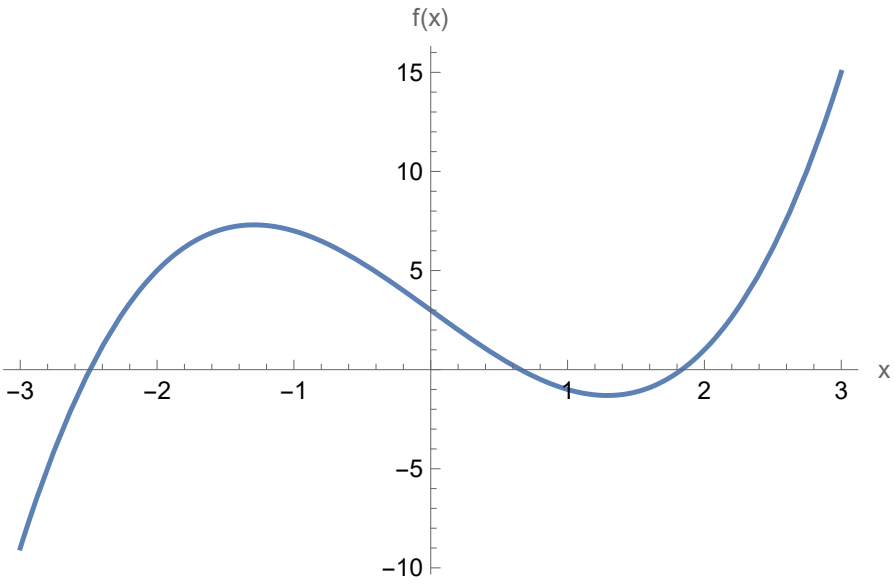

图 1: Plot of $x^3 - 5x + 3 = 0$.

发现有两个正根，分别在 $[0,1]$ 和 $[1,2]$ 之间，有点好奇二分法的具体实现，便先用 Python 写一下二分法的动态过程，见`./Codes/Problem 1/bisection_visual.py`，使用`python -u bisection_visual.py`运行（需要安装`matplotlib,numpy`库）。运行后有三个选项，分别是前两个正根的查找与自定义区间查找，可以自行选择。

`./Codes/Problem 1/`中还有 C++ 实现的二分法、牛顿法、混合法、Brent 法与 Ridder 法，算法实现集成在了`methods.cpp`中，`main.cpp`负责封装与交互，`plotting.cpp`用 ASCII 绘制了草图，`functions.cpp`里面存储了本题函数与导数，拎出来是为了便于灵活测试其它函数，`utils.cpp`里面有两个工具函数，分别负责按照题目三小问的顺序输出结果与对比测试各类方法在三个根上的表现。使用`g++ *.cpp -o main`编译，`./main`运行（也有已经编译好的`main.exe`），按照提示可以选择各类方法或者一起比较，也可以自定义查找区间与容差等参数。

