

计算物理作业 2

谢昀城 22307110070

2024 年 9 月 18 日

1 题目 1：求方程的根

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. Note: You first need to bracket each of the roots.
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.

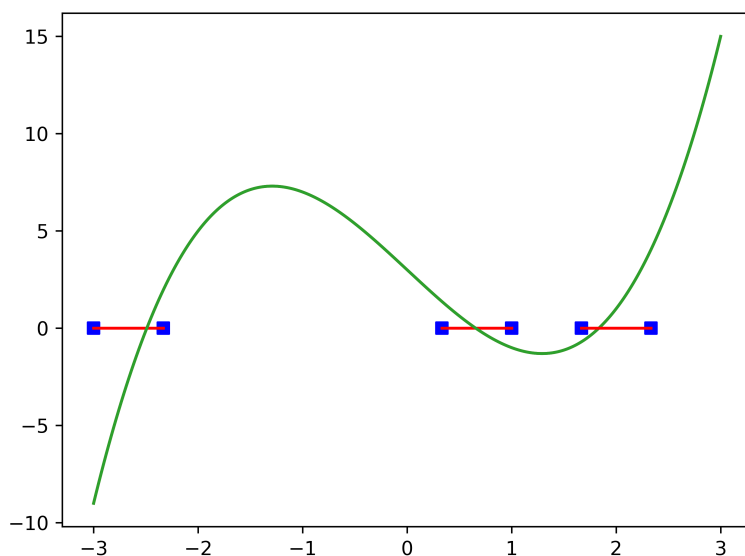


图 1: $f(x)$ - x 图像。其中标出了由 *find_bracket* 函数找出的变号区域

1.2 程序描述

对于题目要求求解的方程，我们首先通过在一定范围内均匀采样（一般采样 10 个点）找到其变号区间，在程序中由 `find_bracket` 完成（见图1）。对于 1.1 题，我们使用 `bisection_method` 函数接收其变号区间，并在区间内使用二分法逼近其中的根，直到误差在容差 10^{-4} 以下，这里根的误差使用二分区间大小 $|a - b|$ 估计。对于 1.2 和 1.3 题，由于 `np.float64` 类型浮点数的精度大约只有 16 位有效位数左右，因此我们使用 `decimal` 库实现 50 位有效数字的计算。我们在 `newton_raphson_method` 和 `hybrid_method` 函数中接收由 1.1 中得到的解，将其误差缩小至 10^{-14} 以下，这里误差我们使用 $|f(x)/f'(x)|$ 估计。

本程序源文件为 `findroot.py`，在终端进入当前目录，使用命令 `python -u findroot.py` 运行本程序。运行时请保证 Python 第三方库 Numpy, Matplotlib, decimal 已安装。程序开发环境为 Python3.12.3，可在 Python3.8 以上版本中运行。

1.3 伪代码

1.3.1 bisection method 伪代码:

Algorithm 1 FindBracket

function FINDBRACKET($f, rg, n, ifplot$)

INPUT: f (function), rg (range), n (number of points), $ifplot$ (boolean)

OUTPUT: $bracket$ (list of tuples)

$x \leftarrow \text{linspace}(rg[0], rg[1], n)$

$y \leftarrow f(x)$

$yRoll \leftarrow \text{roll}(y, 1)$

$yCov \leftarrow y[1:] * yRoll[1:]$

$cr0 \leftarrow \text{where}(yCov < 0)$

▷ Convolve 2 series to find the cross point

$bracket \leftarrow [(x[i], x[i + 1])] \text{ for } i \text{ in } cr0[0]$

return $bracket$

end function

1.3.2 bisection method 伪代码:

Algorithm 2 BisectionMethod

function BISECTIONMETHOD($f, rg, tol, maxIter$)

INPUT: f (function), rg (range), tol (tolerance), $maxIter$ (maximum iterations)

OUTPUT: ($root, error$) (root of the function and the error)

$a, b \leftarrow rg$

if $f(a) * f(b) \geq 0$ **then**

Raise ValueError("Function does not change sign in the interval.")

