Matthew Barnes
0555121

COIS 4470H Modelling and Simulation | Assignment 2

1. **Inventory system:** An automobile dealership uses a weekly periodic inventory review policy. Assume the maximum space for cars is S=80 and the minimum inventory level is s=20. Operation costs are assumed as:

    - Holding cost (C_holding) - $25 per car per week
    - Shortage cost (C_Shortage) - $700 per car per week
    - Set up cost (C_SetUp) - $1000 per order
    - Unit cost (C_Unit) - $8000 each car ordered

(a) Modify the program sis1.c to compute all four components of the total average cost per week.

-- See attached file, sis1.c for all code samples, and methods in question 1.

(b) Use your program to compute and complete the following table (S=80):

s	0	5	10	15	20	25	30	35	40
Average holding cost/week	854.55	917.49	917.49	955.71	1060.03	1144.44	1207.93	1262.14	1277.42
Average shortage cost/week	795.77	374.34	374.34	345.50	172.47	15.88	1.48	0.29	1.19
Average setup cost/week	320.00	340.00	340.00	350.00	390.00	440.00	470.00	500.00	510.00
Sum of the three costs/week	1970.31	1631.84	1631.84	1651.21	1622.51	1600.32	1679.41	1762.43	1788.61

(c) What could be the optimum value for s? Explain.

The optimum value for s would be 25, with the sum of the three costs/week being as low as possible.

(d) Redo (a)-(c). Instead of reading the demands from the input file (sis1.dat), using random-variate generation techniques. Assume demands are uniformly distributed in the same range as the data in file sis.dat.

Max of sis.dat is 48, Min of sis.dat is 17, Max index is 100.

s	0	5	10	15	20	25	30	35	40
Avg Demand	32.93	31.94	33.26	33.79	31.65	34.51	31.83	31.86	31.73
Average holding cost/week	861.94	906.67	929.78	989.97	1056.67	1057.26	1154.95	1260.68	1303.17
Average shortage cost/week	981.83	753.77	497.85	310.65	120.31	59.65	23.11	5.14	1.27
Average setup cost/week	350.00	360.00	380.00	420.00	410.00	470.00	460.00	530.00	560.00
Sum of the three costs/week	2193.77	2020.44	1807.63	1720.62	1586.99	1586.91	1638.06	1795.82	1864.44

The optimal value of s is 25 again, but only by a slim margin. This is due to the random demand being generated.

(e) Redo (a)-(c). Instead of reading the demands from the input file (sis1.dat), using random-variate generation techniques. Assume demands follow Geometric Distribution with the same mean as the data in file sis.dat.

Mean of sis.data is 29.29. P = 1/mean

s	0	5	10	15	20	25	30	35	40
Avg Demand	28.68	24.98	25.80	24.51	25.41	30.71	27.01	32.60	26.34
Average holding cost/week	1009.13	1137.89	1075.83	1129.34	1170.53	1261.06	1294.54	1289.81	1378.64
Average shortage cost/week	1599.68	899.86	695.26	713.07	424.34	1160.04	396.57	1436.69	528.85
Average setup cost/week	270.00	260.00	280.00	280.00	300.00	340.00	360.00	380.00	380.00
Sum of the three costs/week	2878.82	2297.75	2051.09	2122.42	1894.87	2751.10	2051.11	3106.50	2287.49

The optimal value for s is 20, in this case, by a large margin. This is because of the high level of Demand when s = 25, and a more manageable demand when s = 20.

(f) Compare results obtained for different demand distributions.

The demand distributions are not as accurate as I would've hoped. This is due to the introduction of random variables. This test could be run many more times, to make sure that these datum are not outliers.

2. A Random number generator can be developed by combining *two* Linear Congruential Generators using the following algorithm:

The first generator has multiplier a_1 and modulus m_1,
The second generator has a multiplier a_2 and modulus m_2.

Step 1:

Select seed X 1,0 in the range of [1, m 1 -1] for the first generator and seed X 2,0 in the range of [1, m 2 -1] for the second generator. Set j=0.

