计算物理第二次作业

李柏轩 20300200004

September 25, 2022

1 题目 1：三次方程求根

1.1 题目描述

Sketch the function $x^3-5x+3=0$.

(1) Determine the two positive roots to 4 decimal places using the bisection method.

(2) 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.

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

1.2 程序描述

题目要求求三次方程的两个正根，即求 $f(x)=x^3-5x+3$ 的两个正零点。题目同时规定了求解的方法：二分法、Newton-Raphson 方法以及混合法。于是本程序分别编写了这三种方法的函数并调用求解。首先画出 $f(x)$ 的 Fig. 1可知，其两个正零点分别处于 $(0,1)$ 和 $(1,2)$ 中，其中两区间端点函数值均异号。以这两个区间以及目标精度为参数分别调用二分法求解两个正根，再用求得的两个结果作为初始值，与目标精度一起代入 Newton-Raphson 法函数求得两个正根。混合法则单独调用其函数求解。

本程序源文件为 Root.py，运行依赖 Python 第三方库 Numpy 和 Matplotlib。在终端进入当前目录，使用命令 python -u Root.py 运行本程序。或点击 Root.exe 运行。运行后先输出 Fig. 1所示图像，关闭图像后在终端输出三种方法分别求得的两个正根。

1.3 伪代码

二分法的伪代码如 Alg.1所示。Newton-Raphson 法的伪代码如 Alg.2所示。混合法的伪代码如 Alg. 3所示。

1.4 输入输出示例

程序运行结果如 Fig. 2与 Table 1。可以看到二分法结果精确到四位小数，Newton-Raphson 和混合法精确到 14 位小数，三种方法结果一致。

Algorithm 1 Bisection method for finding roots

Input: The lower side x_l, the upper side x_r, and the permitted error ϵ.

Output: The root of $f(x) = 0$

$x \leftarrow \frac{x_l+x_r}{2}$
repeat
  if $\text{sgn}(f(x)) = \text{sgn}(f(x_l))$ **then**
    $x_l \leftarrow x$
  else if $\text{sgn}(f(x)) = \text{sgn}(f(x_r))$ **then**
    $x_r \leftarrow x$
  else
    return x
  end if
  $x \leftarrow \frac{x_l+x_r}{2}$
until $|x_l - x_r| \leq \epsilon$
return x

Algorithm 2 Newton-Raphson method for finding roots

Input: The initial guess of the root x and the permitted error ϵ

Output: The root of $f(x) = 0$

$x \leftarrow \frac{x_l+x_r}{2}$
repeat
  $x \leftarrow x - \frac{f(x)}{f'(x)}$
until $|\frac{f(x)}{f'(x)}| \leq \epsilon$
return x

Table 1: 题目 1 输出实例

Method	x_1	x_2	Accuracy
Bisection	0.6566	1.8342	4 decimal places
Newton-Raphson	0.65662043104711	1.834243184331392	14 decimal places
Hybrid	0.65662043104711	1.834243184331392	14 decimal places

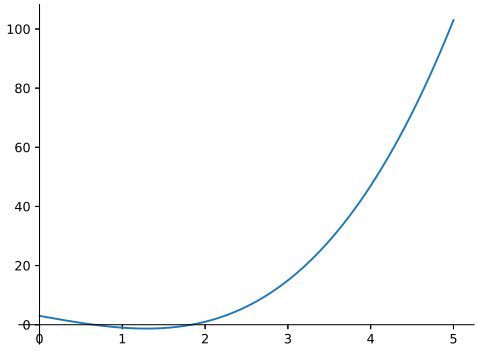

Figure 1: $f(x) = x^3 - 5x + 3$ 的函数图像

```
The result of bisection are:    x1= 0.6566 ,x2= 1.8342
The result of NR method are:    x1= 0.65662043104711 ,x2= 1.83424318431392
The result of hybrid method are:        x1= 0.65662043104711 ,x2= 1.83424318431392
```

Figure 2: 题目 1 运行结果

2 题目 2：求函数极值

2.1 题目描述

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

2.2 程序描述

题目要求函数 $f(x, y) = \sin(x + y) + \cos(x + 2y)$ 在全空间的极小值点。由 Fig. 3可以看出，$f(x, y)$ 在 x, y 方向的周期均为 2π，于是只要我们求出其在一个周期内的极小值点 (x_0, y_0)，则其在全空间的极小值点为 $(x_0 + 2k_1\pi, y_0 + 2k_2\pi), k_1, k_2 \in N$。本程序利用梯度下降法，从一初始点出发，根据函数梯度更新点坐标不断向函数值更低处运动，直到两次迭代间差距小于允许值。

本程序源文件为 Extreme.py，运行依赖 Python 第三方库 Numpy 和 Matplotlib。在终端进入当前目录，使用命令 python -u Extreme.py 运行本程序。或点击 Extreme.exe 运行。运行后先输出 Fig. 3所示图像，关闭图像后在终端输出与初始值有关的某个极小值点的坐标。

2.3 伪代码

梯度下降法的伪代码如 Alg. 4

2.4 输入输出示例

程序运行结果如 Fig. 4与 Table 1。可以看到当初始值在某方向变化 2π 时，得到的极小值的坐标也在同方向变化 2π。最终得到 $f(x, y)$ 在全空间的极小值为 $(2k_1\pi, -\frac{\pi}{2} + 2k_2\pi), k_1, k_2 \in N$。

Algorithm 3 Hybrid method for finding roots

Input: The lower side x_l, the upper side x_r, and the permitted error ϵ.