end if

for $i \leftarrow 0$ **To** $maxIter - 1$ **do**

$c \leftarrow (a + b)/2$

if $|b - a| < tol$ **then**

▷ Convergence condition, using $|a - b|$ as the error

print the root and Break

else if $f(c) * f(a) < 0$ **then**

```

         $b \leftarrow c$ 
    else
         $a \leftarrow c$ 
    end if
end for
return  $(a + b)/2, |b - a|$ 
end function

```

1.3.3 Newton-Raphson method 伪代码:

Algorithm 3 NewtonRaphsonMethod

```

function NEWTONRAPHSONMETHOD( $fDecimal, dfDecimal, x0, tol, maxIter$ )
    INPUT:  $fDecimal$  (function),  $dfDecimal$  (function),  $x0$  (initial point),  $tol$  (tolerance),  $maxIter$  (maximum
iteration)
    OUTPUT:  $x$  (root),  $error$  (absolute error)
     $x \leftarrow \text{Decimal}(x0)$  ▷ Convert  $x0$  to decimal type
     $tol \leftarrow \text{Decimal}(tol)$  ▷ Convert  $tol$  to decimal type
    for  $i \leftarrow 1$  To  $maxIter$  do
         $fx \leftarrow fDecimal(x)$ 
         $dfx \leftarrow dfDecimal(x)$ 
        if  $dfx = 0$  then
            Raise ValueError("Meet zero derivative at",  $x$ )
        end if
         $dx \leftarrow fx/dfx$ 
        if  $|dx| < tol$  then
            print the root and Break
        end if
         $x \leftarrow x - dx$ 
    end for
    return  $x, |dx|$ 
end function

```

1.3.4 Hybrid method 伪代码:

Algorithm 4 HybridMethod

```

function HYBRIDMETHOD( $f, df, rg, tol, maxIter$ )
    INPUT:  $f$  (function),  $df$  (derivative of  $f$ ),  $rg$  (range),  $tol$  (tolerance),  $maxIter$  (maximum iteration)
    OUTPUT:  $x$  (root),  $error$  (absolute error)
     $a, b \leftarrow \text{Decimal}(rg)$  ▷ Convert range to decimal type
     $eps \leftarrow 10^5\text{-environment precision}$ 
     $x \leftarrow (a + b)/2$ 
    if  $f(a) * f(b) \geq 0$  then
        Raise ValueError("Function does not change sign in the interval.")
    end if

```

```

end if
for  $i \leftarrow 1$  To  $maxIter$  do
     $dfx \leftarrow df(x)$ 
     $fx \leftarrow f(x)$ 
    if  $|dfx| < eps$  then
         $x \leftarrow (a + b)/2$ 
         $dx \leftarrow |b - a|$ 
    else
         $dx \leftarrow fx/dfx$ 
         $x \leftarrow x - dx$ 
    end if
    if  $|dx| < tol$  then
        print the root and Break
    end if
end for
return  $x, |dx|$ 
end function

```

▷ If derivative is too small, use bisection method

1.4 输入输出实例

对于本程序，运行后会生成图1为“f(x) with bracket.png”于当前目录下，并输出使用不同方法求得的根。表 1 为不同方法求得的根及其误差，图2为程序运行截图，可以看到 *NewtonRaphson* 方法和混合方法均可以在很少的迭代次数内将根逼近到很高的精度。

Method	Root1	Error1	Root11	Root2
Bisection	0.656619	5.1e-06	1.834241	5.1e-06
Newton-Raphson	0.656620431047110366142231	1.2e-24	1.83424318431392171711564	1.8e-23
Hybrid	0.656620431047110366	1.4e-18	1.83424318431392171711562613	1.3e-26

表 1: 问题 1 的结果实例

2 题目 2：求函数极小值

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

在本程序中，我们使用梯度下降方法来搜索 $g(x,y)$ 的极小值点，其在函数 *gradient_descent* 中实现，并且，为了防止在某些位置梯度消失而导致停留在函数的鞍点上，每次下降会加上一个高斯噪声。

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\homework2> python -u .\findroot.py

#####

The bisection method converged after 17 iterations
The root is 0.656619 The error is 5.1e-06
The bisection method converged after 17 iterations
The root is 1.834241 The error is 5.1e-06

