

# **ZJU Computational Physics: Homework #3**

Due on Monday, December 9, 2024

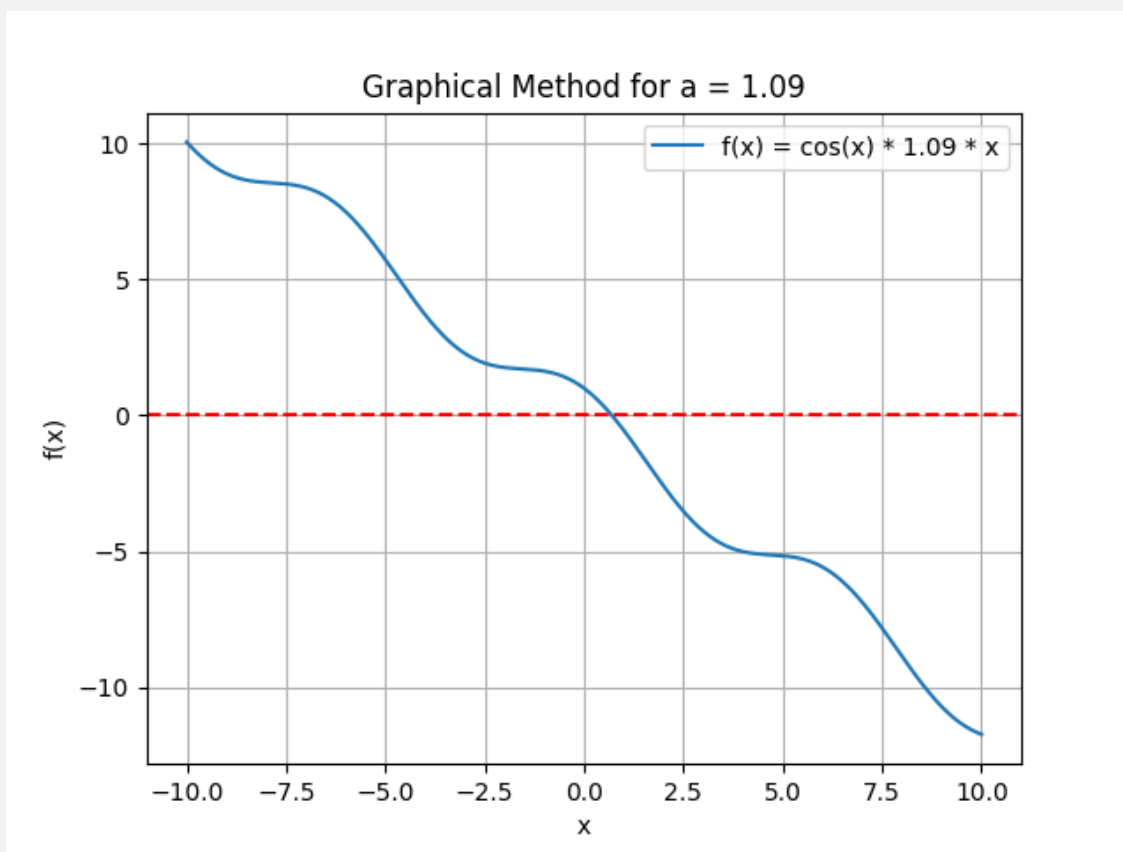
Github: <https://github.com/NAKONAKO4/ZJU-computational-physics-NAKO>

