 assignment #4 for simulation (CAP 4800) <<<

>>> SOLUTIONS <<<

This assignment covers material from the fourth week of class lecture.

Problem #1 (35 points)

Determine $X$, $Ts$, $U$, $W$, $Wq$, $L$, and $Lq$ for the following single-server queueing system for the time period 0 to 150 seconds. Carefully show your work including all pertinent figures and formulas. **Hint:** Review your week #4 reading (MacDougall, Chapter 1).

- Arrival #1 at time = 10 seconds with service time = 20 seconds
- Arrival #2 at time = 20 seconds with service time = 30 seconds
- Arrival #3 at time = 35 seconds with service time = 10 seconds
- Arrival #4 at time = 80 seconds with service time = 120 seconds
- Arrival #5 at time = 100 seconds with service time = 20 seconds

For solution see next page (scan of handwritten solution)
Solution to Problem #2

\[ T = 150 \text{ s} \]
\[ C = 3 \text{ customers} \]
\[ B = 130 \text{ s} \]

\[ \text{Area} = 215 \leq \frac{3}{2} \]

\[ X = \frac{C}{T} = \frac{3}{150} \text{ customers/s} \]
\[ T_0 = B/C = \frac{130}{3} \text{ s} \]
\[ U = B/T = \frac{130}{150} \]
\[ W = \frac{215}{3} \text{ s} \]
\[ W_d = W - T_0 = \frac{215}{3} - \frac{120}{3} = \frac{85}{3} \text{ s} \]
\[ L = \frac{215}{150} \text{ customers} \]
\[ L_q = L - U = \frac{215}{150} - \frac{130}{150} = \frac{85}{150} \]
**Problem #2** (30 points)

Using the `mm1.c` simulation program we discussed in class (and that is available for download via the class website), simulate the following offered loads for an M/M/1 queue: 50%, 60%, 70%, 80%, 85%, 90%, 91%, 92%, ..., 98%. Fix the service time to be 1.0. For each offered load collect results on the mean number of customers in the system ($L$). Use a SIM_TIME of 200000 seconds. Plot both the simulation results and theory results (based on the formula for $L$ for M/M/1) on one graph. Plot a graph of relative error for simulation to theory versus offered load on another graph. Comment on the relative error. Does it stay the same for all offered loads?

**M/M/1 simulation and theory results**

SIM_TIME = 200000 seconds

<table>
<thead>
<tr>
<th>offered</th>
<th>sim L</th>
<th>theory L</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1.008</td>
<td>1.000</td>
<td>-0.82</td>
</tr>
<tr>
<td>0.6</td>
<td>1.503</td>
<td>1.500</td>
<td>-0.21</td>
</tr>
<tr>
<td>0.7</td>
<td>2.349</td>
<td>2.333</td>
<td>-0.67</td>
</tr>
<tr>
<td>0.8</td>
<td>3.948</td>
<td>4.000</td>
<td>1.31</td>
</tr>
<tr>
<td>0.85</td>
<td>5.713</td>
<td>5.667</td>
<td>-0.82</td>
</tr>
<tr>
<td>0.9</td>
<td>8.793</td>
<td>9.000</td>
<td>2.36</td>
</tr>
<tr>
<td>0.91</td>
<td>10.231</td>
<td>10.111</td>
<td>-1.17</td>
</tr>
<tr>
<td>0.92</td>
<td>10.744</td>
<td>11.500</td>
<td>7.04</td>
</tr>
<tr>
<td>0.93</td>
<td>13.337</td>
<td>13.286</td>
<td>-0.38</td>
</tr>
<tr>
<td>0.94</td>
<td>15.453</td>
<td>15.667</td>
<td>1.38</td>
</tr>
<tr>
<td>0.95</td>
<td>21.541</td>
<td>19.000</td>
<td>-11.80</td>
</tr>
<tr>
<td>0.96</td>
<td>23.153</td>
<td>24.000</td>
<td>3.66</td>
</tr>
<tr>
<td>0.97</td>
<td>34.251</td>
<td>32.333</td>
<td>-5.60</td>
</tr>
<tr>
<td>0.98</td>
<td>37.980</td>
<td>49.000</td>
<td>29.02</td>
</tr>
</tbody>
</table>

The absolute magnitude of relative error increases as offered load increases.
Problem #3 (35 points)

Repeat problem #2 for M/D/1 (of course, you can’t use the formula for \( L \) for M/M/1, you must use the P-K formula correctly). You will need to modify mm1.c to model an M/D/1 queue. In addition to the two plots, also submit your modified mm1.c (perhaps call it md1.c?) source code. Comments on the relative error – is it greater or smaller than for the M/M/1 simulation? Speculate on the “why”.

