ZJU Computational Physics: Homework #3

Due on Monday, December 9, 2024

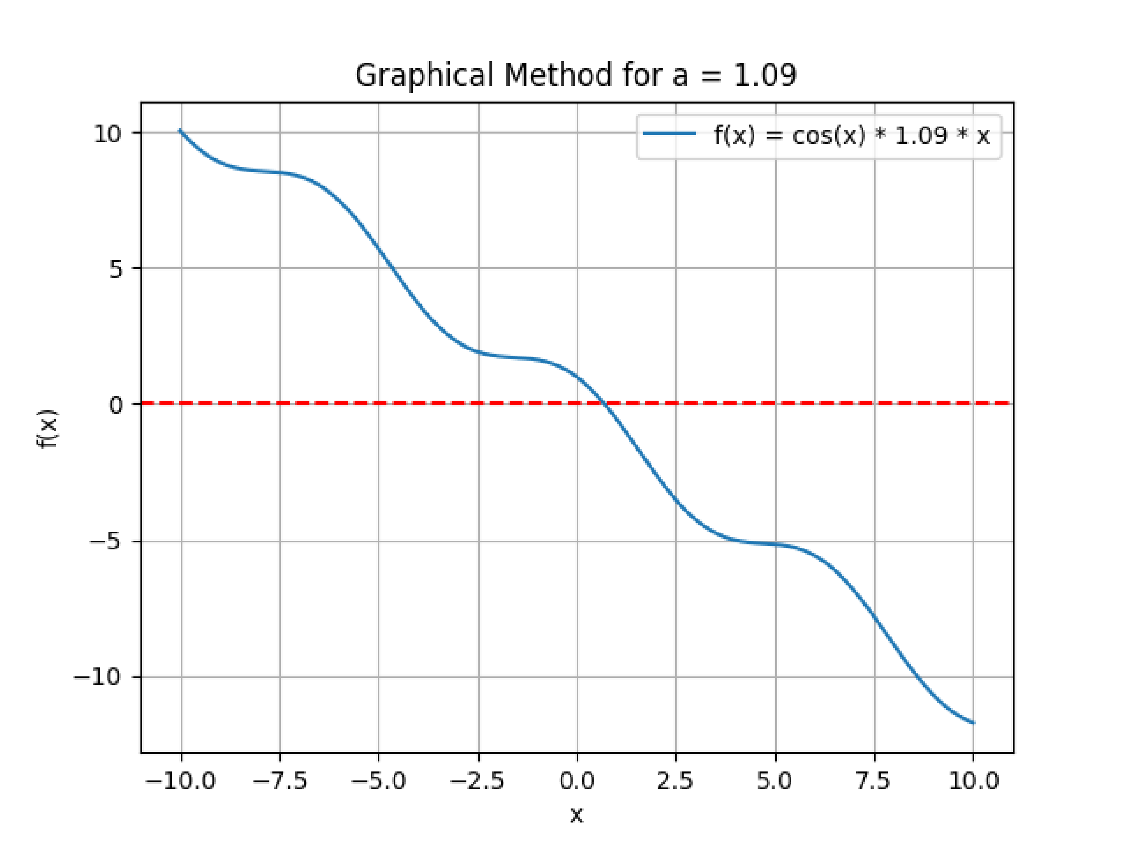
Github: https://github.com/NAKONAKO4/ZJU-computational-physics-NAKO