1.3 伪代码

1.4 结果示例

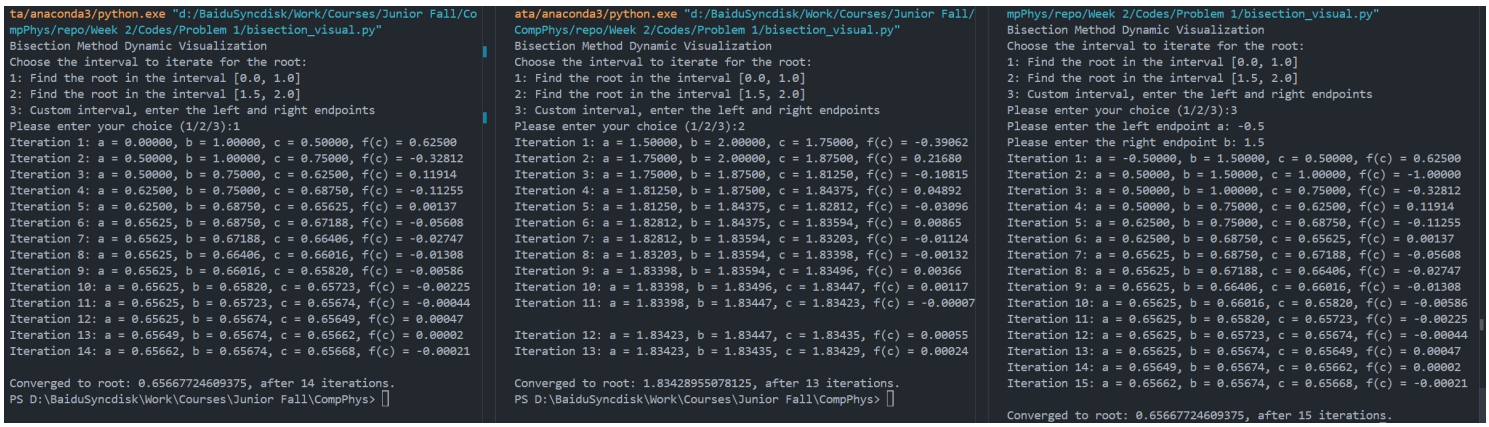

图 2: `bisection_visual.py` 三种选项对比

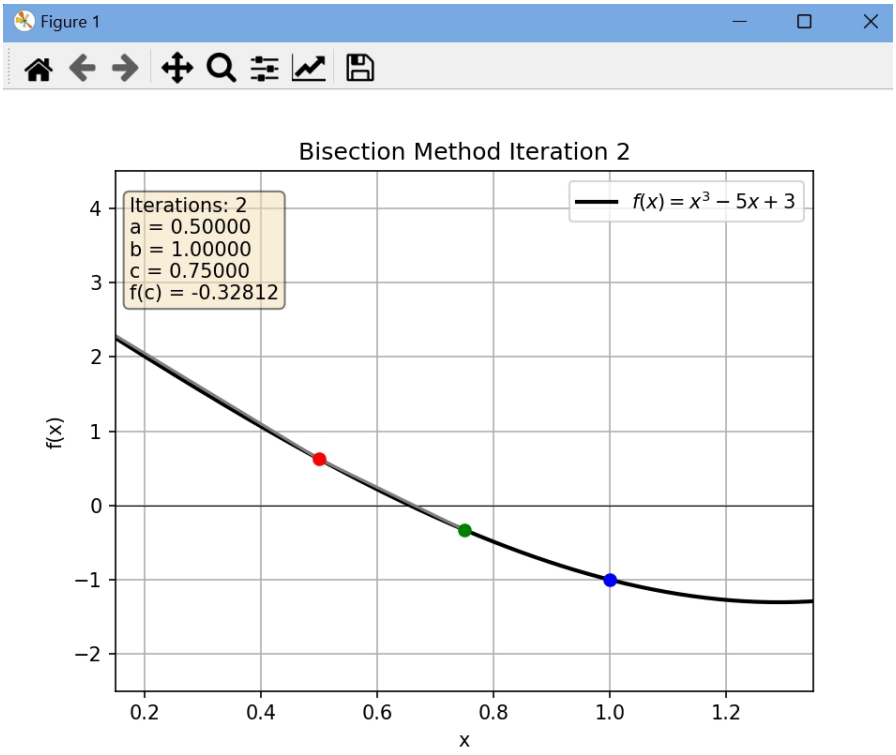

图 3: `bisection_visual.py` 动画示意

```
Select a method (1-7): 6

--- Problem Steps Execution ---

Part (i): Bisection Method to find roots to 4 decimal places
Root in [-3.00, -2.00]: -2.4909
Iterations: 14
Root in [0.00, 1.00]: 0.6567
Iterations: 14
Root in [1.00, 3.00]: 1.8343
Iterations: 15

Part (ii): Newton-Raphson Method to refine roots to 14 decimal places
Refined root starting from -2.4909: -2.49086361536103
Iterations: 3
Refined root starting from 0.6567: 0.65662043104711
Iterations: 3
Refined root starting from 1.8343: 1.83424318431392
Iterations: 3

Part (iii): Hybrid Method to find roots to 14 decimal places
Root in [-3.00, -2.00] (Hybrid): -2.49086361536103
Iterations: 87
Root in [0.00, 1.00] (Hybrid): 0.65662043104711
Iterations: 104
Root in [1.00, 3.00] (Hybrid): 1.83424318431392
Iterations: 110

--- Summary of Problem Steps Results ---

Method: Bisection Method
  Root 1: -2.4909 | Iterations: 14
  Root 2: 0.6567 | Iterations: 14
  Root 3: 1.8343 | Iterations: 15

Method: Hybrid Method
  Root 1: -2.49086361536103 | Iterations: 87
  Root 2: 0.65662043104711 | Iterations: 104
  Root 3: 1.83424318431392 | Iterations: 110

Method: Newton-Raphson Method
  Root 1: -2.49086361536103 | Iterations: 3
  Root 2: 0.65662043104711 | Iterations: 3
  Root 3: 1.83424318431392 | Iterations: 3

Do you want to run the program again? (y/n):
```