M/D/1 simulation and theory results

SIM_TIME = 200000 seconds

<table>
<thead>
<tr>
<th>offered</th>
<th>sim L</th>
<th>theory L</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.753</td>
<td>0.750</td>
<td>-0.45</td>
</tr>
<tr>
<td>0.6</td>
<td>1.052</td>
<td>1.050</td>
<td>-0.18</td>
</tr>
<tr>
<td>0.7</td>
<td>1.518</td>
<td>1.517</td>
<td>-0.12</td>
</tr>
<tr>
<td>0.8</td>
<td>2.398</td>
<td>2.400</td>
<td>0.10</td>
</tr>
<tr>
<td>0.85</td>
<td>3.224</td>
<td>3.258</td>
<td>1.05</td>
</tr>
<tr>
<td>0.9</td>
<td>4.891</td>
<td>4.950</td>
<td>1.20</td>
</tr>
<tr>
<td>0.91</td>
<td>5.421</td>
<td>5.511</td>
<td>1.65</td>
</tr>
<tr>
<td>0.92</td>
<td>6.096</td>
<td>6.210</td>
<td>1.88</td>
</tr>
<tr>
<td>0.93</td>
<td>6.906</td>
<td>7.108</td>
<td>2.93</td>
</tr>
<tr>
<td>0.94</td>
<td>7.938</td>
<td>8.303</td>
<td>4.60</td>
</tr>
<tr>
<td>0.95</td>
<td>9.482</td>
<td>9.975</td>
<td>5.20</td>
</tr>
<tr>
<td>0.96</td>
<td>11.977</td>
<td>12.480</td>
<td>4.20</td>
</tr>
<tr>
<td>0.97</td>
<td>15.327</td>
<td>16.652</td>
<td>8.65</td>
</tr>
<tr>
<td>0.98</td>
<td>21.793</td>
<td>24.990</td>
<td>14.67</td>
</tr>
</tbody>
</table>

The absolute magnitude of relative error is less than for the M/M/1 simulation. This is because there is less variability in the M/D/1 case (deterministic service, which has variance = 0) than the M/M/1 case (exponential service time with non-zero variance).
Appendix – M/D/1 simulation program for problem #3

#ifndef MD1_H
#define MD1_H
#define SIM_TIME 200000.0 // Simulation time
#define ARR_TIME (1.0/0.90) // Mean time between arrivals
#define SERV_TIME 1.00 // Mean service time

#include <stdio.h> // Needed for printf()
#include <stdlib.h> // Needed for exit() and rand()
#include <math.h> // Needed for log()

double rand_val(int seed); // RNG for unif(0,1)
double exponential(double x); // Generate exponential RV with mean x

int main(void)
{

double   end_time = SIM_TIME; // Total time to simulate
double   Ta = ARR_TIME; // Mean time between arrivals
double   Ts = SERV_TIME; // Mean service time
double   time = 0.0; // Simulation time
double   t1 = 0.0; // Time for next event #1 (arrival)
double   t2 = SIM_TIME; // Time for next event #2 (departure)
unsigned int n = 0; // Number of customers in the system
unsigned int c = 0; // Number of service completions
double   b = 0.0; // Total busy time
double   s = 0.0; // Area of number of customers in system
double   tn = time; // Variable for "last event time"
double   tb; // Variable for "last start of busy time"
double   x; // Throughput
double   u; // Utilization
double   l; // Mean number in the system
double   w; // Mean residence time

// Seed the RNG
rand_val(1);

// Main simulation loop
while (time < end_time)
{
    if (t1 < t2) // *** Event #1 (arrival) ***
    {

time = t1;
s = s + n * (time - tn);  // Update area under "s" curve
n++;
tn = time;  // tn = "last event time" for next event
t1 = time + exponential(Ta);
if (n == 1)
{
    tb = time;  // Set "last start of busy time"
    t2 = time + Ts;  // NOTE: This line is modified from mm1.c
}
else
    // *** Event #2 (departure) ***
{
    time = t2;
s = s + n * (time - tn);  // Update area under "s" curve
n--;
tn = time;  // tn = "last event time" for next event
c++;  // Increment number of completions
if (n > 0)
t2 = time + Ts;  // NOTE: This line is modified from mm1.c
else
{
    t2 = SIM_TIME;
b = b + time - tb;  // Update busy time sum if empty
}
}

// End of simulation so update busy time sum
b = b + time - tb;

// Compute outputs
x = c / time;  // Compute throughput rate
u = b / time;  // Compute server utilization
l = s / time;  // Compute mean number in system
w = l / x;  // Compute mean residence or system time

// Output results
printf("================================================================== \n");
printf("= *** Results from M/D/1 simulation *** = \n");
printf("================================================================== \n");
printf("= Total simulated time = %3.4f sec \n", end_time);
printf("================================================================== \n");
printf("= INPUTS: \n");
printf("= Mean time between arrivals = %f sec \n", Ta);
printf("= Mean service time = %f sec \n", Ts);
printf("================================================================== \n");
printf("= OUTPUTS: \n");
printf("= Number of completions = %ld cust \n", c);
printf("= Throughput rate = %f cust/sec \n", x);
printf("= Server utilization = %f %\n", 100.0 * u);
printf("= Mean number in system = %f cust \n", l);
printf("= Mean residence time = %f sec \n", w);
printf("================================================================== \n");
return(0);
double rand_val(int seed)
{
    const long a = 16807;  // Multiplier
    const long m = 2147483647;  // Modulus
    const long q = 127773;  // m div a
    const long r = 2836;  // m mod a
    static long x;               // Random int value (seed is set to 1)
    long x_div_q;         // x divided by q
    long x_mod_q;         // x modulo q
    long x_new;           // New x value
    // Seed the RNG
    if (seed != 0) x = seed;
    // RNG using integer arithmetic
    x_div_q = x / q;
    x_mod_q = x % q;
    x_new = (a * x_mod_q) - (r * x_div_q);
    if (x_new > 0)
        x = x_new;
    else
        x = x_new + m;
    // Return a random value between 0.0 and 1.0
    return((double) x / m);
}

double exponential(double x)
{
    double z;                     // Uniform random number from 0 to 1
    // Pull a uniform RV (0 < z < 1)
    do
    {
        z = rand_val(0);
    }
    while ((z == 0) || (z == 1));
    return(-x * log(z));
}