Step 2: Evaluate each individual generator:

$$X_{1,\, j+1} = a_1 X_{1,\, j} \bmod m_1$$
$$X_{2,\, j+1} = a_2 X_{2,\, j} \bmod m_2$$

Step 3:

$$X_{j+1} = (X_{1,\, j+1} - X_{2,\, j+1}) \bmod m_1$$

Step 4:

$$\text{Return} \qquad R_{j+1} = \begin{cases} \dfrac{X_{j+1}}{m_1}, & \text{if } X_{j+1} > 0, \\ \dfrac{m_1 - 1}{m_1}, & \text{if } X_{j+1} = 0. \end{cases}$$

Step 5:

Set j = j +1 and go to Step 2.

(a) Following this algorithm, develop a Combined Linear Congruential Generator. The seed $X_{1,0}$ and $X_{2,0}$, parameters a_1, m_1 , a_2 , m_2 , and number of random numbers generated should be given by the user at run time.

-- See attached file, a2q3.c for code

(b) Run your program using the input

$$X_{1,0} = 7,\ X_{2,0} = 8,\ a_1 = 11,\ m_1 = 16,\ a_2 = 3,\ m_2 = 32,$$

to generate 100 random variates between 0 and 1.

Every four numbers are repeated.

(c) Apply the Gap Test with the interval (0.2, 0.5) to determine if the random variates generated are independent (α = 5%.).

$s = 24$

Gap Length (i)	$f_e = \delta(1-\delta)^i * s$	f_o
0	7.5	0
1	5.25	0
2	3.675	0
3	2.5725	24
4	1.800750	0
5	1.260525	0
6	0.882367	0
7	0.617657	0
8	0.432360	0
9	0.302652	0

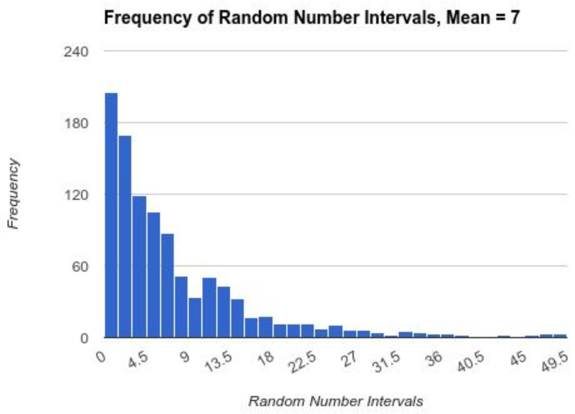
$$x^2 = \sum_{j=1}^{s} \frac{[f_o(j) - f_e(j)]^2}{f_e(j)} = \frac{-7.5^2}{7.5} + \frac{-5.25^2}{5.25} + \frac{-3.675^2}{3.675} + \frac{(24-2.5725)^2}{2.5725} + \frac{-1.8^2}{1.8} + \frac{-1.2^2}{1.2} + \frac{-0.89^2}{0/89} + \frac{-0.617^2}{0.617} + \frac{-0.432^2}{0.432} + \frac{-0.303^2}{0.303}$$

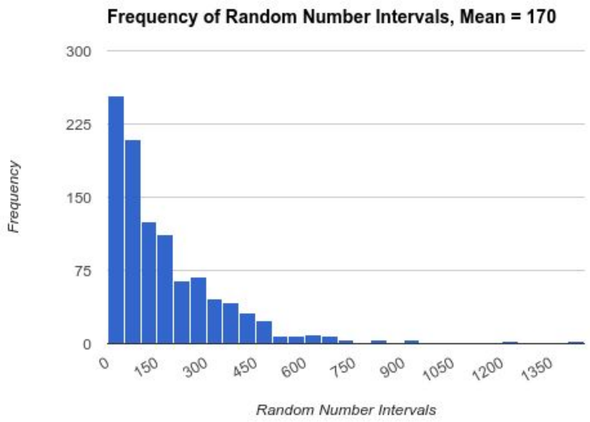
$= 197.44$

$x^2_{0.05,\,23} = 35.172 < 197.44$, Therefore the values are not independent.

3. Apply the formulas discussed in class to develop a random variate generator for exponential distribution. Run the program using μ = 7 and μ = 170 to generate 1000 random variate respectively. Plot histograms for the obtained random variates.

-- See attached file, a2q3.c for code,
a2q3a.csv for mean = 7, and a2q3b.csv for mean = 170





4. Applying the Frequency Test to test the following sequence of numbers for uniformity, using s = 10 subintervals and α = 5%.

