

# Homework2

Zric 23307110123

January 8, 2026

## 1 Problem1: Find roots

### 1.1 Problem description

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

(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.

### 1.2 Algorithm description

(1) For the first question, we first need to bracket each of the roots by taking sample points  $x_1, x_2, \dots$  of  $f(x)$ , comparing the signs between  $f(x_i)$  to identify the possible region of each root. For an order-3 equation, there is at most only 3 real roots. And we have

$$f(-3) = -9 \quad f(0) = 3 \quad f(1) = -1 \quad f(2) = 1 \quad (1)$$

Using continuity of function  $f(x) = x^3 - 5x + 3$ , we can identify that the 3 roots lie in  $[-3, 0]$ ,  $[0, 1]$ ,  $[1, 2]$  separately. To find the 2 positive roots, we will first search separately in interval  $[0, 1]$  and  $[1, 2]$  using bisection method, the break condition of the iteration is that  $|high - low| < 1 \times 10^{-4}$ ,  $high$  denotes the upper bound in interval  $[low, high]$ ,  $low$  the lower bound.

(2) For the second question, we will use the answer from (1) as the start point  $x_0$  of Newton-Raphson method, and iteration will not stop until  $|x_{k+1} - x_k| < 1 \times 10^{-14}$ , the relation between  $x_k, x_{k+1}$  can be written as:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (2)$$

(3) For the third question, we use the hybrid method, combining both N-R method and bisection method in searching roots, this method avoids some intrinsic problems of N-R method so it's more robust, for example, N-R method is invalid at  $f'(x) = 0$ , but here in hybrid method this problem can be avoided by switching on bisection method, the algorithm flowchart is shown below:

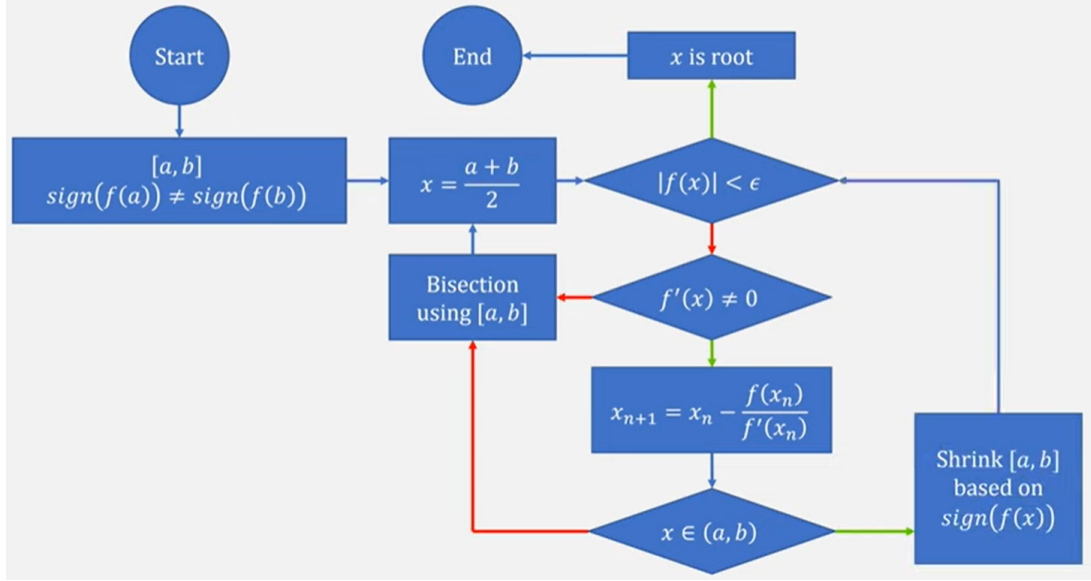


Figure 1: algorithm flowchart of hybrid method

### 1.3 Output

run code `problem1.py` in terminal:

```

Bisection [0,1]: root=0.6566, iters=14
Bisection [1,2]: root=1.8343, iters=14
Newton Raphson root1: 0.65662043104711, iters=2
Newton Raphson root2: 1.83424318431392, iters=2
Hybrid [0,1]: 0.65662043580238, iters=71
Hybrid [1,2]: 1.83424318431392, iters=56
  
```

Figure 2: output of 3 methods

Here iters denotes the number of iteration steps,  $[0,1],[1,2]$  means the 2 intervals of 2 positive roots. As can be seen from above, the strategy of first determining roots to 4-dp accuracy and then improving accuracy to 14-dp (total iters = 16 here) takes less iteration steps than the hybrid method(iters = 71 and 56).

