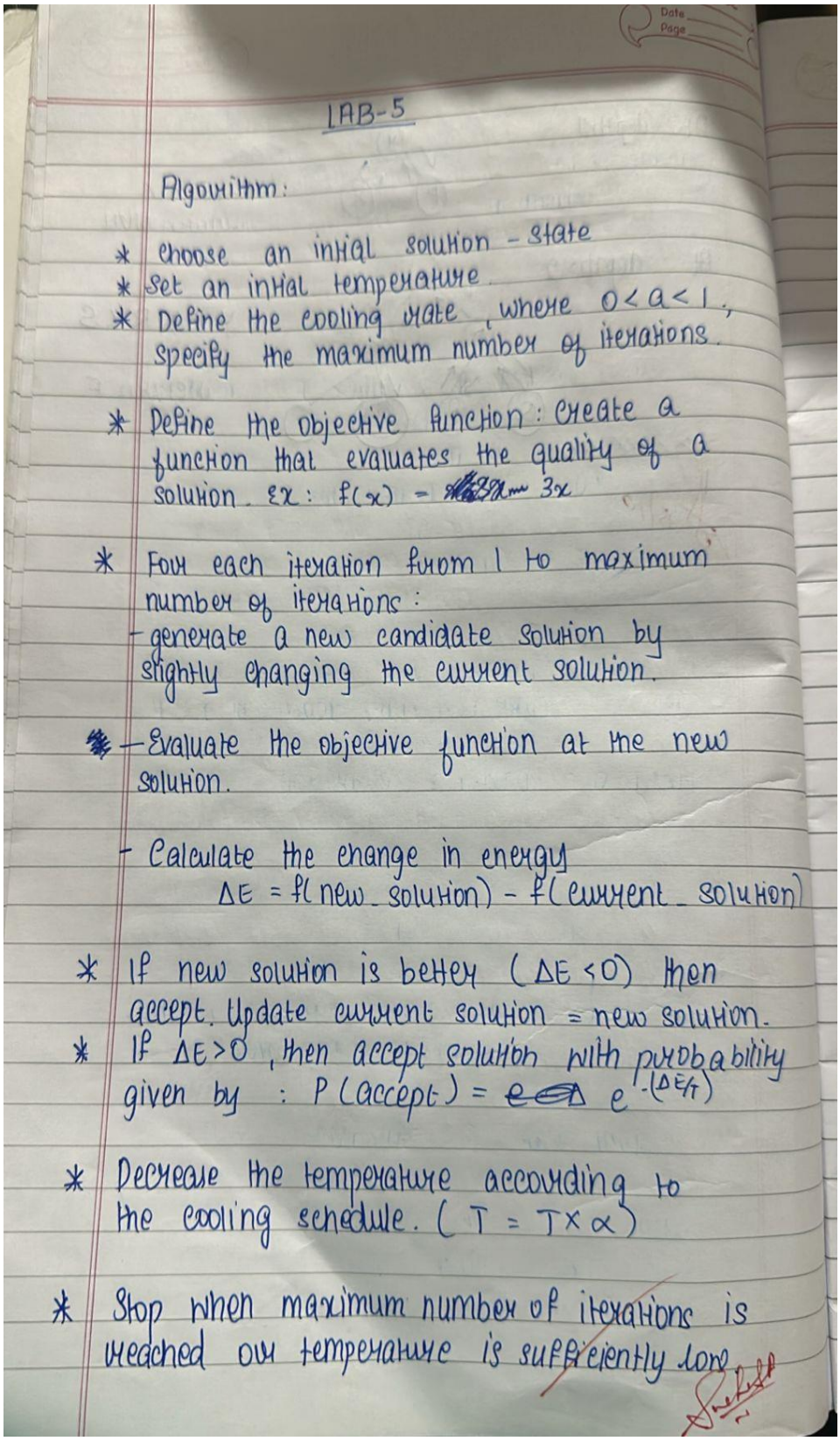
LAB 5: Simulated Annealing Algorithm

Observation book:



LAB-5

Algorithm:

* Choose an initial solution - state
* Set an initial temperature.
* Define the cooling rate, where $0 < \alpha < 1$, specify the maximum number of iterations.
* Define the objective function: create a function that evaluates the quality of a solution. Ex: $f(x) = 3x$
* For each iteration from 1 to maximum number of iterations:
  - generate a new candidate solution by slightly changing the current solution.
  - Evaluate the objective function at the new solution.
  - Calculate the change in energy
    $\Delta E = f(\text{new\_solution}) - f(\text{current\_solution})$
* If new solution is better ($\Delta E < 0$) then accept. Update current solution = new solution.
* If $\Delta E > 0$, then accept solution with probability given by: $P(\text{accept}) = e^{-(\Delta E/T)}$
* Decrease the temperature according to the cooling schedule. ($T = T \times \alpha$)
* Stop when maximum number of iterations is reached or temperature is sufficiently low.