0.594, 0.928, 0.515, 0.055, 0.507, 0.351, 0.262, 0.797, 0.788, 0.442, 0.097, 0.798, 0.227, 0.127, 0.474, 0.825, 0.007, 0.182, 0.929, 0.852

20 numbers in total, $n = 20$ $f_e = \frac{n}{s} = \frac{20}{10} = 2$
10 subintervals over (0, 1), $s = 10$

Interval	f_e	f_o	
0-0.1	2	*0.055, 0.097, 0.007*	[3]
0.1-0.2	2	*0.127, 0.182*	[2]
0.2-0.3	2	*0.262, 0.227*	[2]
0.3-0.4	2	*0.351*	[1]
0.4-0.5	2	*0.442, 0.474*	[2]
0.5-0.6	2	*0.594, 0.515, 0.507*	[3]
0.6-0.7	2		[0]
0.7-0.8	2	*0.797, 0.788, 0.798*	[3]
0.8-0.9	2	*0.825, 0.852*	[2]
0.9-1.0	2	*0.928, 0.929*	[2]

$$x^2 = \frac{s}{n}\sum_{j=1}^{s}(f_o(j) - \frac{n}{s})^2 = \frac{10}{20}\sum_{j=1}^{10}(f_o(j) - 2)^2$$

$$= \frac{1}{2} * (1+0+0+1+0+1+4+1+0+0)$$

$$= \frac{8}{2}, \quad x^2 = 4, \ \textit{compared to} \quad x^2_{\alpha,\,(s-1)} = x^2_{0.05,\,9} = 16.919$$

Because x^2 < $x^2_{0.05,\,9}$, the sequence of numbers must be uniformly distributed.

Note:
Give all your answers and discussions in a pdf file named: yourLastName-A2. Submit both the answer file and all programs on Blackboard by 11:59pm on the due date. Code is also listed below.

