>>> Solution for HW #6 for Capacity Planning (Fall 2001) <<<

```
For this assignment you will compare a simulation of an M/M/1 queue with analytical
results.  You will also compare M/M/1 against M/D/1.

1) Simulate utilization values ranging from 10% to 98% (say, 10% to 90% in steps of 10%
and then 90% to 98% in steps of 1%) for a mean service rate of 1.0 customers per second.
Plot the utilization versus mean customer delay (response time).  Plot also analytical
results for response time (i.e., as computed using the M/M/1 formulas).  How close are
the simulation versus analytical results?  What is the cause of the difference?  How
might the difference be reduced?

2) Change mm1_smpl.c to have deterministic service times (this is now a model of an M/D/1
queue, call it md1_smpl.c).  Rerun part (1) above and plot the results on the same graph
as the M/M/1 results.  Is the response time of an M/D/1 less or greater than that of an
M/D/1?  Why?  Determine a "formula" for the difference in response time between M/M/1 and
M/D/1.
```

Figure 1 shows a plot of response time (W) for M/M/1 simulation and M/D/1 simulation for 10% to 90% offered load and Figure 2 for 90% to 98% offered load. All plotted points result from a simulation run of 10 million seconds (for $\mu = 1.0$ second). The analytical results for M/M/1 are visually "on top" of the simulation results. The analytical W for M/M/1 with $\mu = 1.0$ is simply $W = 1 / (1 - \lambda)$. Figure 3 is a plot of the percentage error for simulation versus analytical (for the full range of 10% to 98% offered load). Appendix A contains the M/M/1 simulation model and Appendix B the M/D/1 model.

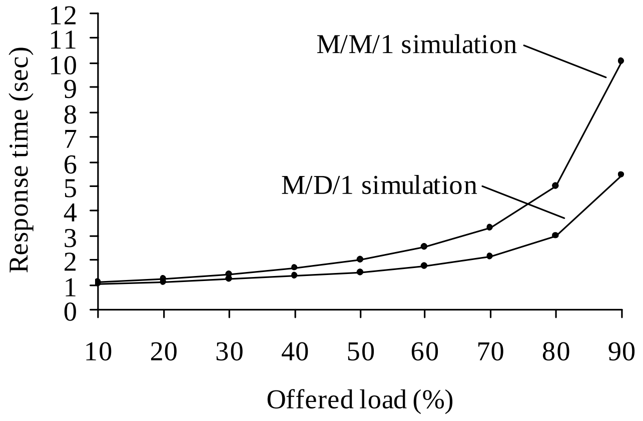

Figure 1- M/M/1 and M/D/1 simulation (10% to 90%)

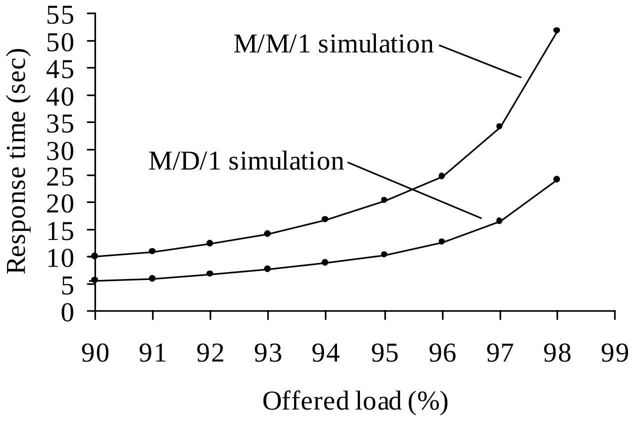

Figure 2 - M/M/1 and M/D/1 simulation (90% to 98%)