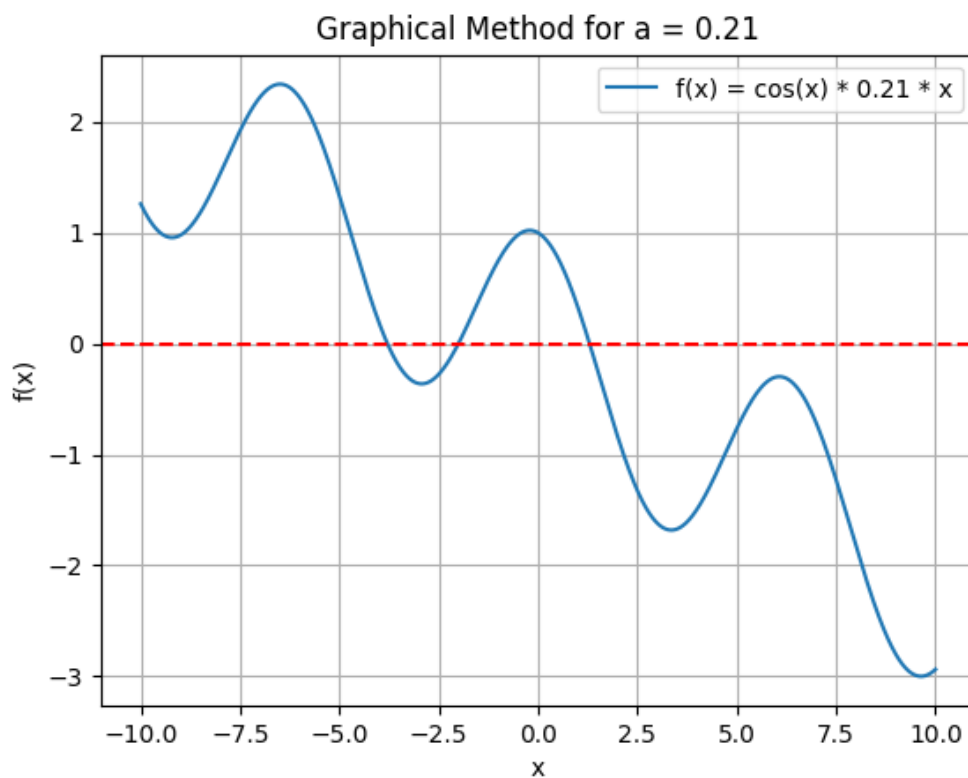
管一沚

## Problem 1

### Root-finding

a.  $a=1.09$  和  $a=0.21$  分别绘图可得:





可得在  $a=1.09$  时只有一根，大约为 0.7；在  $a=0.21$  时有 3 根，大约为 -3.7、-2.0、1.3。

b-f. 运行结果为

```

Bisection Method (5 significant figures): Root1 = -3.79140, Iterations = 17
Bisection Method (8 significant figures): Root1 = -3.79140782, Iterations = 27
Bisection Method (5 significant figures): Root2 = -2.00551, Iterations = 17
Bisection Method (8 significant figures): Root2 = -2.00551915, Iterations = 27
Bisection Method (5 significant figures): Root3 = 1.29531, Iterations = 17
Bisection Method (8 significant figures): Root3 = 1.29530988, Iterations = 27
Newton-Raphson Method (5 significant figures): Root1 = -3.79141, Iterations = 3
Newton-Raphson Method (8 significant figures): Root1 = -3.79140782, Iterations = 4
Newton-Raphson Method (5 significant figures): Root2 = -2.00551, Iterations = 1
Newton-Raphson Method (8 significant figures): Root2 = -2.00551915, Iterations = 2
Newton-Raphson Method (5 significant figures): Root3 = 1.29531, Iterations = 3
Newton-Raphson Method (8 significant figures): Root3 = 1.29530989, Iterations = 3
Secant Method (5 significant figures): Root1 = -3.79141, Iterations = 7
Secant Method (8 significant figures): Root1 = -3.79140782, Iterations = 8
Secant Method (5 significant figures): Root2 = -2.00552, Iterations = 5
Secant Method (8 significant figures): Root2 = -2.00551915, Iterations = 5
Secant Method (5 significant figures): Root3 = 1.29531, Iterations = 5
Secant Method (8 significant figures): Root3 = 1.29530988, Iterations = 6
False Position Method (5 significant figures): Root1 = -3.79141, Iterations = 10000
False Position Method (8 significant figures): Root1 = -3.79140782, Iterations = 10000
False Position Method (5 significant figures): Root2 = -2.00552, Iterations = 10
False Position Method (8 significant figures): Root2 = -2.00551915, Iterations = 10
False Position Method (5 significant figures): Root3 = 1.29531, Iterations = 8
False Position Method (8 significant figures): Root3 = 1.29530988, Iterations = 8
Simple Iteration Method (5 significant figures): Root1 = -3.79141, Iterations = 11
Simple Iteration Method (8 significant figures): Root1 = -3.79140782, Iterations = 18
Simple Iteration Method (5 significant figures): Root2 = -2.00552, Iterations = 1
Simple Iteration Method (8 significant figures): Root2 = 1.29530988, Iterations = 51
Simple Iteration Method (5 significant figures): Root3 = 1.29531, Iterations = 5
Simple Iteration Method (8 significant figures): Root3 = 1.29530988, Iterations = 8