#####

The Newton-Raphson method converged after 2 iterations
The root is 0.656620431047110366142231
The error is 1.2e-24
The Newton-Raphson method converged after 2 iterations
The root is 1.83424318431392171711564
The error is 1.8e-23

#####

The hybrid method converged after 3 iterations
The root is 0.656620431047110366
The error is 1.4e-18
The hybrid method converged after 5 iterations
The root is 1.83424318431392171711562613
The error is 1.3e-26
请按任意键继续...
```

图 2: 题目 1 运行结果

由于 $\sin(x)$ 和 $\cos(x)$ 的周期性, $g(x, y)$ 实际上有无数多个极值点, 只需满足 $x_1 + y_1 = 2m\pi, x_2 + 2y_2 = 2n\pi, (m, n) \in \mathbb{Z}$ 。并且, 由于:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2, -1 \\ -1, 1 \end{pmatrix} \begin{pmatrix} x + y \\ x + 2y \end{pmatrix}$$

故只需要在图3(a) 红线所围成的元胞中找到一个极小值点 x_0, y_0 即可, 其余极小值点由 $x_m = x_0 + 2m\pi, y_n = y_0 + 2n\pi, (m, n) \in \mathbb{Z}$ 给出。

本程序源文件为 findmin.py, 在终端进入当前目录, 使用命令 `python -u findmin.py` 运行本程序。运行时请保证 Python 第三方库 Numpy, Matplotlib 已安装。程序开发环境为 Python3.12.3, 可在 Python3.8 以上版本中运行。

2.3 伪代码

梯度下降法伪代码:

Algorithm 5 GradientDescent

function GRADIANTDESCENT($X_0, g, dg, ap, tol, maxIter$)

INPUT: X_0 (initial guess), g (function), dg (gradient of g), ap (learning rate), tol (tolerance), $maxIter$ (maximum iterations)

OUTPUT: X (final point), his (history of points)

$X \leftarrow \text{array}(X_0)$

▷ Initialize X with X_0 as a float array

$his \leftarrow []$

▷ Initialize empty history list

for $i \leftarrow 1$ **To** $maxIter$ **do**

$dX \leftarrow \nabla dg(X)$

▷ Calculate the gradient at current X

$X \leftarrow X - ap \times dX + \text{randomNormal}(0, tol)$

▷ Update X with learning rate and noise

$his \leftarrow \text{append}(his, \text{copy}(X))$

▷ Store the current X in history

if $\|dX\| < tol$ **then**

Break

```

end if
end for
return X, his
end function

```

2.4 输入输出实例

对于本程序，运行后会生成图3为“find g(x,y) min.png”于当前目录下，并输出使用梯度下降法找到的极小值点。图4为程序运行截图，得到极小值为-2.00000，极小值点为

$$X_{min} = 0.00000 + 2m\pi, Y_{min} = -1.57079 + 2n\pi, (m, n) \in \mathbb{Z}$$

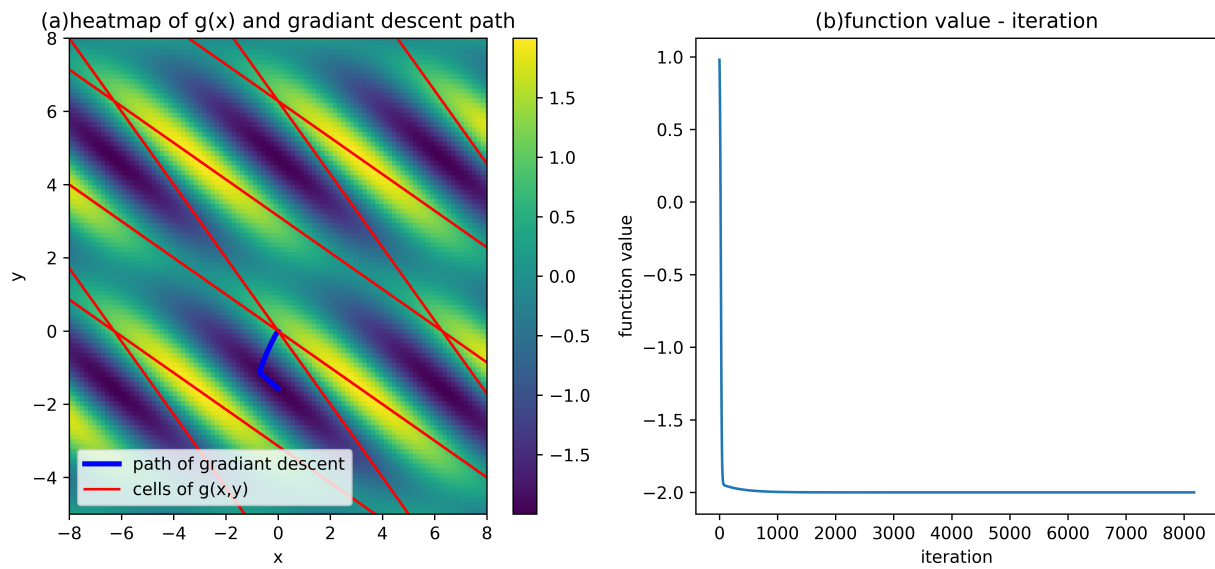


图 3: 题目 2 (a) $g(x,y)$ 的热力图，用红线标注了其单元，蓝线为梯度下降路径。(b) 函数值随迭代次数的变化