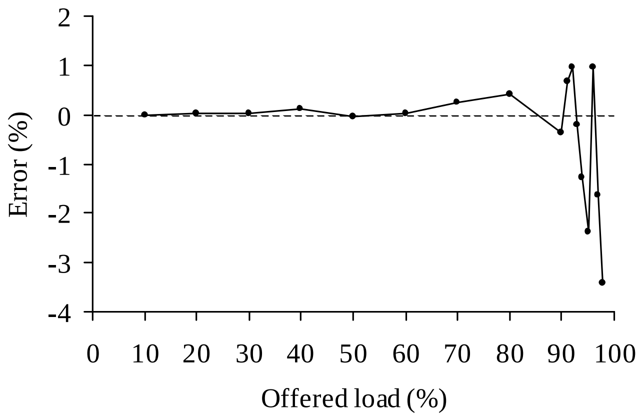

Figure 3- M/M/1 and M/D/1 simulation (10% to 90%)

The simulation and analytical results for the M/M1 case are within +/- 0.50% for offered load up to 80% and then increase upto about +/- 4.0% for higher offered loads. This error is due to the statistical nature of a simulation "sample" which can never produce the exact true population mean (i.e., the analytical result) without having an infinite run time. Greater run times will reduce the errors. For higher offered loads, the variability of response time is greater (than at low offered loads) hence the greater error.

M/D/1 response time is lower than that of M/M/1 due to the lower variance in service time. For high offered loads, the difference between M/M/1 and M/D/1 is approximately a factor of 2. The exact difference can be gleaned from the P-K solution to M/D/1 which is a $(1 - \rho/2)$ factor (or simply $(1 - \lambda/2)$ for the case of $\mu = 1.0$ between W for M/M/1 and M/D/1.

Appendix A - M/M/1 simulation model source code

```c
//========================================================== file = mm1.c =====
//=  A simple M/M/1 queue simulation using SMPL                              =
//=============================================================================
//----- Include files ---------------------------------------------------------
#include <stdio.h>      // Needed for printf()
#include "smpl.h"       // Needed for SMPL

//===== Main program ==========================================================
void main(void)
{
  real Ta;              // Mean interarrival time (seconds)
  real Ts = 1;          // Mean service time (seconds)
  real te = 1.0e7;      // Total simulation time
  int customer = 1;     // Customer id (always ’1’ for this simulation)
  int event;            // Event (1 = arrival, 2 = request, 3 = completion)
  int server;           // Handle for server facility
  int i;                // Loop counter

  for (i=0; i<17; i++)
  {
    // Set Ta value
    if (i ==  0) Ta = 1.0 / 0.10;
    if (i ==  1) Ta = 1.0 / 0.20;
    if (i ==  2) Ta = 1.0 / 0.30;
    if (i ==  3) Ta = 1.0 / 0.40;
    if (i ==  4) Ta = 1.0 / 0.50;
    if (i ==  5) Ta = 1.0 / 0.60;
    if (i ==  6) Ta = 1.0 / 0.70;
    if (i ==  7) Ta = 1.0 / 0.80;
    if (i ==  8) Ta = 1.0 / 0.90;
    if (i ==  9) Ta = 1.0 / 0.91;
    if (i == 10) Ta = 1.0 / 0.92;
    if (i == 11) Ta = 1.0 / 0.93;
    if (i == 12) Ta = 1.0 / 0.94;
    if (i == 13) Ta = 1.0 / 0.95;
    if (i == 14) Ta = 1.0 / 0.96;
    if (i == 15) Ta = 1.0 / 0.97;
    if (i == 16) Ta = 1.0 / 0.98;

    // Initialize SMPL subsystem
    smpl(0, "M/M/1 Queue");

    // Initialize server facility (single server)
    server=facility("server", 1);

    // Schedule arrival event at time 0 to kick-off simulation
    schedule(1, 0.0, customer);

    // Loop while simulation time is less than te
    while (time() < te)
    {
      // "Cause" the next event on the event list
      cause(&event,&customer);

      // Process the event
      switch(event)
      {
       case 1:  // *** Arrival
          schedule(2, 0.0, customer);
          schedule(1, expntl(Ta), customer);
          break;

        case 2:  // *** Request Server
          if (request(server, customer, 0) == 0)
            schedule(3, expntl(Ts), customer);
          break;
        case 3:  // *** Release server
          release(server, customer);
          break;
      }
    }

    // Output mean length and response time
    printf("===== M/M/1 =====================================================\n");
    printf("=  Lambda                = %f cust/sec \n", 1.0 / Ta);
    printf("=  Mu                    = %f cust/sec \n", 1.0 / Ts);
    printf("=  Utilization           = %f %% \n", 100.0 * U(server));
    printf("=  Mean number in system = %f \n",
      (Lq(server) + U(server)));
    printf("=  Mean response time    = %f sec \n",
      (Lq(server) + U(server)) / U(server));
    printf("================================================================\n");
  }
}
```