```

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x, a):
5     return np.cos(x) - a * x
6
7 def f_prime(x, a):
8     return -np.sin(x) - a
9
10 def bisection_method(func, a, lower, upper, tol, max_iter):
11     iterations = 0
12     while (upper - lower) / 2 > tol and iterations < max_iter:
13         midpoint = (lower + upper) / 2
14         if func(midpoint, a) == 0:
15             return midpoint, iterations
16         elif func(lower, a) * func(midpoint, a) < 0:
17             upper = midpoint
18         else:
19             lower = midpoint
20     iterations += 1

```

```

21     return (lower + upper) / 2, iterations
22
23
24 def newton_raphson(func, func_prime, a, x0, tol, max_iter):
25     x = x0
26     iterations = 0
27     while abs(func(x, a)) > tol and iterations < max_iter:
28         x = x - func(x, a) / func_prime(x, a)
29         iterations += 1
30     return x, iterations
31
32
33 def secant_method(func, a, x0, x1, tol, max_iter):
34     iterations = 0
35     while abs(x1 - x0) > tol and iterations < max_iter:
36         x_temp = x1 - func(x1, a) * (x1 - x0) / (func(x1, a) - func(x0, a))
37         x0, x1 = x1, x_temp
38         iterations += 1
39     return x1, iterations
40
41
42 def false_position_method(func, a, lower, upper, tol, max_iter):
43     iterations = 0
44     while abs(upper - lower) > tol and iterations < max_iter:
45         root = upper - func(upper, a) * (upper - lower) / (func(upper, a) - func(lower, a))
46         if func(root, a) == 0:
47             return root, iterations
48         elif func(lower, a) * func(root, a) < 0:
49             upper = root
50         else:
51             lower = root
52         iterations += 1
53     return root, iterations
54
55 def simple_iteration(a, x0, lambda_, epsilon, max_iter):
56     x_prev = x0
57     for i in range(max_iter):
58         x_next = x_prev + lambda_ * (np.cos(x_prev) - a * x_prev)
59         if abs(x_next - x_prev) < epsilon:
60             return x_next, i + 1
61         x_prev = x_next
62     return None, max_iter
63
64 a_values = [1.09, 0.21]
65
66 for a in a_values:
67     x = np.linspace(-10, 10, 1000)

```