Output: The root of $f(x) = 0$

$x_0 \leftarrow \frac{x_l + x_r}{2}$
repeat
  if $f'(x) = 0$ **then**
    Update x_l, x_r, x_0 in the bisection way
  else
    $x_0 \leftarrow x_0 - \frac{f(x_0)}{f'(x_0)}$
    if $x \in (x_l, x_r)$ **then**
      Update x_l, x_r in the bisection way
    else
      $x_0 \leftarrow \frac{x_l + x_r}{2}$
      Update x_l, x_r, x_0 in the bisection way
    end if
  end if
  $x \leftarrow \frac{x_l + x_r}{2}$
until $|x_l - x_r| \leq \epsilon$
return x

Algorithm 4 Gradient Descent method for finding extremes

Input: The initial guess (x_0, y_0), the step of iteration, and the permitted error ϵ

Output: One local minima of $f(x, y)$

repeat
  $(x_0, y_0) \leftarrow (x_0, y_0) - step \cdot \frac{\nabla f}{|\nabla f|}$
until $|\nabla f| \leq \epsilon$
return (x_0, y_0)

Table 2: 题目 2 输出实例

Local minima / Initial x / Initial y	-6.2832	0	6.2832
-6.2832	(-6.2832,-7.854)	(0,-7.854)	(6.2832,-7.854)
0	(-6.2832,-1.5708)	(0,-1.5708)	(6.2832,-1.5708)
6.2832	(-6.2832,4.7124)	(0,4.7124)	(6.2832,4.7124)

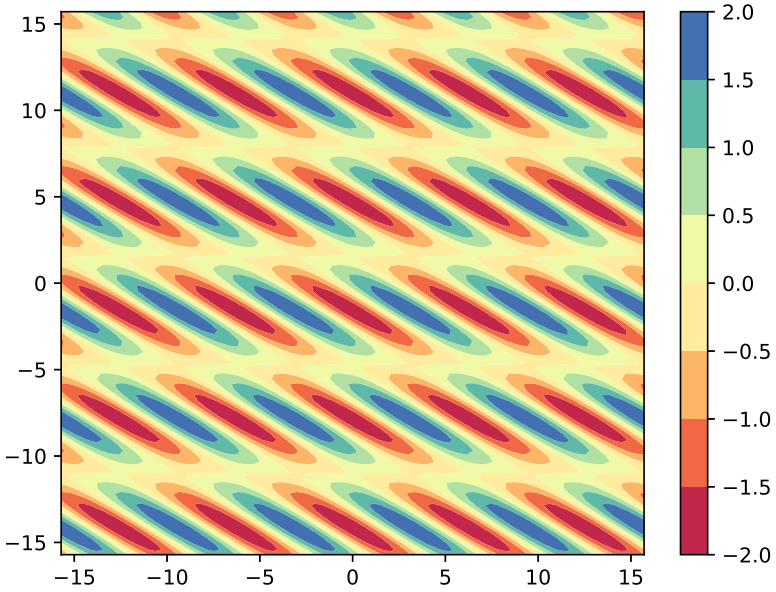

Figure 3: $f(x,y) = \sin(x+y) + \cos(x+2y)$ 的函数图像

```
For initial guess( -6.2832 -6.2832 ) , One local minima is: ( -6.2832 -7.854 )
For initial guess( -6.2832 0.0 ) , One local minima is: ( -6.2832 -1.5708 )
For initial guess( -6.2832 6.2832 ) , One local minima is: ( -6.2832 4.7124 )
For initial guess( 0.0 -6.2832 ) , One local minima is: ( -0.0 -7.854 )
For initial guess( 0.0 0.0 ) , One local minima is: ( -0.0 -1.5708 )
For initial guess( 0.0 6.2832 ) , One local minima is: ( -0.0 4.7124 )
For initial guess( 6.2832 -6.2832 ) , One local minima is: ( 6.2832 -7.854 )
For initial guess( 6.2832 0.0 ) , One local minima is: ( 6.2832 -1.5708 )
For initial guess( 6.2832 6.2832 ) , One local minima is: ( 6.2832 4.7124 )
```

Figure 4: 题目 2 运行结果

3 题目 3：磁化强度的温度依赖

3.1 题目描述

Determine M(T) the magnetization as a function of temperature T for simple magnetic materials.

$$m(t) = \tanh(\frac{m(t)}{t}) \tag{1}$$

in which $m(T) = \frac{M(T)}{\mu N}, t = \frac{T}{T_c}, T_c = \frac{N\mu^2\lambda}{k_B}$. For a given t,solve m, and plot m as a function of t.