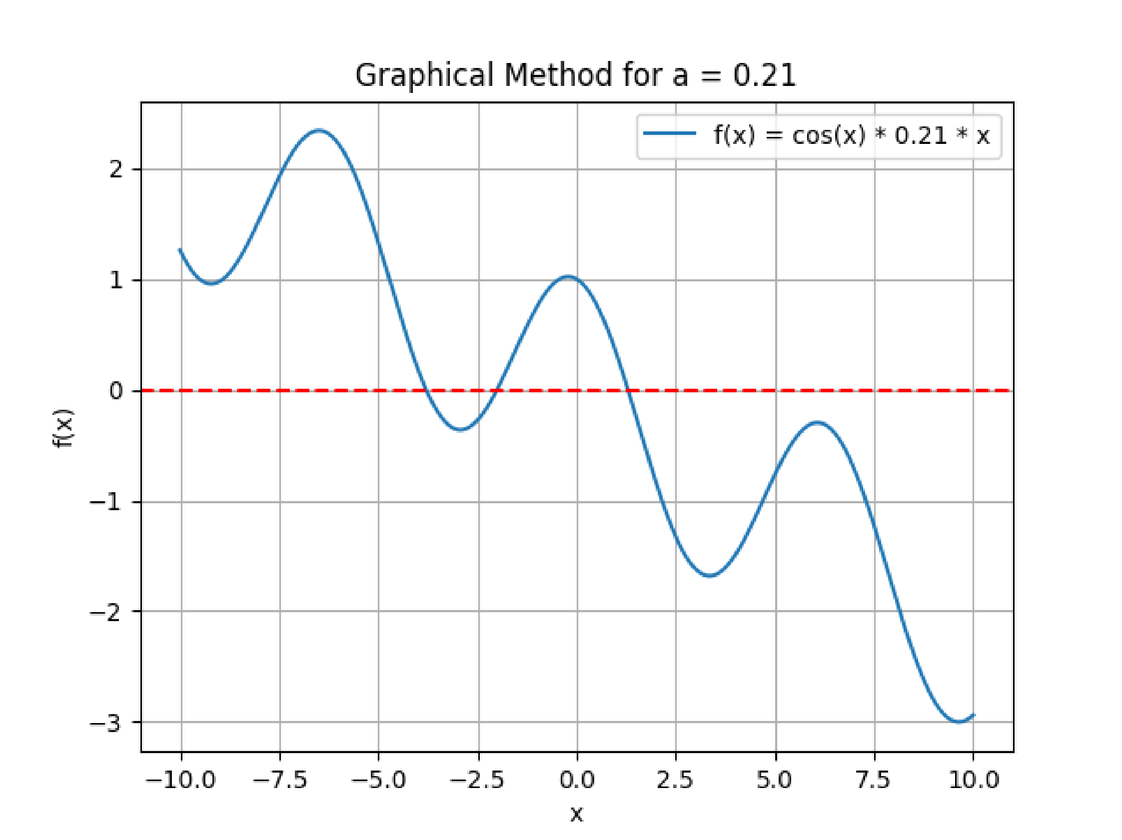
NAKO

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

Problem 1 continued on next page...

```
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
```

```python
import numpy as np
import matplotlib.pyplot as plt

def f(x, a):
    return np.cos(x) - a * x

def f_prime(x, a):
    return -np.sin(x) - a

def bisection_method(func, a, lower, upper, tol, max_iter):
    iterations = 0
    while (upper - lower) / 2 > tol and iterations < max_iter:
        midpoint = (lower + upper) / 2
        if func(midpoint, a) == 0:
            return midpoint, iterations
        elif func(lower, a) * func(midpoint, a) < 0:
            upper = midpoint
        else:
            lower = midpoint
        iterations += 1
```

```python
    return (lower + upper) / 2, iterations


def newton_raphson(func, func_prime, a, x0, tol, max_iter):
    x = x0
    iterations = 0
    while abs(func(x, a)) > tol and iterations < max_iter:
        x = x - func(x, a) / func_prime(x, a)
        iterations += 1
    return x, iterations


def secant_method(func, a, x0, x1, tol, max_iter):
    iterations = 0
    while abs(x1 - x0) > tol and iterations < max_iter:
        x_temp = x1 - func(x1, a) * (x1 - x0) / (func(x1, a) - func(x0, a))
        x0, x1 = x1, x_temp
        iterations += 1
    return x1, iterations


def false_position_method(func, a, lower, upper, tol, max_iter):
    iterations = 0
    while abs(upper - lower) > tol and iterations < max_iter:
        root = upper - func(upper, a) * (upper - lower) / (func(upper, a) - func(lower, a))
        if func(root, a) == 0:
            return root, iterations
        elif func(lower, a) * func(root, a) < 0:
            upper = root
        else:
            lower = root
        iterations += 1
    return root, iterations

def simple_iteration(a, x0, lambda_, epsilon, max_iter):
    x_prev = x0
    for i in range(max_iter):
        x_next = x_prev + lambda_ * (np.cos(x_prev) - a * x_prev)
        if abs(x_next - x_prev) < epsilon:
            return x_next, i + 1
        x_prev = x_next
    return None, max_iter

a_values = [1.09, 0.21]

for a in a_values:
    x = np.linspace(-10, 10, 1000)
```

