

>>> SOLUTIONS <<<

Welcome to the Midterm Exam for *Simulation*. Read each problem carefully. There are eight required problems (each worth 12 points – you get 4 points for correctly following these instructions). There is also an additional extra credit question worth 10 points. You may have with you a calculator, pencils and/or pens, erasers, blank paper, and one 8.5 x 11 inch “formula sheet”. On this formula sheet you may have anything you want (definitions, formulas, homework answers, old exam answers, etc.) as **handwritten by you in pencil or ink** on both sides of the sheet. Photocopies, scans, or computer generated and/or printed text are not allowed on this sheet. Note to tablet (iPad, etc.) users – you may **not** print-out your handwritten text for the formula sheet. You have 130 minutes for this exam. **Please use a separate sheet of paper for the answer to each question.** Good luck and be sure to show your work!

Problem #1

Each sub-problem worth 3 points.

Answer the following questions regarding the basics of modeling.

a) What are the four possible ways to study a system?

We can 1) do a mathematical analysis, 2) build a simulation model and experiment on it, 3) build a prototype system and experiment on it, and 4) build the actual system and experiment on it.

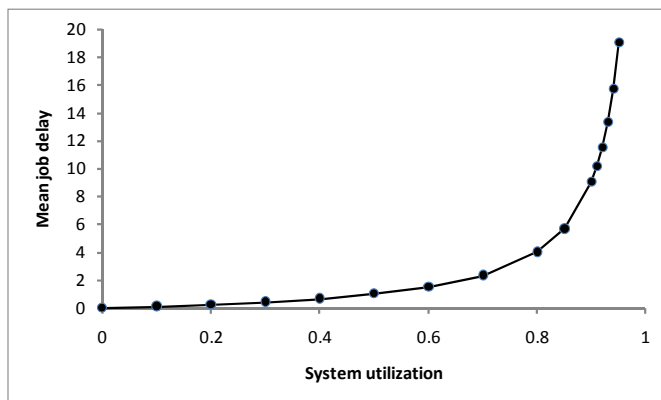
b) What is a model? Give a short formal definition.

“A model is a representation (physical, logical, or functional) that mimics another object under study.” (Molloy 1989).

c) What is a simulation? Give a short formal definition.

“Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a computer, and analyzing the execution output.” (Fishwick, 1995)

d) Sketch a graph of system utilization versus mean job delay or response time for a typical computer or communications system. From a modeling point of view, what is the “big challenge”?



X-axis can be load or utilization. The big challenge is determining the shape of the curve, that is, where the elbow occurs.

Problem #2

Sub-problems (a) thru (d) worth 2 points each, (e) worth 4

Answer the following questions regarding performance and design of experiments.

a) Give four typical performance measures or metrics of interest for a computer or communications system.

Delay, throughput, utilization, and reliability (other possibilities are efficiency and availability).

b) Two factors interact if the effect of one depends on the level of the other. Give an example (for example, for two factors A and B).

	A1	A2
B1	3	5
B2	6	9

In this example, as factor A increases in level for B at level B1 the response variables increases by 2, however for this same increase in A for B at level B2 the response increases by 3. This shows that A and B interact (are not independent) on the effect on the response.

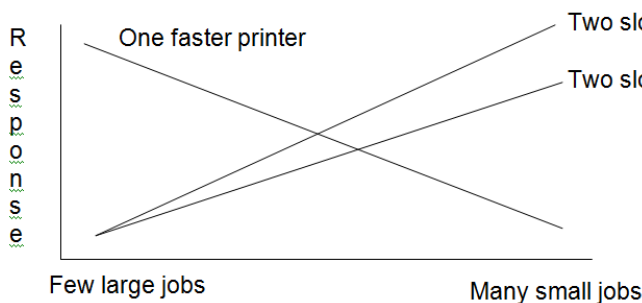
c) What is an experiment and what is the goal for a good design of experiments?

An experiment is a method of investigating causal relationships among variables. The goal of design of experiments is to determine the maximum amount of information with the least effort.

d) Describe full factorial and simple experimental design. What the advantages and disadvantages of each?