图 4: `main.cpp` 模式 6，按题目顺序尝试三种方法

```
PS D:\BaiduSyncdisk\Work\Courses\Junior Fall\CompPhys\repo\Week 2\Codes\Problem 1> ./main.exe

Function Plot:
                            |
                            |                       *
                            |                      *
            **********      |
          **          **    |                     *
         **             ***  |                    *
        *                  **|                   *
       *                    **                  *
      *                     | **               *
                            |   **           **
-----*----------------------+-----****-----***------------
    *                       |         *****
   *                        |
                            |
  *                         |
                            |
 *                          |
                            |
*                           |
                            |
-3      -2       -1        0       1        2        3
x range: [-3, 3]
y range: [-10, 10]


Available methods:
1. Bisection Method
2. Hybrid Method
3. Brent Method
4. Ridder Method
5. Newton-Raphson Method
6. Problem Steps Mode
7. Compare All Methods
Select a method (1-7): 7

--- Comparison of All Methods (Precision: 1e-14) ---

Method                  Root 1 (-3,-2)              Iterations     Root 2 (0,1)                Iterations     Root 3 (1,3)                Iterations
-------------------------------------------------------------------------------------------------------------------------------------------------
Bisection Method        -2.490863615361029          47             0.656620431047109           47             1.834243184313927           48
Newton-Raphson Method   -2.490863615361032          4              0.656620431047110           5              1.834243184313922           6
Hybrid Method           -2.490863615361032          87             0.656620431047110           104            1.834243184313922           110
Brent's Method          -2.490863615361032          43             0.656620431047110           21             1.834243184313922           36
Ridder's Method         -2.490863615361032          3              0.656620431047111           5              1.834243184313922           5

Note: Precision is set to 1e-14, output displays 15 decimal places.
```

图 5: `main.cpp` 模式 7，对比五种方法

2 题目 2：二维函数的极小值搜索

2.1 题目描述

Search for the minimum of the function $g(x, y) = \sin(x + y) + \cos(x + 2y)$ in the whole space.

2.2 程序描述

实现了四种优化算法：最速下降法，共轭梯度下降法，模拟退火，遗传算法

使用`g++ *.cpp -o main`编译，`./main`运行（也有已经编译好的 `main.exe`），按照提示可以选择各类方法或者一起比较，也可以自定义各种算法初始值。

2.3 伪代码

2.4 结果示例

```
PS D:\BaiduSyncdisk\Work\Courses\Junior Fall\CompPhys\repo\Week 2\Codes\Problem 2> .\main.exe

  Optimization Algorithms Menu:
  1. Steepest Descent Method
  2. Conjugate Gradient Method
  3. Simulated Annealing
  4. Genetic Algorithm
  5. Compare All Methods
  Enter your choice (1-5): 5

  Comparing All Methods with Default Parameters...

  Default Parameters:
  Initial x: 0.0000, Initial y: 0.0000
  Steepest Descent alpha: 0.0050, maxIter: 100000, tol: 1.000e-08
  Conjugate Gradient maxIter: 100000, tol: 1.000e-08
  Simulated Annealing T0: 2000.000, Tmin: 1.000e-08, alpha: 0.990, maxIter: 200000
  Genetic Algorithm populationSize: 100, generations: 5000, mutationRate: 0.0200, crossoverRate: 0.8000

  Results:
  Steepest Descent Method:
  Minimum at (-0.00000, -1.57080)
  Minimum value: -2.00000
  Total iterations: 22503
  Execution Time: 9.970e-04 seconds

  Conjugate Gradient Method:
  Minimum at (0.00000, -1.57080)
  Minimum value: -2.00000
  Total iterations: 75
  Execution Time: 0.000e+00 seconds

  Simulated Annealing:
  Minimum at (6.28167, -1.56745)
  Minimum value: -1.99998
  Total iterations: 2590
  Execution Time: 0.000e+00 seconds

  Genetic Algorithm:
  Minimum at (0.00761, -1.57460)
  Minimum value: -1.99999
  Total iterations: 500000
  Execution Time: 6.543e-02 seconds

  Do you want to run the program again? (y/n):
```