Problem 1 continued on next page...

```
    y = f(x, a)

    plt.figure()
    plt.plot(x, y, label=f"f(x) = cos(x) * {a} * x")
    plt.axhline(0, color="red", linestyle="--")
    plt.title(f"Graphical Method for a = {a}")
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.grid()
    plt.legend()
    plt.show()

tol_5sf = 1e-5
tol_8sf = 1e-8
max_iter = 10000

a = 0.21
lower, upper = -2.5, 0

# Bisection method
root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, -5, -2.5, tol_5sf, max_iter)
root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, -5, -2.5, tol_8sf, max_iter)
print(f"Bisection Method (5 significant figures): Root1 = {root_bisection_5sf:.5f}, Iterations = {iter_bisection_5sf}")
print(f"Bisection Method (8 significant figures): Root1 = {root_bisection_8sf:.8f}, Iterations = {iter_bisection_8sf}")

root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, -2.5, 0, tol_5sf, max_iter)
root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, -2.5, 0, tol_8sf, max_iter)
print(f"Bisection Method (5 significant figures): Root2 = {root_bisection_5sf:.5f}, Iterations = {iter_bisection_5sf}")
print(f"Bisection Method (8 significant figures): Root2 = {root_bisection_8sf:.8f}, Iterations = {iter_bisection_8sf}")

root_bisection_5sf, iter_bisection_5sf = bisection_method(f, a, 0, 2.5, tol_5sf, max_iter)
root_bisection_8sf, iter_bisection_8sf = bisection_method(f, a, 0, 2.5, tol_8sf, max_iter)
print(f"Bisection Method (5 significant figures): Root3 = {root_bisection_5sf:.5f}, Iterations = {iter_bisection_5sf}")
print(f"Bisection Method (8 significant figures): Root3 = {root_bisection_8sf:.8f}, Iterations = {iter_bisection_8sf}")

# Newton-Raphson method
root_newton, iter_newton = newton_raphson(f, f_prime, a, -5, tol_5sf, max_iter)
print(f"Newton-Raphson Method (5 significant figures): Root1 = {root_newton:.5f}, Iterations = {iter_newton}")
root_newton, iter_newton = newton_raphson(f, f_prime, a, -5, tol_8sf, max_iter)
print(f"Newton-Raphson Method (8 significant figures): Root1 = {root_newton:.8f}, Iterations
```