Full factorial design = every possible combination of factors and factor levels. Simple design = vary one factor at a time (for all its levels). Factorial has advantage of measuring all factor interactions and disadvantage of large number of experiments. Simple is the exact opposite (interactions may be missed, but small number of experiments).

e) Suppose you could have one 100 page-per-minute printer or two 75 page-per-minute printers, which would be better (assume that individual print jobs cannot be split)? Carefully explain. A graph might be helpful. What is it that a model could help us determine and/or improve?



Note: (1) = perfect job distribution (load balancing)
(2) = less-than-perfect job distribution

Which is better (measured in terms of lower response time to complete a print job) is a function of the workload. Few large jobs would favor one fast printer; many small jobs would favor two medium-speed printers. A model can help us determine the cross-over point and help us design and evaluate the best possible load balancing method.

Problem #3 (Appendix A contains formulas for key distributions)

Each sub-problem worth 3 points.

Answer the following questions regarding probability theory.

a) Give the experimental definition of probability.

$$\Pr[\text{outcome}] = \lim_{n \rightarrow \infty} \frac{\text{Number of observed outcomes}}{n \text{ repetitions of experiment}}$$

b) If $\Pr[A | B] = \Pr[A]$ what can we say about events A and B?

Events A and B are independent.

- c) Given a data message of length 20 bytes (call this mini-Twitter if you will) where the probability of a given byte being in error is p , what is the probability of a message having 0 erroneous bytes, 1 erroneous byte, 2 erroneous bytes, and more than 2 erroneous bytes? Let byte errors be independent and $p = 0.01$.

This is a Binomial distribution problem. So...

$$\Pr[0 \text{ errors}] = \binom{20}{0} (0.01)^0 (0.99)^{20-0} = 0.818$$

$$\Pr[1 \text{ error}] = \binom{20}{1} (0.01)^1 (0.99)^{20-1} = 0.165$$

$$\Pr[2 \text{ errors}] = \binom{20}{2} (0.01)^2 (0.99)^{20-2} = 0.016$$

$$\Pr[\text{more than 2 errors}] = 1 - \Pr[0 \text{ errors}] - \Pr[1 \text{ error}] - \Pr[2 \text{ errors}] = 0.001$$

- d) Define random variable.

A random variable is a function that maps a real number to every possible outcome in the sample space.

Problem #4 (Appendix A contains formulas for key distributions)

Each sub-problem worth 4 points.

Answer the following questions regarding probability theory (specific to distributions and expectation).

- a) What is a probability density function? What is a Probability Distribution Function? What are the key properties of each?

A probability density function $f(x)$ is simply a probability function for a random variable X where

$$f(x) = \Pr[X = x].$$

The properties are $f(x) \geq 0$ and $\sum_{i=0}^{\infty} f(x_i) = 1$.

A Probability Distribution Function $F(x)$ is the sum of the probability density function values,

$$F(x_k) = \sum_{i=0}^k f(x_i) \text{ for discrete and } F(x) = \int_{y=0}^x f(y) dy \text{ for continuous.}$$

The properties are $F(-\infty) = 0$, $F(\infty) = 1$, and $F(x)$ is monotonically increased for x increasing.

- b) Derive the expression for mean (μ) for a uniformly distributed, continuous random variable with minimum value a and maximum value b .

$$\mu = \int_a^b x f(x) dx = \int_a^b x \left(\frac{1}{b-a} \right) dx = \frac{a+b}{2}$$

- c) The time between arrivals in a Poisson distribution are exponentially distributed. Show (or derive) this.

Given that $F(t) = \Pr[T \leq t] = 1 - \Pr[T > t]$ and that $\Pr[t > T]$ is the probability of no events occurring in $[0, T) = \Pr_0[t]$ we have, $F(t) = 1 - \Pr_0[t]$, $\Pr_0[t] = \frac{(\lambda t)^0}{0!} e^{-\lambda t} = e^{-\lambda t}$ which is then $F(t) = 1 - e^{-\lambda t}$ and this is the CDF of an exponential distribution.

Problem #5 Each sub-problem worth 4 points.

Answer the following questions regarding random number generation.

a) What are the desired properties of an RNG?