图 6: `main.cpp` 模式 5，对比四种方法

3 题目 3：有限深方势阱中的电子能级与波函数

3.1 题目描述

Electron in the finite square-well potential is, ($V_0 = 10\,\mathrm{eV}$, $a = 0.2\,\mathrm{nm}$)

$$V(x) = \begin{cases} V_0 & x \le -a \quad \text{Region I} \\ 0 & -a < x < a \quad \text{Region II} \\ V_0 & x \ge a \quad \text{Region III} \end{cases}$$

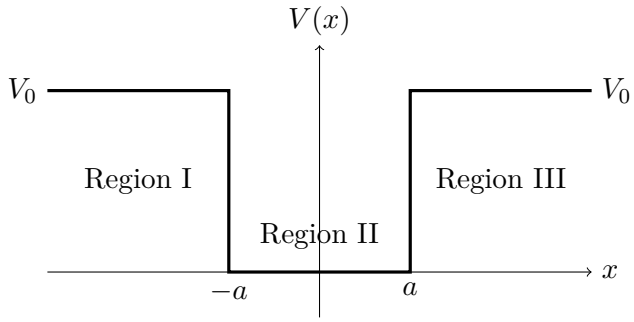

图 7: Finite square-well potential and its corresponding mathematical expression.

Find the three lowest eigen states (**both** energies and wavefunctions).

3.2 程序描述

两种算法，一种针对本题的解析求解，即超越方程图解法：`./Codes/Problem 3/potential.py`，另一种是数值求解：`./Codes/Problem 3/schrodinger.py`，差分表示哈密顿矩阵，适用于更广泛的势能，偷懒都用 Python 实现了。