## 2 Problem2: Find minimum

### 2.1 problem description

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

Clearly  $g_{min} = -2$ , and is reached when  $x = m\pi, y = \frac{2n+1}{2}\pi$  with  $m + n$  being odd.

### 2.2 algorithm description

Here I use steepest-descent method to find the minimum of  $g(x, y)$  from different initial guess points. This method updates  $(x, y)$  in the opposite direction of gradient of  $g(x, y)$

,the algorithm is shown below:

$$x_{k+1} = x_k - a \cdot \nabla g(x_k) \quad (3)$$

here  $x_k$  can represent high dimensional vector:  $x_k = (x_1, x_2, \dots, x_n)_k$ ,  $k$  means iteration step index.  $a$  denotes the step length, and it's variable during the iteration. The explanation is given below:

---

```
1 Inputs:
2   f(x): objective function
3   g(x): gradient of f
4   x0:    initial point
5   tol:   tolerance for convergence
6   lr:    initial learning rate (step length)
7   min_lr: minimal allowed step length
8   p in (0,1): shrink factor for backtracking, e.g., 0.5
9   e > 1: gentle enlarge factor for next round (optional), e.g. 1.5
10
11 Algorithm:
12 1. x = x0
13    f0 = f(x)
14    k = 0
15    cur_lr = lr
16
17 2. main loop:
18    2.1 gk = g(x)
19        if |gk|^2 < tol:
20            return (x, f0, k)
21
22    2.2 d = -gk                # steepest descent direction
23
24    2.3 (Backtracking line search)
25        decreased = false
26        a = cur_lr
27        while a > min_lr:
28            x_try = x + a d
29            f_try = f(x_try)
30            if f_try < f0:      # sufficient decrease (simple check)
31                decreased ← true
32                break
33            a = p·a             # shrink step (p in (0,1))
34        if decreased = false:
35            return (x, f0, k)  # step too small and no decrease ->
                                # consider converged
36
37    2.4 dx = |x_try - x|
38        df = |f_try - f0|
39
40    2.5 x = x_try
41        f0 = f_try
42        k = k + 1
43
```

```

44     2.6 if dx < tol or df < tol:
45         return (x, f0, k)
46
47     2.7 cur_lr = min(e*a, lr) # gently enlarge base step for next
        round

```

---

## 2.3 Output

run code `problem2.py`, you can set different start point as well as learning-rate.

```

• Steepest Descent method, initial point: (7.4, -3.6) lr: 0.2
  final (x,y) = (6.2831979686, -1.5708041520), min g(x,y) = -2.0000000000, iters = 403
  Steepest Descent method, initial point: (13.4, -5.6) lr: 0.1
  final (x,y) = (18.8495376863, -7.8539703640), min g(x,y) = -2.0000000000, iters = 885

```

Figure 3: steepest-descent method output

As can be seen, different initial points can lead to different final points, but the minimum value of  $g(x, y)$  is the same. Generally, smaller lr means more iteration steps.

## 3 Problem3: Find eigen-states

### 3.1 Problem description

Electron in the finite square-well potential is:

$$V(x) = \begin{cases} V_0 & x \leq -a \quad \text{Region I} \\ 0 & -a < x < a \quad \text{Region II} \\ V_0 & x \geq a \quad \text{Region III} \end{cases} \quad V_0 = 10 \text{ eV}, a = 0.2 \text{ nm}$$

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

### 3.2 Algorithm description

If energy  $0 < E < V_0$ , the wave function has the following forms:

$$\begin{cases} \psi_I(x) = Ae^{\kappa x}, & x < -a, & \kappa = \sqrt{\frac{2m(V_0 - E)}{\hbar^2}}, \\ \psi_{II}(x) = B \sin(kx) + C \cos(kx), & -a < x < a, & k = \sqrt{\frac{2mE}{\hbar^2}}, \\ \psi_{III}(x) = De^{-\kappa x}, & x > a. \end{cases} \quad (4)$$

There are four coefficients  $A, B, C, D$ . Continuity of the wave function and its derivative at  $x = \pm a$  give boundary conditions:

$$\psi_I(-a) = \psi_{II}(-a), \quad \psi'_I(-a) = \psi'_{II}(-a), \quad (5)$$

$$\psi_{II}(a) = \psi_{III}(a), \quad \psi'_{II}(a) = \psi'_{III}(a). \quad (6)$$

In addition, the normalization condition is

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1. \quad (7)$$

This gives five equations to determine  $A, B, C, D$  and  $E$ . From (5)(6), we can get:

$$\begin{cases} (\frac{k}{\kappa} - \tan(ka))B + (\frac{k}{\kappa} \tan(ka) + 1)C = 0 \\ (\frac{k}{\kappa} + \tan(ka))B + (-\frac{k}{\kappa} \tan(ka) + 1)C = 0 \end{cases} \quad (8)$$