Problem 1 continued on next page...

```python
 = {iter_newton}")

root_newton, iter_newton = newton_raphson(f, f_prime, a, -2, tol_5sf, max_iter)
print(f"Newton-Raphson Method (5 significant figures): Root2 = {root_newton:.5f}, Iterations = {iter_newton}")
root_newton, iter_newton = newton_raphson(f, f_prime, a, -2, tol_8sf, max_iter)
print(f"Newton-Raphson Method (8 significant figures): Root2 = {root_newton:.8f}, Iterations = {iter_newton}")

root_newton, iter_newton = newton_raphson(f, f_prime, a, 2, tol_5sf, max_iter)
print(f"Newton-Raphson Method (5 significant figures): Root3 = {root_newton:.5f}, Iterations = {iter_newton}")
root_newton, iter_newton = newton_raphson(f, f_prime, a, 2, tol_8sf, max_iter)
print(f"Newton-Raphson Method (8 significant figures): Root3 = {root_newton:.8f}, Iterations = {iter_newton}")

# Secant method
x0, x1 = -5, -3
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
print(f"Secant Method (5 significant figures): Root1 = {root_secant:.5f}, Iterations = {iter_secant}")
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_8sf, max_iter)
print(f"Secant Method (8 significant figures): Root1 = {root_secant:.8f}, Iterations = {iter_secant}")
x0, x1 = -2.5, 0
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
print(f"Secant Method (5 significant figures): Root2 = {root_secant:.5f}, Iterations = {iter_secant}")
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
print(f"Secant Method (8 significant figures): Root2 = {root_secant:.8f}, Iterations = {iter_secant}")
x0, x1 = 0, 2
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_5sf, max_iter)
print(f"Secant Method (5 significant figures): Root3 = {root_secant:.5f}, Iterations = {iter_secant}")
root_secant, iter_secant = secant_method(f, a, x0, x1, tol_8sf, max_iter)
print(f"Secant Method (8 significant figures): Root3 = {root_secant:.8f}, Iterations = {iter_secant}")

# False position method
root_false_position, iter_false_position = false_position_method(f, a, -5, -3, tol_5sf, max_iter)
print(f"False Position Method (5 significant figures): Root1 = {root_false_position:.5f}, Iterations = {iter_false_position}")
root_false_position, iter_false_position = false_position_method(f, a, -5, -3, tol_8sf, max_iter)
print(f"False Position Method (8 significant figures): Root1 = {root_false_position:.8f}, 
```

```python
Iterations = {iter_false_position}")

root_false_position, iter_false_position = false_position_method(f, a, -2.5, 0, tol_5sf, max_iter)
print(f"False Position Method (5 significant figures): Root2 = {root_false_position:.5f}, Iterations = {iter_false_position}")
root_false_position, iter_false_position = false_position_method(f, a, -2.5, 0, tol_8sf, max_iter)
print(f"False Position Method (8 significant figures): Root2 = {root_false_position:.8f}, Iterations = {iter_false_position}")

root_false_position, iter_false_position = false_position_method(f, a, 0, 2.5, tol_5sf, max_iter)
print(f"False Position Method (5 significant figures): Root3 = {root_false_position:.5f}, Iterations = {iter_false_position}")
root_false_position, iter_false_position = false_position_method(f, a, 0, 2.5, tol_8sf, max_iter)
print(f"False Position Method (8 significant figures): Root3 = {root_false_position:.8f}, Iterations = {iter_false_position}")

# Simple iteration method
root_simple, iter_simple = simple_iteration(a, -5, 0.8, tol_5sf, max_iter)
print(f"Simple Iteration Method (5 significant figures): Root1 = {root_simple:.5f}, Iterations = {iter_simple}")
root_simple, iter_simple = simple_iteration(a, -5, 0.8, tol_8sf, max_iter)
print(f"Simple Iteration Method (8 significant figures): Root1 = {root_simple:.8f}, Iterations = {iter_simple}")

root_simple, iter_simple = simple_iteration(a, -2.00551913, 0.8, tol_5sf, max_iter)
print(f"Simple Iteration Method (5 significant figures): Root2 = {root_simple:.5f}, Iterations = {iter_simple}")
root_simple, iter_simple = simple_iteration(a, -2.00551913, 0.8, tol_8sf, max_iter)
print(f"Simple Iteration Method (8 significant figures): Root2 = {root_simple:.8f}, Iterations = {iter_simple}")

root_simple, iter_simple = simple_iteration(a, 2, 0.8, tol_5sf, max_iter)
print(f"Simple Iteration Method (5 significant figures): Root3 = {root_simple:.5f}, Iterations = {iter_simple}")
root_simple, iter_simple = simple_iteration(a, 2, 0.8, tol_8sf, max_iter)
print(f"Simple Iteration Method (8 significant figures): Root3 = {root_simple:.8f}, Iterations = {iter_simple}")
```