The four desired properties are 1) return uniformly distributed and independent values, 2) be fast and not require much storage, 3) stream of values should be reproducible, and 4) provision for multiple independent streams.

b) What methods are there for generating random numbers?

Three methods are: 1) Use a physical system known to produce random values (e.g., radioactive decay), 2) use a table of known good random values (e.g., generated from a physical system), and 3) use an algorithm.

c) Here is a C function that returns a double value between 0 and 1, is it a good random number generator? Explain your answer.

```
double randU01(double seed)
{
    const double pi = 3.14159265;
    static double z;
    double val;

    assert((seed >= 0.0) && (seed <= 1.0));
    if (seed > 0) z = seed;

    z = z + (0.01 * log(pi));
    if (z > 1.0) z = 0.0;
    val = z;

    return(val);
}
```

This is a very bad RNG, the autocorrelation will be very close to 1 for short lags. Note that each successive value is larger than the previous value until 1.0 is reached and then the increasing values are repeated. Natural log(pi) is about 1, so the increment is about 0.01 for a cycle length of about 100. Whether or not the distribution of values is uniform is unclear, but it should be at least "close" to uniform.

Problem #6 (Appendix A contains formulas for key distributions) Each sub-problem worth 6 points.

Answer the following questions regarding workload. For both questions you may assume that you have a function `randUnif()` that returns a uniformly distributed random value between 0 and 1 as a double (the function does not need to be seeded).

a) The Rayleigh distribution with shape parameter σ has CDF $F(x) = 1 - e^{-\frac{x^2}{2\sigma^2}}$. Write a C function that returns a Rayleigh distributed random variable with parameter σ .

First we invert $F(x)$...

$$U = 1 - e^{-\frac{x^2}{2\sigma^2}}$$

$$1 - U = e^{-\frac{x^2}{2\sigma^2}}$$

$$\ln(1 - U) = -\frac{x^2}{2\sigma^2}$$

$$x = \sigma \sqrt{-2 \cdot \ln(1 - U)}$$

The C function is then:

```

double rayleigh(double sigma)
{
    double z;          // Uniform random number (0 < z < 1)
    double ray_value; // Computed Rayleigh value to be returned

    // Pull a uniform random number (0 < z <= 1)
    do
    {
        z = randUnif();
    } while (z == 0.0);

    // Compute Rayleigh random variable using inversion method
    ray_value = sigma*sqrt(-2 * log(1 - z));

    return(ray_value);
}

```

- b) Assume you have made measurements on the size of file downloads. You have observed 1 million file downloads and found that 900,000 downloads were exactly 10 KBytes, 100,000 downloads were exactly 100 Kbytes, and 100,000 downloads were exactly 1 MByte. Download sizes are independent of each other. Write a C function that returns a download size based on an empirical distribution from the measurements made.

The C function is:

```

int genEmp()
{
    double z;
    int size;

    z = rand_val(0);
    if (z <= (9.0 / 11.0)) size = 8 * 10 * 1024;
    else if (z <= (10.0 / 11.0)) size = 8 * 100 * 1024;
    else size = 8 * 1024 * 1024;

    return(size);
}

```

Problem #7 Sub-problems (a) thru (c) worth 2 points each, (d) and (e) worth 3 points each.

Answer the following questions about queueing.

- a) What is Kendall notation. Describe it.

A queue is described as A/S/c/k/m where A is the arrival distribution, S is the service distribution, c is the number of servers, k is the capacity of the system in number of customers, and m is the number of customers in the universe. A and S can be "M" for Markov, "D" for deterministic, "G" for general, and others.

- b) State Little's Law.

$L = \lambda W$ for L is the mean number in the system, λ is the arrival rate, and W is the mean wait in the system

- c) For a single-server queue with no loss and known arrival rate (λ) and service rate (μ), how can you determine L_q (number of customers in the buffer), and W_q (average wait in the buffer) given that you have solved for or otherwise determined L (number of customers in the system)?

If you know λ and μ then you have $\rho = \lambda / \mu$ and $T_s = 1 / \mu$. You know L , then from Little's Law $W = L / \lambda$ and then you can solve for $L_q = L - \rho$ and $W_q = W - T_s$.