```

(base) PS C:\Users\ASUS\Desktop\计算物理基础\homework2> python -u .\findmin.py
gradient_descent converged after 7514 iterations
minimum of g(x) is -2.00000
Xmin is -0.00000+ 2 m pi,Ymin is -1.57079+2 n pi, m,n are any integers

```

图 4: 题目 2 程序运行截图

3 题目 3

3.1 题目描述

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

$$V(x) = \begin{cases} V_0, & \text{if } x \leq -a, \text{Region I} \\ 0, & \text{if } -a < x < a, \text{Region II} \\ V_0, & \text{if } x \geq a, \text{Region III} \end{cases}$$

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

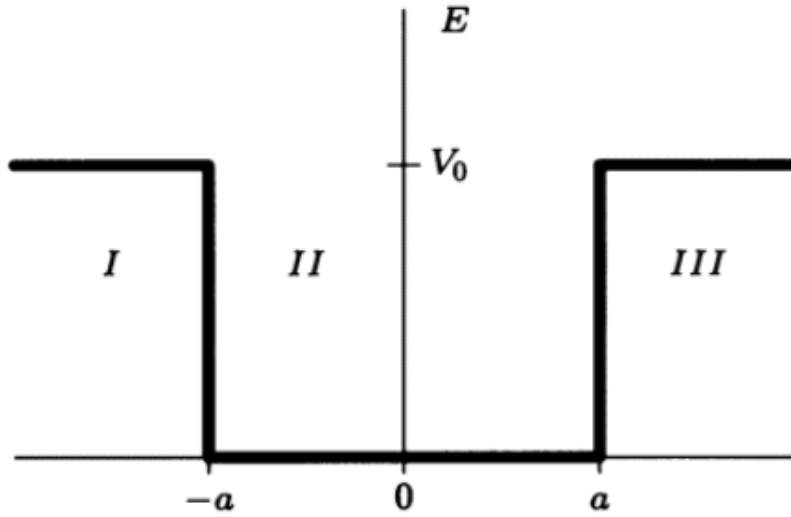


图 5: 题目 3 方势阱示意图

3.2 程序描述

在本程序中, 我们通过求解奇、偶宇称的波函数的边界条件方程: $f_{\text{odd}} = \alpha \sin \alpha a + \beta \cos \alpha a = 0$ 和 $f_{\text{even}} = \alpha \sin \alpha a + \beta \cos \alpha a = 0$, 其中 $\alpha = \sqrt{2mE}/\hbar$, $\beta = \sqrt{2m(V_0 - E)}/\hbar$, 得到波函数的三个能量最小的束缚态 (由图6可看出事实上只存在 3 个束缚态)。

对 f_{even} 和 f_{odd} 的求根使用在题目 1 中实现的 `bisection_method` 实现, 并重新计算出 A,B,C,D 系数得到波函数。接着对波函数的平方进行积分, 将波函数归一化。积分在 `intTrapezoidal` 中使用梯形法实现, 即 $N_e = h(1/2\rho_0 + \rho_{01} + \dots + \rho_{N-1} + 1/2\rho_N)$, 其中 $h = \frac{b-a}{N}$, a, b 为上下界, N 为采样点数。由于 $|x| > a$ 波函数为指数下降, 因此我们对波函数在 $|x| > 10a$ 处截断。

为了降低数值误差, 本程序中能量单位采用 eV , 长度单位采用 nm , 并定义常数 $k = \frac{2m_e}{\hbar^2} = 26.24684351\text{nm}^{-1} \cdot \text{eV}^{-1}$

