Numerical Analysis Coursework

YAN WENHAO K2478001

July 19, 2024

Abstract

This report shows the solutions of the student projects of Numerical Analysis Course. Respectively, the Problem I is solved by a **binary search** algorithm, which shares a similar philosophy with bracketing numerical methods, while the Problem II is addressed by the **finite element method**. To make the content clear and easy-to-read, **only the main parts or functions of Python code are attached to this report**. The complete program is available in the github repository at https://github.com/yanwunhao/numerical-analysis-coursework

1 Solution to Problem I

First, rewrite the *van der Waals equation* into the format we expect:

$$\rho v - \rho b + \frac{a}{v} - \frac{ab}{v^2} - RT = 0$$

Then declare function $F(v)$ looking like:

$$F(v) = \rho v - \rho b + \frac{a}{v} - \frac{ab}{v^2} - RT$$

Finally, the solution turns out to be searching the root of F(v). The next step is introducing the provided parameter, F(v), which becomes the following:

$$F(v) = 2v - 0.16814 + \frac{12.2}{v} - \frac{1.0102314}{v^2} - 30.77025$$

With the assistance of the open-source Python library matplotlib, we can obtain the curve of $F(v)$:

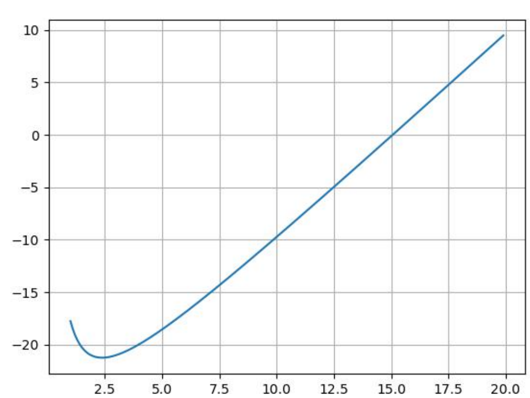

Figure 1: Equally spaced sampling of the function F(v) in the interval [0.1, 20]

From Figure 1, we can notice that the root should be around v=15. Therefore, we can launch a **Binary Search** [Int18] (bracketing numerical method) task between 12 and 16. The Python source code of the searching function is displayed as follows:

```python
# Binary search to solve the root of Fv
def binarysearch(left_bound, right_bound):
    left = left_bound
    right = right_bound
    for i in range(10):
        mid = (left + right)/2.
        fv_left = fv(left)
        fv_mid = fv(mid)
        fv_right = fv(right)
        print(f"Epoch {i}: left point is {left}, right point is {right}")
        if fv_left * fv_mid < 0:
            print(f"Update right bound: {right} -> {mid}")
            right = mid
        else:
            print(f"Update left bound: {left} -> {mid}")
            left = mid
    return (left + right) / 2

result = binarysearch(12, 16)
print(f"The result of binary search is {result}")
```

There will be 10 iterations, and then work out the answer $v = 15.068359375$

2 Solution to Problem II

To solve Problem 2 in the numerical analysis coursework, the two-dimensional Laplace equation in electrostatics needs to be addressed. From here, the **finite element method (FEM)** [Red93] is introduced to work out the solution.

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

Firstly, we should divide the region into small finite elements. For simplicity, we assume a rectangular domain discretized into quadrilateral elements. Assume the domain is discretized into N quadrilateral elements with M nodes. Each node will have a potential value V_i.

Then, convert the Laplace equation into its weak form.

$$\int_\Omega w(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}) = 0$$

$$\int_\Omega w(\frac{\partial w}{\partial x}\frac{\partial V}{\partial x} + \frac{\partial w}{\partial y}\frac{\partial V}{\partial y})d\Omega = 0$$

For each element, the element stiffness matrix K^e is computed as:

$$K^e = \int_\Omega B^T D B d\Omega$$

where B is the strain-displacement matrix and D is the material property matrix (for electrostatics, $D = \epsilon I$). The global stiffness matrix K and force vector F are assembled by summing the contributions of all elements.

Next, import the given parameters and build the system of linear equations: (1) A voltage of 1200 along the circular boundary. (2) A voltage of 0 along the base. (3) $\epsilon = 2$. The main part of the Python code is shown as follows:

```python
# Apply boundary conditions function
def boundary_conditions(A, b, V_boundary, V_base, nx, ny):
    for i in range(nx):
        for j in range(ny):
            if i == 0 or i == nx - 1 or j == 0 or j == ny - 1:
```

```
            k = i + j * nx
            A[k, :] = 0
            A[k, k] = 1
            if i == 0 or i == nx - 1:
                b[k] = V_base
            else:
                b[k] = V_boundary
    return A, b

# Build a finite element system
for i in range(1, nx - 1):
    for j in range(1, ny - 1):
        k = i + j * nx
        A[k, k] = -4
        A[k, k - 1] = 1
        A[k, k + 1] = 1
        A[k, k - nx] = 1
        A[k, k + nx] = 1
```

Finally, it turns possible to calculate the electric flux density D using the gradient of the potential:

$$D = -\epsilon \bigtriangledown V$$

For each element, the components of D are computed using finite differences:

$$D_x = -\epsilon \frac{\partial V}{\partial x} \approx -\epsilon \frac{V_{i+1,j} - V_{i-1,j}}{2\triangle x}$$

$$D_y = -\epsilon \frac{\partial V}{\partial y} \approx -\epsilon \frac{V_{i+1,j} - V_{i-1,j}}{2\triangle y}$$

The solved V matrix can be drawn as the heating map as follows:

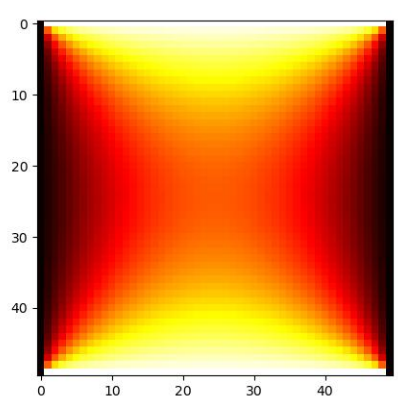

Figure 2: Heating map of solved potential V

where the max value in the matrix is 1200.

References

[Int18] Somkid Intep. A review of bracketing methods for finding zeros of nonlinear functions. *Applied mathematical sciences*, 12:137–146, 2018.

[Red93] Junuthula Narasimha Reddy. An introduction to the finite element method. *New York*, 27:14, 1993.