Appendix B - M/D/1 simulation model source code

```
//============================================================ file = md1.c =====
//=  A simple M/D/1 queue simulation using SMPL                                  =
//===============================================================================
//----- Include files -----------------------------------------------------------
#include <stdio.h>      // Needed for printf()
#include "smpl.h"       // Needed for SMPL

//===== Main program ============================================================
void main(void)
{
  real Ta;              // Mean interarrival time (seconds)
  real Ts = 1;          // Mean service time (seconds)
  real te = 1.0e7;      // Total simulation time
  int customer = 1;     // Customer id (always '1' for this simulation)
  int event;            // Event (1 = arrival, 2 = request, 3 = completion)
  int server;           // Handle for server facility
  int i;                // Loop counter

  for (i=0; i<17; i++)
  {
    // Set Ta value
    if (i ==  0) Ta = 1.0 / 0.10;
    if (i ==  1) Ta = 1.0 / 0.20;
    if (i ==  2) Ta = 1.0 / 0.30;
    if (i ==  3) Ta = 1.0 / 0.40;
    if (i ==  4) Ta = 1.0 / 0.50;
    if (i ==  5) Ta = 1.0 / 0.60;
    if (i ==  6) Ta = 1.0 / 0.70;
    if (i ==  7) Ta = 1.0 / 0.80;
    if (i ==  8) Ta = 1.0 / 0.90;
    if (i ==  9) Ta = 1.0 / 0.91;
    if (i == 10) Ta = 1.0 / 0.92;
    if (i == 11) Ta = 1.0 / 0.93;
    if (i == 12) Ta = 1.0 / 0.94;
    if (i == 13) Ta = 1.0 / 0.95;
    if (i == 14) Ta = 1.0 / 0.96;
    if (i == 15) Ta = 1.0 / 0.97;
    if (i == 16) Ta = 1.0 / 0.98;

    // Initialize SMPL subsystem
    smpl(0, "M/M/1 Queue");

    // Initialize server facility (single server)
    server=facility("server", 1);

    // Schedule arrival event at time 0 to kick-off simulation
    schedule(1, 0.0, customer);

    // Loop while simulation time is less than te
    while (time() < te)
    {
      // "Cause" the next event on the event list
      cause(&event,&customer);

      // Process the event
      switch(event)
      {
       case 1:  // *** Arrival
          schedule(2, 0.0, customer);
          schedule(1, expntl(Ta), customer);
          break;

        case 2:  // *** Request Server
          if (request(server, customer, 0) == 0)
            schedule(3, Ts, customer);
          break;
        case 3:  // *** Release server
          release(server, customer);
          break;
      }
    }

    // Output mean length and response time
    printf("===== M/D/1 ===================================================\n");
    printf("=  Lambda                 = %f cust/sec \n", 1.0 / Ta);
    printf("=  Mu                     = %f cust/sec \n", 1.0 / Ts);
    printf("=  Utilization            = %f %% \n", 100.0 * U(server));
    printf("=  Mean number in system = %f \n",
      (Lq(server) + U(server)));
    printf("=  Mean response time    = %f sec \n",
      (Lq(server) + U(server)) / U(server));
    printf("===============================================================\n");
  }
}
```