本程序源文件为 `solvesquarepotential.py`, 在终端进入当前目录, 使用命令 `python -u solvesquarepotential.py` 运行本程序。运行时请保证此程序与题目 1 程序 `findroot.py` 在同一文件夹下, 且 Python 第三方库 Numpy, Matplotlib 已安装。程序开发环境为 Python3.12.3, 可在 Python3.8 以上版本中运行。

3.3 伪代码

3.3.1 Trapezoidal integrate 伪代码:

Algorithm 6 IntTrapezoidal

function INTTRAPEZOIDAL(f, a, b, n)

INPUT: f (function), a (lower limit), b (upper limit), n (number of points)

OUTPUT: $intfx$ (integration result)

$x \leftarrow \text{linspace}(a, b, n + 1)$

▷ Generate $n + 1$ points between a and b

$intfx \leftarrow \left(\sum_{i=0}^n f(x[i]) - \frac{1}{2}f(x[0]) - \frac{1}{2}f(x[n]) \right) \cdot \frac{b-a}{n}$

return $intfx$

end function

Algorithm 7 Calculate the wave function

$Ei, erri \leftarrow \text{BisectionMethod}(f_{Even/odd}, bracket_{Even/odd}[0])$

$F \leftarrow 1$

if Even Wave Fuction **then**

$A \leftarrow 0, C \leftarrow F, B \leftarrow Fe^{-\beta a} / \cos(\alpha a)$

else if Odd Wave Function **then**

$B \leftarrow F, C \leftarrow -F$

$A \leftarrow Fe^{-\beta a} / \sin(\alpha a)$

▷ Calculate A,B,C,F

end if

$norm \leftarrow \text{IntTrapezoidal}(f, -10a, 10a, 1000)$ $A, B, C, F \leftarrow (A, B, C, F) / norm$

▷ Normalize the wave function

Plot and Output the wave function

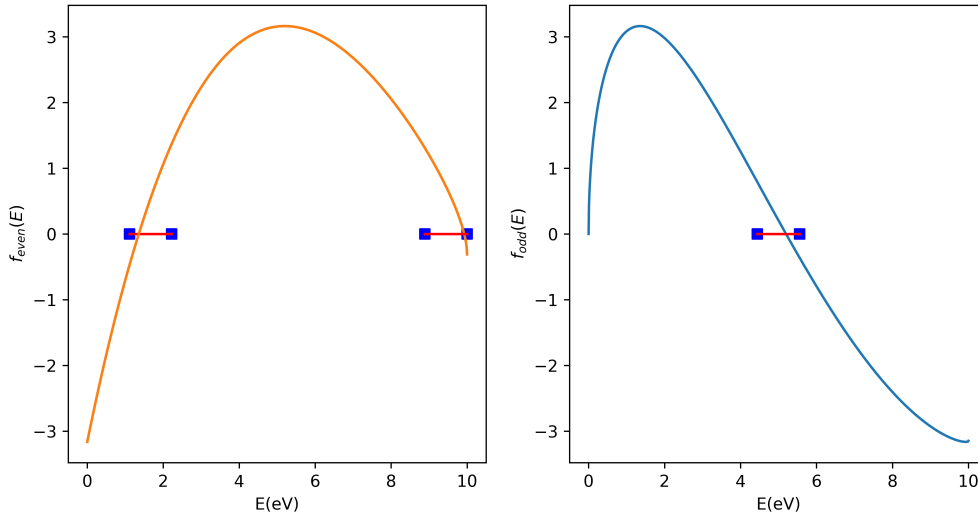


图 6: f_{odd} 和 f_{even} 根位置示意图

3.4 输入输出实例

对于本程序，运行后会生成图6为”f_even and f_odd.png” 和7为”wavefunction.png” 于当前目录下，并输出三个束缚态的能量和波函数表达式。图8为程序运行截图。程序得到束缚态能量为 $E_0 = 1.35689\text{eV}$, $E_1 = 5.19897\text{eV}$, $E_2 = 9.92541\text{eV}$ ，波函数表达式分别为 (x 单位为 nm)