- d) Given an M/M/1 queue with an arrival rate of 6 jobs per second and a service rate of 10 jobs per second what is the mean number of customers in the system? What is the mean customer delay?

For an M/M/1 queue we know that $L = \frac{\rho}{1-\rho}$. We know that $\rho = \frac{\lambda}{\mu} = \frac{6}{10} = 0.60$. Solving for L we get $L = 1.5$.

- e) Repeat (d) for an M/D/1 queue.

Here we need to use the P-K formula and we know that $C_s = 0$ for deterministic service. The P-K formula is,

$$L = \rho + \frac{\rho^2(1+C_s^2)}{2(1-\rho)}$$

For $\rho = 0.6$ we get $L = 1.05$

Problem #8

Sub-problem (a) worth 6 points, subproblems (b), (c), and (d) are worth 2 points each.

Answer the following questions about discrete event simulation.

- a) Appendix B contains our M/M/1 simulation. Modify this program (you may make the modifications in the appendix itself) to model a single-server system with Poisson arrivals with mean interarrival time of 0.75 seconds per customer and service time equal 1 second per customer when the number of customers in the system is 10 or less and service time 0.5 seconds per customer when the number of customers in the system is greater than 10. When a customer is in service, its service time does not change. Will this system be stable? If yes, what (roughly) will the mean number of customers in the system be (and why)?

See appendix for the modified program (the yellow highlights indicate the changes made).

The system will be stable because the service time when 10 or more customers are in the system is less than the customer interarrival time. The mean queue length will be about 10 because with less than 10 customers in the system the service time is greater than the interarrival time making this a level that the queue will only rarely drop below. At 10 or more the system services customers faster than they arrive.

- b) What are the components of a discrete event simulation (hint: system state is a key component)?

The components are system state, simulation clock (or time), event list, event routine, statistical counters, library routines, report generator, initialization, and main program.

- c) Explain what an event list is, what it contains, and how it is used by event routines. What type of data structure is used to implement an event list?

The event list in a DES is a time-ordered list of events. Each event minimally contains a time (of occurrence) and id. Event routines change the system state based on the next event in the event list, the event routines then generate the next event, which is inserted into the event list in time-order. Typically event lists are implemented as linked lists.

- d) We discussed several options for a simulation language (that is, the method used to program or build a simulation model). What were the options? What are the key parameters to evaluate these options (that is, to evaluate the advantages and disadvantages of each option)?

Key parameters are flexibility, fidelity of results, programmer versus non-programmer, learning curve, model building time, model execution time, graphical output, animation, and cost.

Extra Credit

Basic loop structure is 5 points, key line is 3 points, the rest is 2 points.

Write a Monte Carlo simulation to estimate the area of a 2-dimensional donut (or toroid) with outside radius of 3 inches and inside radius of 2 inches. You may assume that you have a function named `randUnif()` that returns a uniformly distributed random value between 0 and 1 as a double (the function does not need to be seeded).

```
#include <stdio.h>           // Needed for printf()
#include <math.h>           // Needed for pow() and sqrt()

#define NUM_ITER           100000 // Number of iterations to run
#define OUTSIDE_R           3.0 // Outside radius of donut
#define INSIDE_R            2.0 // Inside radius of donut

double randUnit();         // RNG for uniform(0.0, 1.0)

int main()
{
    double    x, y;         // X and Y values
    double    len;         // Length from (0,0) of X,Y point
    double    hitCount;    // Count of hits within the quarter donut
    double    areaEstimate; // Estimated area of the donut
    int       i;           // Loop counter

    // Do for NUM_ITER iterations
    hitCount = 0.0;
    for (i=0; i<NUM_ITER; i++)
    {
        // Throw the dart (dart lands at an X,Y coordinate)
        x = OUTSIDE_R * randUnif();
        y = OUTSIDE_R * randUnif();

        // Compute the length to X,Y using Pythagorean
        len = sqrt(pow(x, 2.0) + pow(y, 2.0));

        // Test if dart is inside quarter donut and increment hitCount if so
        if ((len >= 2.0) && (len <= 3.0)) hitCount++;
    }

    // Estimate area
    areaEstimate = 4.0 * (hitCount / NUM_ITER) * (OUTSIDE_R * OUTSIDE_R);

    // Output results
    printf("----- \n");
    printf("--   *** Results from Monte Carlo donut area estimator ***   -- \n");
    printf("----- \n");
    printf("- Number of iterations = %d          \n", NUM_ITER);
    printf("----- \n");
    printf("- Estimated area = %f sq inches \n", areaEstimate);
    printf("----- \n");
    printf("*** END SIMULATION *** \n");

    return(0);
}
```