3.3 伪代码

3.4 结果示例

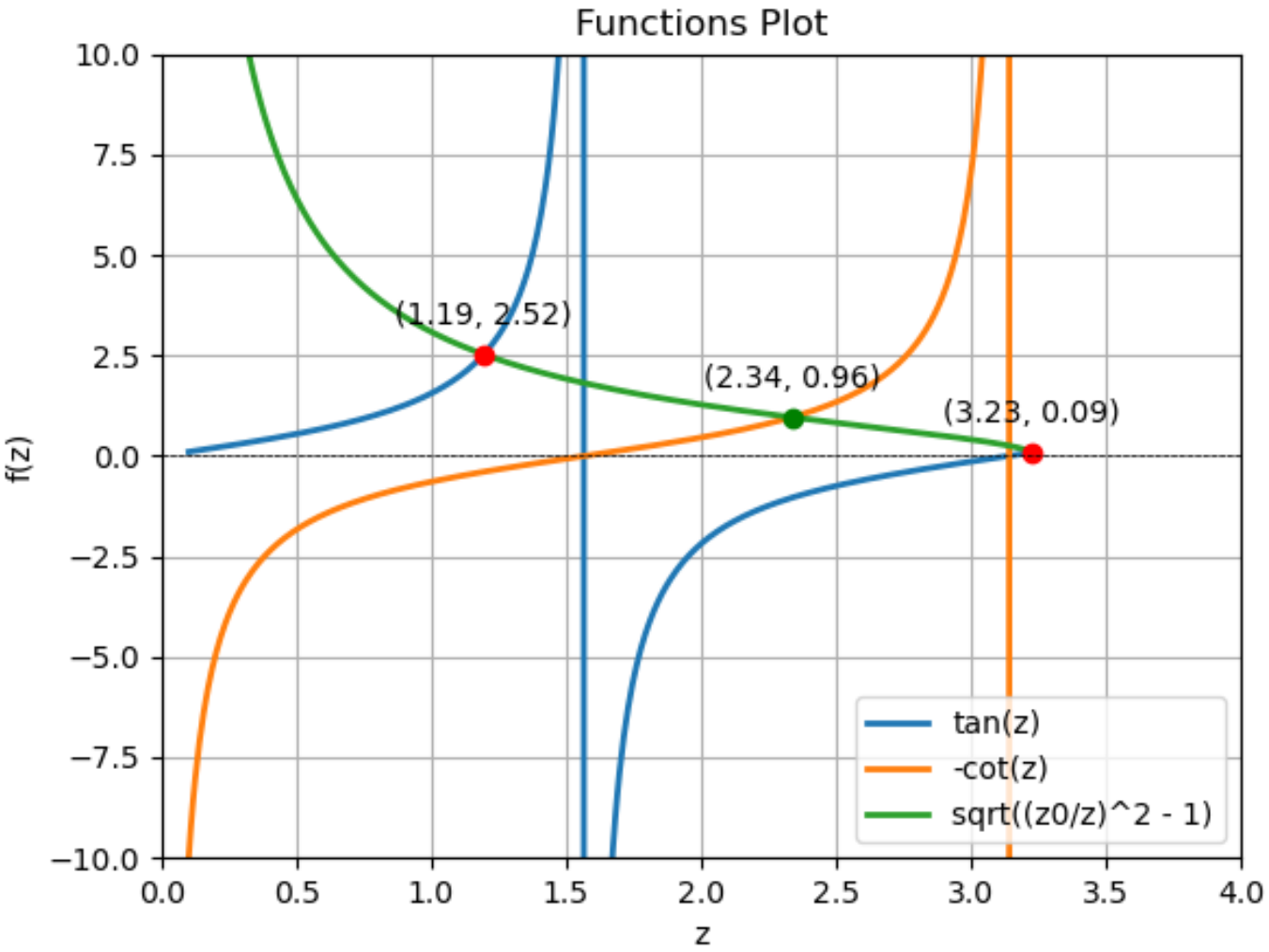

图 8: `potential.py` 解析图解法

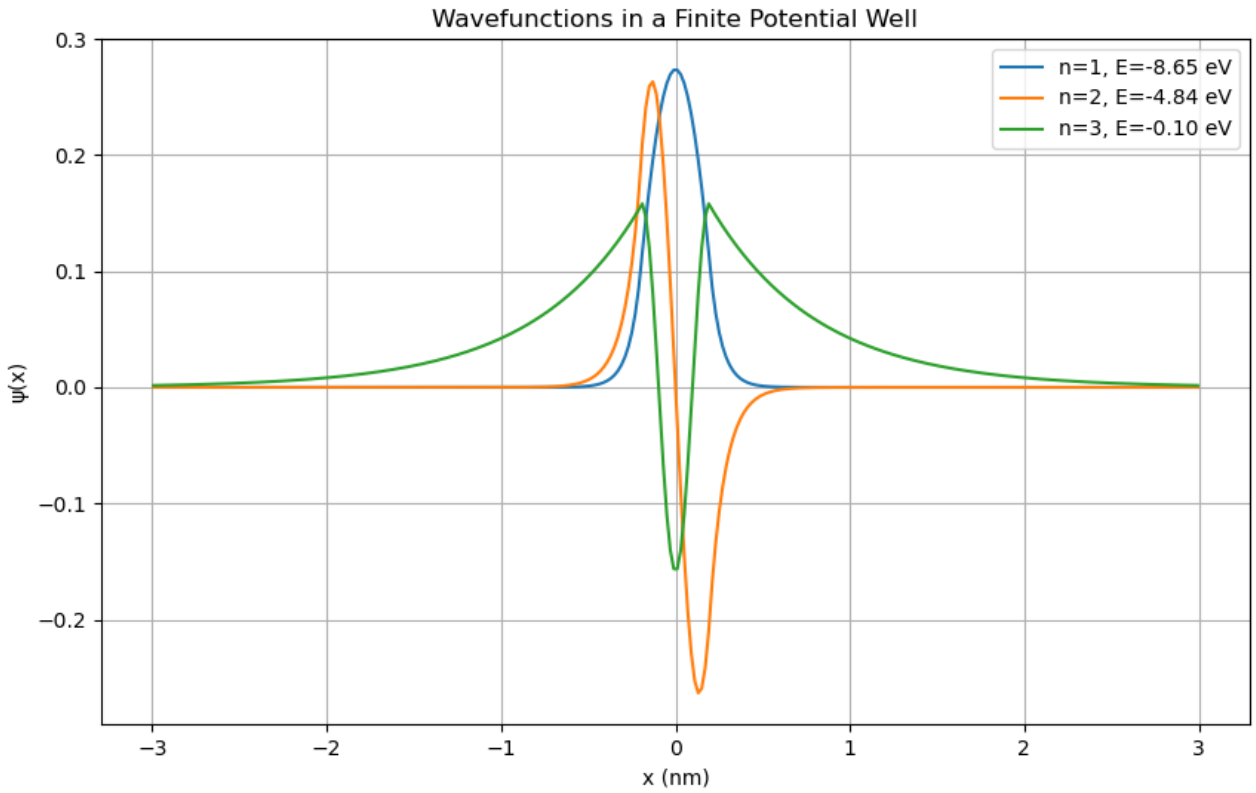

图 9: `schrodinger.py` 数值差分法

今天原本应该是鏖战 *Griffths* 的猫与蔡先生的日子，可惜平板没电。