$$\psi_0(x) = \begin{cases} 14.51282 e^{-15.06168 x}, & x > a \\ 14.51282 e^{15.06168 x}, & x < -a \\ 1.93748 \cos(5.96776 x), & -a \leq x \leq a \end{cases}$$

$$\psi_1(x) = \begin{cases} 12.66141 e^{-11.22550 x}, & x > a \\ -12.66141 e^{11.22550 x}, & x < -a \\ 1.85990 \sin(11.68146 x), & -a \leq x \leq a \end{cases}$$

$$\psi_2(x) = \begin{cases} 1.38157 e^{-1.39923 x}, & x > a \\ 1.38157 e^{1.39923 x}, & x < -a \\ -1.04824 \cos(16.14034 x), & -a \leq x \leq a \end{cases}$$

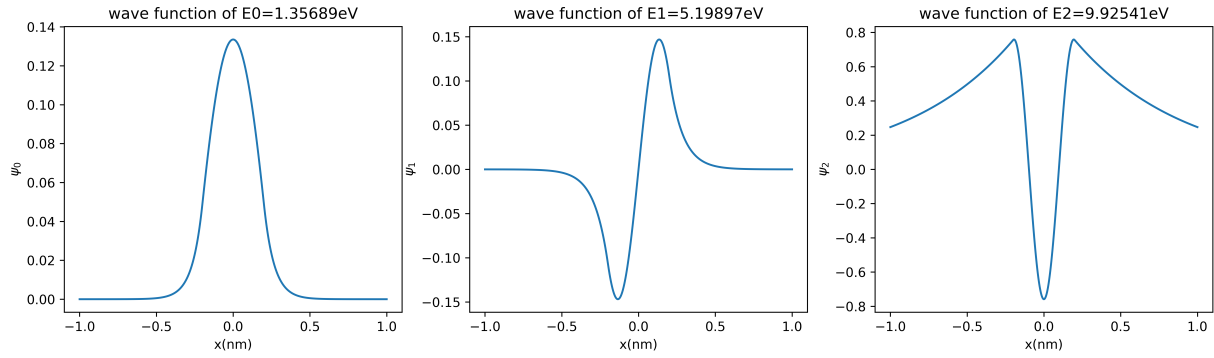


图 7: 不同能量的波函数图像

```
(base) PS C:\Users\ASUS\Desktop\计算物理基础\hw2\homework2> python -u .\solvesquarepotential.py
The bisection method converged after 27 iterations
The root is 1.356892449 The error is 8.3e-09
The bisection method converged after 27 iterations
The root is 5.198969555 The error is 8.3e-09
The bisection method converged after 27 iterations
The root is 9.925406256 The error is 8.3e-09

wave function of E0=1.35689eV:
14.51282 Exp(-15.06168*x) for x>a
14.51282 Exp(15.06168*x) for x<-a
1.93748 cos(5.96776*x) for -a<x<a

wave function of E1=5.19897eV:
12.66141 Exp(-11.22550*x) for x>a
-12.66141 Exp(11.22550*x) for x<-a
1.85990 sin(11.68146*x) for -a<x<a

wave function of E2=9.92541eV:
1.38157 Exp(-1.39923*x) for x>a
1.38157 Exp(1.39923*x) for x<-a
-1.04824 cos(16.14034*x) for -a<x<a
```

图 8: 题目 3 程序运行截图