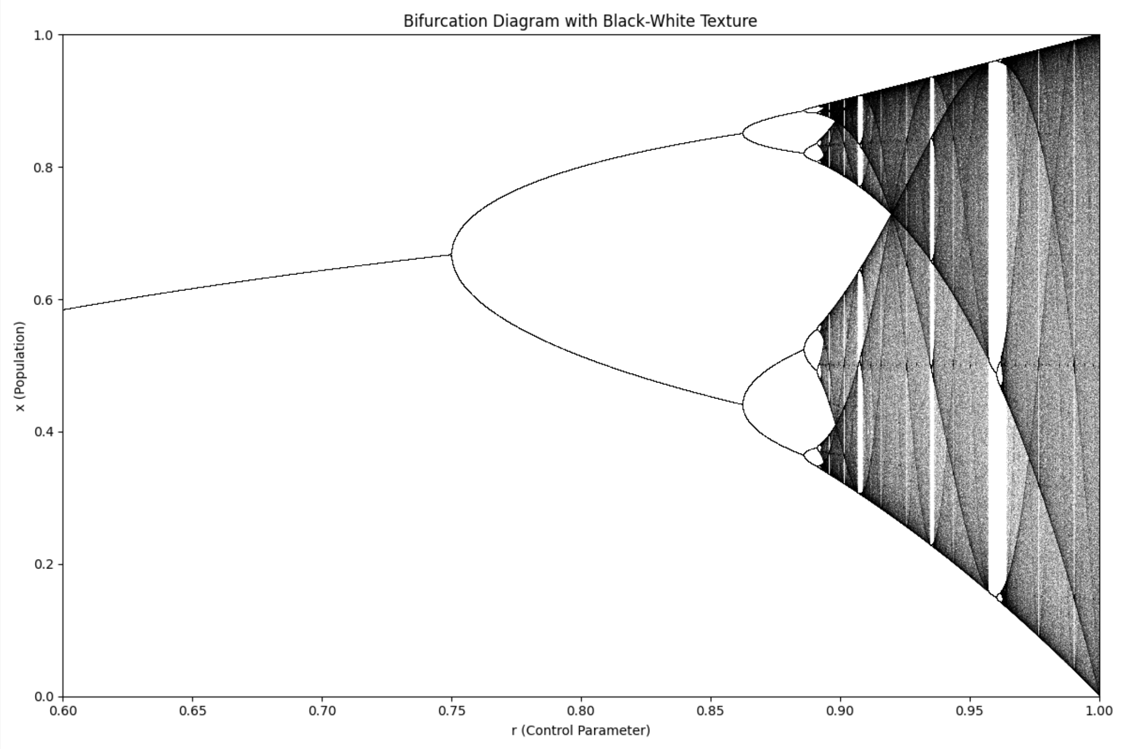
Problem 2

Bifurcation Diagram

```python
import matplotlib.pyplot as plt
import numpy as np
P=np.linspace(0.6,1,20000)
X = []
Y = []
for u in P:

    m = np.random.random()
    for n in range(2000):
        m=(u*m)*(1-m)*4

    for n in range(300):
        m=(u*m)*(1-m)*4
        Y.append(m)
        X.append(u)

plt.figure(figsize=(12, 8))
plt.plot(X, Y, ',k', alpha=0.1)
plt.title("Bifurcation Diagram")
plt.xlabel("r")
plt.ylabel("x")
plt.xlim([0.6, 1.0])
plt.ylim([0, 1])
plt.tight_layout()
plt.show()
```

运行结果为:

Problem 2 continued on next page...


Bifurcation Diagram with Black-White Texture
x (Population)
r (Control Parameter)
0.0
0.2
0.4
0.6
0.8
1.0
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00