In order that  $B, C$  has nontrivial solutions,  $\det = 0$  for the coefficients, we can thus simplify the question as two independent equations (corresponding to the two cases of odd functions and even functions) and solve them separately:

$$\begin{cases} k \tan(ka) = \kappa & \text{even} \\ k \cot(ka) = -\kappa & \text{odd} \end{cases} \quad (9)$$

These transcendental equations determine the allowed bound-state energies  $E$ , and  $A, B, C, D$  can then be determined once  $E$  is defined.

To solve the equation(9), I adopt bisection method, and to avoid singularity points  $\frac{(n+1)\pi}{2}, n \in \mathbb{Z}$ , at which  $\tan(x), \cot(x) \rightarrow 0$  or  $\infty$ , here I divide the positive number interval into open intervals  $(\frac{n\pi}{2}, \frac{(n+1)\pi}{2}), n \in \mathbb{Z}^+$  for bisection method to search in so that the singularity points are excluded.

### 3.3 Output

run code `problem3.py`:

```
Bounded eigenstates (coefficients A,B,C,D and energy E):
n= 1  parity=even  E = 1.356892494283 eV
      A = 4.589358e+05, B = 0.000000e+00, C = 6.126862e+04, D = 4.589358e+05
n= 2  parity= odd  E = 5.198969709998 eV
      A = -4.003890e+05, B = 5.881510e+04, C = 0.000000e+00, D = 4.003890e+05
n= 3  parity=even  E = 9.925406350646 eV
      A = -4.357884e+04, B = 0.000000e+00, C = 3.306479e+04, D = -4.357884e+04
```

Figure 4: output of 3 eigenstates

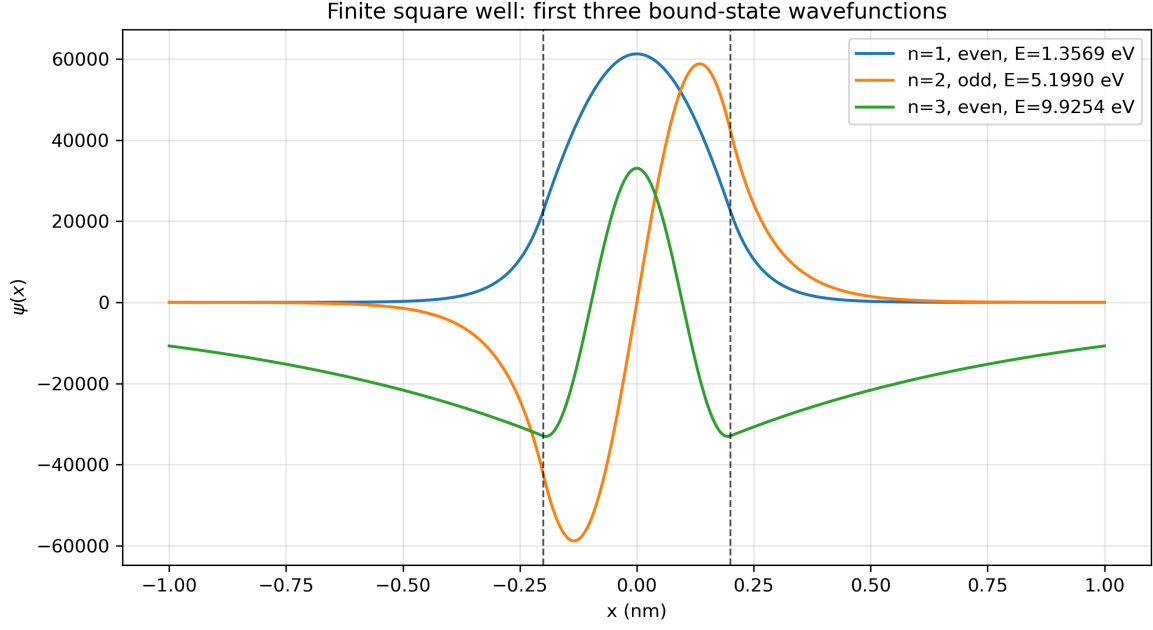


Figure 5: wavefunction

From Figure 4 and Figure 5, we can see that for given parameter  $V_0 = 10\text{eV}$ ,  $a = 0.2\text{nm}$ , there are 3 bounded eigenstates in the well. The coefficients  $A, B, C, D$  of eigen-functions in equation(4) are given as well.