3.2 程序描述

题目要求求解由 Eq. 1支配的约化磁化强度和约化温度的关系，也即在每个给定的 t 下求解 $f(m;t) = \tanh(\frac{m}{t}) - m = 0$ 的根。由体系的自发对称性破缺得该方程存在唯一零解或存在零解与正负对称的另外两个解。对 $t = 0.1, 0.7, 0.9$ 绘制 $f(m;t)$ 图像如 Fig. 5所示。可以看到当 t 由 1 向

0 变化时，m 由 0 向 1 移动；当 $t > 1$ 时 $f(m;t) = 0$ 无正根。于是对于每个 t，采用问题 1 中的 Newton-Raphson 法求根，最终得到 $m - t$ 图像。

本程序源文件为 Megnet.py，运行依赖 Python 第三方库 Numpy 和 Matplotlib。在终端进入当前目录，使用命令 python -u Megnet.py 运行本程序。

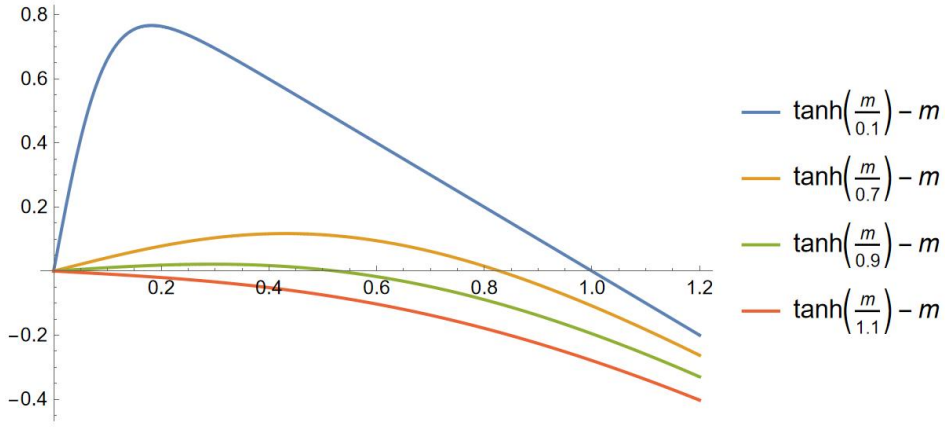

Figure 5: $f(m;t) = \tanh(\frac{m}{t}) - m = 0$ 的函数图像

3.3 伪代码

Algorithm 5 Solve the numerical dependence of m on t

Input: The lower and upper bound of t and the permitted error ϵ.

Output: The numerical dependence of m on t.

1: **for** $i \leftarrow t_l$ to t_h **do**

2: $m[i]$ =Newton$(1, i, \epsilon)$//Newton(m_{ini}, t, ϵ) is the Newton-Raphson method in Alg. 2

3: **end for**

3.4 输入输出示例

运行结果如 Fig. 6。与理论预期相符。

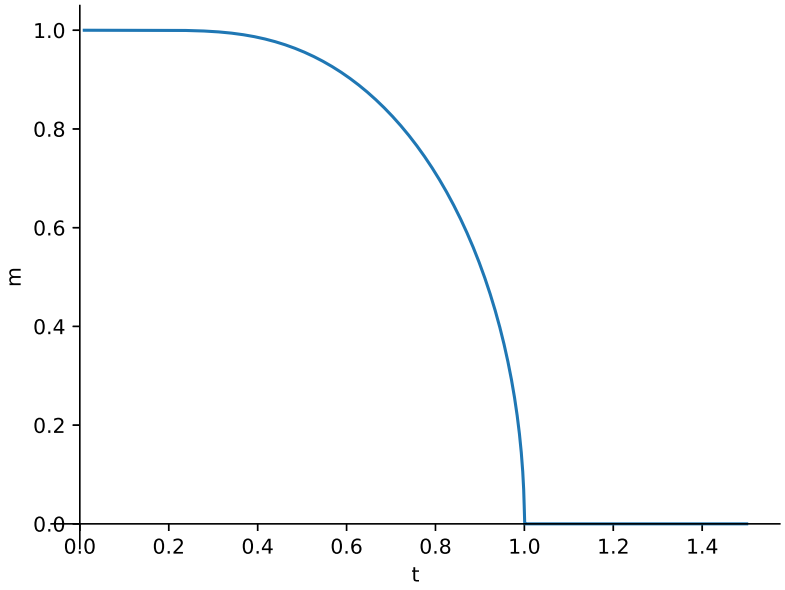

Figure 6: m 对 t 的依赖关系