Code:

```
import math
import random

def func(x):
    return 3x

def annealing (initial_s, initial_t, cool_rate,
               max_itns):
    curr_state = initial_s
    curr_val = func(curr_state)
    best_state = curr_state
    best_val = curr_val
    temp = initial_t

for iteration in range(max_itns):
    new_s = curr_state + random.uniform(-1,1)
    new_val = func(new_s)
    delta_val = new_val - curr_val

if delta_value < 0:
    curr_state = new_s
    curr_val = new_val
else:
    acc_prob = math.exp(-delta_val/temp)
    if random.random() < acc_prob:
        curr_state = new_s
        curr_val = new_val

if curr_val < best_val:
    best_state = curr_state
    best_val = curr_val
```

```
    temp *= cool_rate

    print(f"Iteration {iteration+1}: Current
    State = {curr_state:.4f},
    Current value = {curr_val:.4f},
    Best state = {best_state:.4f}
    Best value = {best_val:.4f}")

return best_state, best_val

if __name__ == "__main__":
    initial_s = random.uniform(-10,10)
    inital_t = 100
    cool_rate = 0.95
    max_itns = 100

best_state, best_val = simu annealing(inital_s,
initial_t, cool_rate, max_itns)

print(f"\n Best State Found: {best_state:.4f},
      Best Value: {best_val:.4f}")
```

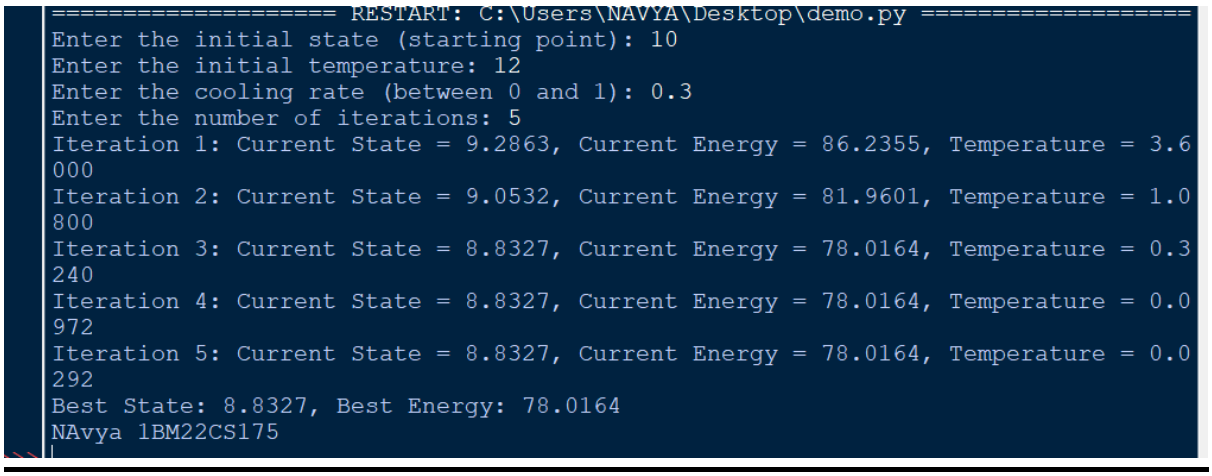
Output:

Iteration 1: Current state = -1.6967, Current Value = 2.8789, Best State = -1.3766, Best Value = 1.8950

Iteration 2: Current state = -2.1096, Current value = 4.4503, Best State = -1.3766, Current value = 1.8950.

22/10/24

Output:


========== RESTART: C:\Users\NAVYA\Desktop\demo.py ==========
Enter the initial state (starting point): 10
Enter the initial temperature: 12
Enter the cooling rate (between 0 and 1): 0.3
Enter the number of iterations: 5
Iteration 1: Current State = 9.2863, Current Energy = 86.2355, Temperature = 3.6
000
Iteration 2: Current State = 9.0532, Current Energy = 81.9601, Temperature = 1.0
800
Iteration 3: Current State = 8.8327, Current Energy = 78.0164, Temperature = 0.3
240
Iteration 4: Current State = 8.8327, Current Energy = 78.0164, Temperature = 0.0
972
Iteration 5: Current State = 8.8327, Current Energy = 78.0164, Temperature = 0.0
292
Best State: 8.8327, Best Energy: 78.0164
NAvya 1BM22CS175