```

68     y = f(x, a)
69
70     plt.figure()
71     plt.plot(x, y, label=f"f(x)=cos(x)*{a}*x")
72     plt.axhline(0, color="red", linestyle="--")
73     plt.title(f"Graphical_Method_for_a={a}")
74     plt.xlabel("x")
75     plt.ylabel("f(x)")
76     plt.grid()
77     plt.legend()
78     plt.show()
79
80     tol_5sf = 1e-5
81     tol_8sf = 1e-8
82     max_iter = 10000
83
84     a = 0.21
85     lower, upper = -2.5, 0
86
87     # BISECTION METHOD
88     root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, -5, -2.5, tol_5sf, max_iter)
89     root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, -5, -2.5, tol_8sf, max_iter)
90     print(f"Bisection_Method_(5_significant_figures):_Root1={root_bisection_5sf:.5f},_
          Iterations={iter_bisection_5sf}")
91     print(f"Bisection_Method_(8_significant_figures):_Root1={root_bisection_8sf:.8f},_
          Iterations={iter_bisection_8sf}")
92
93     root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, -2.5, 0, tol_5sf, max_iter)
94     root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, -2.5, 0, tol_8sf, max_iter)
95     print(f"Bisection_Method_(5_significant_figures):_Root2={root_bisection_5sf:.5f},_
          Iterations={iter_bisection_5sf}")
96     print(f"Bisection_Method_(8_significant_figures):_Root2={root_bisection_8sf:.8f},_
          Iterations={iter_bisection_8sf}")
97
98     root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, 0, 2.5, tol_5sf, max_iter)
99     root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, 0, 2.5, tol_8sf, max_iter)
100    print(f"Bisection_Method_(5_significant_figures):_Root3={root_bisection_5sf:.5f},_
          Iterations={iter_bisection_5sf}")
101    print(f"Bisection_Method_(8_significant_figures):_Root3={root_bisection_8sf:.8f},_
          Iterations={iter_bisection_8sf}")
102
103    # NEWTON-RAPHSON METHOD
104    root_newton, iter_newton = newton_raphson(f, f_prime, a, -5, tol_5sf, max_iter)
105    print(f"Newton-Raphson_Method_(5_significant_figures):_Root1={root_newton:.5f},_Iterations
          _={iter_newton}")
106    root_newton, iter_newton = newton_raphson(f, f_prime, a, -5, tol_8sf, max_iter)
107    print(f"Newton-Raphson_Method_(8_significant_figures):_Root1={root_newton:.8f},_Iterations

```

```

        _={iter_newton}")
108
109 root_newton, iter_newton = newton_raphson(f, f_prime, a, -2, tol_5sf, max_iter)
110 print(f"Newton-Raphson_Method_(5_significant_figures):_Root2=_{root_newton:.5f},_Iterations
        _={iter_newton}")
111 root_newton, iter_newton = newton_raphson(f, f_prime, a, -2, tol_8sf, max_iter)
112 print(f"Newton-Raphson_Method_(8_significant_figures):_Root2=_{root_newton:.8f},_Iterations
        _={iter_newton}")
113
114 root_newton, iter_newton = newton_raphson(f, f_prime, a, 2, tol_5sf, max_iter)
115 print(f"Newton-Raphson_Method_(5_significant_figures):_Root3=_{root_newton:.5f},_Iterations
        _={iter_newton}")
116 root_newton, iter_newton = newton_raphson(f, f_prime, a, 2, tol_8sf, max_iter)
117 print(f"Newton-Raphson_Method_(8_significant_figures):_Root3=_{root_newton:.8f},_Iterations
        _={iter_newton}")
118
119 # SECANT METHOD
120 x0, x1 = -5, -3
121 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
122 print(f"Secant_Method_(5_significant_figures):_Root1=_{root_secant:.5f},_Iterations=_{
        iter_secant}")
123 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_8sf, max_iter)
124 print(f"Secant_Method_(8_significant_figures):_Root1=_{root_secant:.8f},_Iterations=_{
        iter_secant}")
125 x0, x1 = -2.5, 0
126 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
127 print(f"Secant_Method_(5_significant_figures):_Root2=_{root_secant:.5f},_Iterations=_{
        iter_secant}")
128 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_8sf, max_iter)
129 print(f"Secant_Method_(8_significant_figures):_Root2=_{root_secant:.8f},_Iterations=_{
        iter_secant}")
130 x0, x1 = 0, 2
131 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
132 print(f"Secant_Method_(5_significant_figures):_Root3=_{root_secant:.5f},_Iterations=_{
        iter_secant}")
133 root_secant, iter_secant = secant_method(f, a, x0, x1, tol_8sf, max_iter)
134 print(f"Secant_Method_(8_significant_figures):_Root3=_{root_secant:.8f},_Iterations=_{
        iter_secant}")
135
136 # FALSE POSITION METHOD
137 root_false_position, iter_false_position = false_position_method(f, a, -5, -3, tol_5sf,
        max_iter)
138 print(f"False_Position_Method_(5_significant_figures):_Root1=_{root_false_position:.5f},_
        Iterations=_{iter_false_position}")
139 root_false_position, iter_false_position = false_position_method(f, a, -5, -3, tol_8sf,
        max_iter)
140 print(f"False_Position_Method_(8_significant_figures):_Root1=_{root_false_position:.8f},_

```