Humor

Hmm.... maybe studying is not a good idea after all...

$$\begin{array}{r} \text{No Study} = \text{Fail} \\ \text{Study} = \text{No Fail} \\ + \quad \underline{\hspace{2cm}} \\ \text{No Study} + \text{Study} = \text{Fail} + \text{No Fail} \\ \rightarrow (\cancel{\text{No}} + 1) \text{ Study} = (\cancel{\text{No}} + 1) \text{ Fail} \\ \therefore \text{Study} = \text{Fail} \end{array}$$

From: <http://totalgadha.com/tgtown/mangoman/2010/03/07/126-and-still-counting/>

I hope that everyone did well ☺

Appendix A

A random variable uniformly distributed (continuous) in $a \leq x \leq b$ has probability density function,

$$f(x) = \begin{cases} \frac{1}{b-a} & a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases}$$

A random variable binomially distributed for n trials with probability p of success for each trial has probability mass function,

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad 0 \leq k \leq n.$$

A random variable Poisson distributed for a rate λ of arrivals has probability mass function,

$$f(k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad k = 0, 1, 2, \dots$$

A random variable exponentially distributed for rate λ has density function,

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & t > 0 \\ 0 & t \leq 0 \end{cases}$$

and distribution function,

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & x > 0 \\ 0 & x \leq 0. \end{cases}$$

Appendix B

```
//===== file = mml.c =====
//= A simple "straight C" M/M/1 queue simulation =
//=====
//= Notes: =
//= 1) This program is adapted from Figure 1.6 in Simulating Computer =
//= Systems, Techniques and Tools by M. H. MacDougall (1987). =
//= 2) The values of SIM_TIME, ARR_TIME, and SERV_TIME need to be set. =
//=====
//= Build: gcc mml.c -lm, bcc32 mml.c, cl mml.c =
//=====
//= Execute: mml =
//=====
//= History: KJC (03/09/99) - Genesis =
//= KJC (05/23/09) - Added RNG function (so not to use compiler RNG) =
//= KJC (05/24/09) - Added updated of busy time at end of main loop =
//=====
//----- Include files -----
#include <stdio.h> // Needed for printf()
#include <stdlib.h> // Needed for exit() and rand()
#include <math.h> // Needed for log()

//----- Constants -----
#define SIM_TIME 1.0e6 // Simulation time
#define ARR_TIME 0.75 // Mean time between arrivals
#define SERV_TIME1 1.00 // Mean service time #1
#define SERV_TIME2 0.50 // Mean service time #2

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

//===== Main program =====
int main(void)
{
    double end_time = SIM_TIME; // Total time to simulate
    double Ta = ARR_TIME; // Mean time between arrivals
    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 + exponential(SERV_TIME1);
            }
        }
    }
}
```

```

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)
    {
        if (n < 10)
            t2 = time + exponential(SERV_TIME1);
        else
            t2 = time + exponential(SERV_TIME2);
    }
    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 modified M/M/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 #1 = %f sec \n", SERV_TIME1);
printf("= Mean service time #2 = %f sec \n", SERV_TIME2);
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);
}

//=====
//= Multiplicative LCG for generating uniform(0.0, 1.0) random numbers =
//=====
double rand_val(int seed)
{
    <SNIP SNIP>
}

//=====
//= Function to generate exponentially distributed RVs using inverse method =
//= - Input: x (mean value of distribution) =
//= - Output: Returns with exponential RV =
//=====
double exponential(double x)
{
    <SNIP SNIP>
}

```