M/M/1 Queue Simulation - Code Explanation

Overview

This code simulates an M/M/1 queue system where:

- **M/M/1**: Markovian arrivals/Markovian service/1 server
- Arrivals follow exponential distribution (Poisson process)
- Service times follow exponential distribution
- Single server with FIFO discipline

Code Structure

1. Class Initialization

```python
def __init__(self, arrival_rate, service_rate, sim_time=10000):
```

- Converts hourly rates to per-minute rates
- Initializes queue data structure and server state
- Sets up metric tracking variables

2. Core Simulation Logic

Event-Driven Simulation

```python
while self.time < self.sim_time:
    next_event = min(next_arrival, self.server_end)
```

- Processes events chronologically
- Two event types: arrivals and departures

Arrival Events

```python
if not self.server_busy:
    # Start service immediately
else:
    # Add to queue
```

- If server idle: begin service
- If server busy: join queue

Departure Events

python

```python
if self.queue:
    # Start serving next customer
else:
    # Server becomes idle
```

- Customer leaves system
- Next customer begins service (if queue not empty)

3. Metric Collection

State Tracking

python

```python
def update_state(self):
    duration = self.time - self.last_change
    self.state_times[self.customers_in_system] += duration
```

- Records time spent in each system state
- Used for calculating time-averaged metrics

Key Metrics Calculated

- **Customers served**: Count of completed services
- **Total system time**: Sum of all customer system times
- **Total queue time**: Sum of all customer waiting times
- **Server busy time**: Accumulated service time
- **Average customers**: Time-weighted averages
- **State probabilities**: Proportion of time in each state

4. Exponential Random Variables

python

```python
def exp_time(self, rate):
    return np.random.exponential(1 / rate)
```

- Generates inter-arrival and service times
- Uses numpy's exponential distribution

Simulation Scenarios

Scenario Comparison

Scenario	λ	μ	ρ	Behavior
Original	4	12	0.33	Low utilization, short queues
Scenario 1	6	12	0.50	Moderate utilization
Scenario 2	10.8	12	0.90	High utilization, long queues

Key Observations

- **Queue times grow exponentially** as ρ approaches 1
- **System becomes unstable** at high utilization
- **State probabilities shift** toward higher customer counts

Implementation Features

Simplicity

- Minimal code structure
- Essential functionality only
- Clean metric output

Accuracy

- Proper event ordering
- Correct state transitions
- Time-averaged calculations

Efficiency

- Single simulation loop
- Efficient data structures
- Focused metric collection

Output Interpretation

Each scenario displays:

1. **Customers served**: System throughput
2. **Total times**: Cumulative customer experience
3. **Server busy time**: Resource utilization
4. **Average customers**: System capacity usage
5. **State probabilities**: System behavior distribution

The results demonstrate classic M/M/1 queue theory where performance degrades exponentially as utilization approaches 100%.