```

    Iterations_={iter_false_position}")
141
142 root_false_position, iter_false_position = false_position_method(f, a, -2.5, 0, tol_5sf,
    max_iter)
143 print(f"False_Position_Method_(5_significant_figures):_Root2_={root_false_position:.5f},_
    Iterations_={iter_false_position}")
144 root_false_position, iter_false_position = false_position_method(f, a, -2.5, 0, tol_8sf,
    max_iter)
145 print(f"False_Position_Method_(8_significant_figures):_Root2_={root_false_position:.8f},_
    Iterations_={iter_false_position}")
146
147 root_false_position, iter_false_position = false_position_method(f, a, 0, 2.5, tol_5sf,
    max_iter)
148 print(f"False_Position_Method_(5_significant_figures):_Root3_={root_false_position:.5f},_
    Iterations_={iter_false_position}")
149 root_false_position, iter_false_position = false_position_method(f, a, 0, 2.5, tol_8sf,
    max_iter)
150 print(f"False_Position_Method_(8_significant_figures):_Root3_={root_false_position:.8f},_
    Iterations_={iter_false_position}")
151
152 # SIMPLE ITERATION METHOD
153 root_simple, iter_simple = simple_iteration(a, -5, 0.8, tol_5sf, max_iter)
154 print(f"Simple_Iteration_Method_(5_significant_figures):_Root1_={root_simple:.5f},_
    Iterations_={iter_simple}")
155 root_simple, iter_simple = simple_iteration(a, -5, 0.8, tol_8sf, max_iter)
156 print(f"Simple_Iteration_Method_(8_significant_figures):_Root1_={root_simple:.8f},_
    Iterations_={iter_simple}")
157
158 root_simple, iter_simple = simple_iteration(a, -2.00551913, 0.8, tol_5sf, max_iter)
159 print(f"Simple_Iteration_Method_(5_significant_figures):_Root2_={root_simple:.5f},_
    Iterations_={iter_simple}")
160 root_simple, iter_simple = simple_iteration(a, -2.00551913, 0.8, tol_8sf, max_iter)
161 print(f"Simple_Iteration_Method_(8_significant_figures):_Root2_={root_simple:.8f},_
    Iterations_={iter_simple}")
162
163 root_simple, iter_simple = simple_iteration(a, 2, 0.8, tol_5sf, max_iter)
164 print(f"Simple_Iteration_Method_(5_significant_figures):_Root3_={root_simple:.5f},_
    Iterations_={iter_simple}")
165 root_simple, iter_simple = simple_iteration(a, 2, 0.8, tol_8sf, max_iter)
166 print(f"Simple_Iteration_Method_(8_significant_figures):_Root3_={root_simple:.8f},_
    Iterations_={iter_simple}")

```

---

## Problem 2

### Bifurcation Diagram



```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 P=np.linspace(0.6,1,20000)
4 X = []
5 Y = []
6 for u in P:
7
8     m = np.random.random()
9     for n in range(2000):
10         m=(u*m)*(1-m)*4
11
12     for n in range(300):
13         m=(u*m)*(1-m)*4
14         Y.append(m)
15         X.append(u)
16
17 plt.figure(figsize=(12, 8))
18 plt.plot(X, Y, ',k', alpha=0.1)
19 plt.title("Bifurcation Diagram")
20 plt.xlabel("r")
21 plt.ylabel("x")
22 plt.xlim([0.6, 1.0])
23 plt.ylim([0, 1])
24 plt.tight_layout()
25 plt.show()
```

运行结果为：

