HOMEWORK 3 – PYTHON SIMULATION

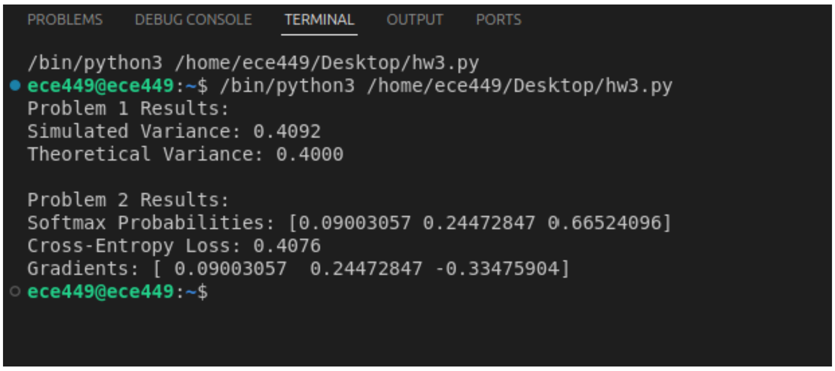
CODE :

```
import numpy as np


### Problem 1: Variance Simulation
def problem_1_variance_simulation(num_samples=100, m=10, mu=0, sigma=2, num_trials=10000):
    """
    Simulates the variance of the sample mean estimator for a Gaussian distribution.
    """
    sample_means = []
    for _ in range(num_trials):
        # Generate 'm' random samples from a Gaussian distribution
        samples = np.random.normal(mu, sigma, size=m)
        sample_mean = np.mean(samples)  # Compute the sample mean
        sample_means.append(sample_mean)

    # Calculate simulated variance
    simulated_variance = np.var(sample_means)

    # Theoretical variance
    theoretical_variance = sigma**2 / m

    return simulated_variance, theoretical_variance



### Problem 2: Neural Network Simulation
# Softmax function
def softmax(z):
    exp_z = np.exp(z - np.max(z))  # Numerical stability
    return exp_z / np.sum(exp_z)
```

```
# Cross-entropy loss function
def cross_entropy_loss(probabilities, y_true):
    return -np.log(probabilities[y_true])

# Gradients with respect to z1, z2, z3
def compute_gradients(probabilities, y_true):
    dz = probabilities.copy()
    dz[y_true] -= 1  # Subtract 1 for the true class
    return dz

def problem_2_neural_network_simulation():
    """
    Simulates forward and backward propagation through a simple neural network
    for a single training sample (x1, x2, y=2).
    """
    # Example inputs
    z = np.array([1.0, 2.0, 3.0])  # Example logits z1, z2, z3
    y_true = 2  # True label index (y = 2)

    # Forward pass: Compute softmax probabilities
    probabilities = softmax(z)

    # Compute cross-entropy loss
    loss = cross_entropy_loss(probabilities, y_true)

    # Backward pass: Compute gradients
    gradients = compute_gradients(probabilities, y_true)

    return probabilities, loss, gradients
```

```
# MAIN EXECUTION
if __name__ == "__main__":
    # Problem 1 Simulation
    simulated_var, theoretical_var = problem_1_variance_simulation()
    print(f"Problem 1 Results:")
    print(f"Simulated Variance: {simulated_var:.4f}")
    print(f"Theoretical Variance: {theoretical_var:.4f}\n")

    # Problem 2 Simulation
    probabilities, loss, gradients = problem_2_neural_network_simulation()
    print(f"Problem 2 Results:")
    print(f"Softmax Probabilities: {probabilities}")
    print(f"Cross-Entropy Loss: {loss:.4f}")
    print(f"Gradients: {gradients}")
```

RESULTS :



```
/bin/python3 /home/ece449/Desktop/hw3.py
● ece449@ece449:~$ /bin/python3 /home/ece449/Desktop/hw3.py
Problem 1 Results:
Simulated Variance: 0.4092
Theoretical Variance: 0.4000

Problem 2 Results:
Softmax Probabilities: [0.09003057 0.24472847 0.66524096]
Cross-Entropy Loss: 0.4076
Gradients: [ 0.09003057  0.24472847 -0.33475904]
○ ece449@ece449:~$
```