Sis1.c -- Question 1

```c
/* -------------------------------------------------------------------------
 * This program simulates a simple (s,S) inventory system using demand read
 * from a text file.  Backlogging is permitted and there is no delivery lag.
 * The output statistics are the average demand and order per time interval
 * (they should be equal), the relative frequency of setup and the time
 * averaged held (+) and short (-) inventory levels.
 *
 * NOTE: use 0 <= MINIMUM < MAXIMUM, i.e., 0 <= s < S.
 *
 * Name              : sis1.c  (Simple Inventory System, version 1)
 * Authors           : Steve Park & Dave Geyer
 * Language          : ANSI C
 * Compile with      : gcc sis1.c -lm
 * -------------------------------------------------------------------------
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define FILENAME  "sis1.dat"            /* input data file                */
#define MINIMUM   40                    /* 's' inventory policy parameter */
#define MAXIMUM   80                    /* 'S' inventory policy parameter */
#define sqr(x)    ((x) * (x))

/* ======================== */
   long GetDemand(FILE *fp)
/* ======================== */
{
  long d;
  fscanf(fp, "%ld\n", &d);
  return (d);
}
/* ======================== */
   long GenDemand()
/* ======================== */
{
  long d;
  long max = 49;
  long min = 17;
  d = (rand() % (max - min)) + min;    /* generate demand between min and max of sis1.dat*/
  return (d);
}
```

```
/* ======================== */
   long GeoDemand()
/* ======================== */
{
  long d;                   /*           mean = 29.29;       */
  double p = 1/29.29;       /* i guess this equals 1 / 29.29 */
  double random = ((double) rand() / RAND_MAX);
  d = (fabs((long) (((double) log(1.0 - random) / (double) log(1-p)))));
  return (d);
}
/* ============== */
   int main(void)
/* ============== */
{
  FILE *fp;                                /* input data file         */
  long index     = 0;                      /* time interval index     */
  long inventory = MAXIMUM;                /* current inventory level */
  long demand;                             /* amount of demand        */
  long order;                              /* amount of order         */
  long C_holding = 25;                     /* cost per car per week   */
  long C_shortage = 700;                   /* cost per car per week   */
  long C_setUp = 1000;                     /* cost per order          */
  long C_unit = 8000;                      /* each car ordered        */

  srand(time(NULL));
  /* used for calculating min and max values in sis1.data
    long max = 0;
    long min = 9999;
    long temp = 0;
  */
  struct {                                 /* sum of ...              */
    double setup;                          /*   setup instances       */
    double holding;                        /*   inventory held (+)    */
    double shortage;                       /*   inventory short (-)   */
    double order;                          /*   orders                */
    double demand;                         /*   demands               */
  } sum = { 0.0, 0.0, 0.0, 0.0, 0.0 };

  fp = fopen(FILENAME, "r");
  if (fp == NULL) {
    fprintf(stderr, "Cannot open input file %s\n", FILENAME);
    return (1);
  }
```

```
/* used for calculating min and max values in sis1.dat */
   while (!feof(fp)) {
     temp = GetDemand(fp);
     if (max < temp)
       max = temp;
     if (min > temp)
       min = temp;
   }
   printf("max: %ld\n", max);
   printf("min: %ld\n", min);
   fclose(fp);
   return(0);
 */
 while (index < 100) {               /*!feof(fp)) {*/
   index++;
   if (inventory < MINIMUM) {             /* place an order          */
     order         = MAXIMUM - inventory;
     sum.setup++;
     sum.order    += order;
   }
   else                                   /* no order                 */
     order          = 0;
   inventory      += order;               /* there is no delivery lag */
   demand          = GeoDemand();             /*GetDemand(fp);*/
   sum.demand     += demand;
   if (inventory > demand)
     sum.holding  += (inventory - 0.5 * demand);
   else {
     sum.holding  += sqr(inventory) / (2.0 * demand);
     sum.shortage += sqr(demand - inventory) / (2.0 * demand);
   }
   inventory      -= demand;
 }
 if (inventory < MAXIMUM) {               /* force the final inventory to */
   order           = MAXIMUM - inventory; /* match the initial inventory  */
   sum.setup++;
   sum.order      += order;
   inventory      += order;
 }
 printf("\nfor %ld time intervals ", index);
 printf("with an average demand of %6.2f\n", sum.demand / index);
 printf("and policy parameters (s, S) = (%d, %d)\n\n", MINIMUM, MAXIMUM);
 printf("   average order ............ = %6.2f\n", sum.order / index);
 printf("   setup frequency .......... = %6.2f\n", sum.setup / index);
 printf("   setup cost ............... = %6.2f\n", sum.setup * C_setUp / index);
 printf("   average holding level .... = %6.2f\n", sum.holding / index);
 printf("   average holding cost ..... = %6.2f\n", sum.holding * C_holding / index);
 printf("   average shortage level ... = %6.2f\n", sum.shortage / index);
 printf("   average shortage cost .... = %6.2f\n", sum.shortage * C_shortage / index);
```

```
  printf("   sum of the three costs ... = %6.2f\n", (sum.shortage * C_shortage / index) +
(sum.holding * C_holding / index) + (sum.setup * C_setUp / index));
  fclose(fp);
  return (0);
}
```

A2Q2.C -- Question 2

```c
/* -------------------------------------------------------------------------
 * Demonstrates how a random number generator can be developed by
 * combining two Linear Congruential Generators using the following
 * algorithm.
 *
 * The first generator has multiplier a_1, and modulus m_1,
 * The second generator has multiplier a_2, and modulus m_2,
 *
 * NOTE: use 0 <= MINIMUM < MAXIMUM, i.e., 0 <= s < S.
 *
 * Name              : a2q2.c
 * Author            : Matthew Barnes
 * Language          : ANSI C
 * Compile with      : gcc a2q2.c -lm
 * -------------------------------------------------------------------------
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define MAX_LEN 100
/* ============== */
   int main(void)
/* ============== */
{
  int m_1 = 16;
  int m_2 = 32;
  long a_1 = 11;
  long a_2 = 3;
  long x_1 = 7;                    /* must be [1, (m_1 - 1)] */
  long x_2 = 8;                    /* must be [1, (m_2 - 1)] */
  double temp;
  double max_range = 0.5;
  double min_range = 0.2;
  int gap_size = 0;
  long num_rand = 100;              /* number of random numbers to be generated */
  long index = 0;
  double rand_store[MAX_LEN];
  int gap_store[MAX_LEN];
  int start_gap_flag = 0;
  int total_samples = 0;

  for (int j = 0; j < MAX_LEN; j++) {
    gap_store[j] = 0;
  }
```

```
printf ("Enter a seed for m_1: ");
scanf ("%d", &m_1);
printf ("Enter a seed for m_2: ");
scanf ("%d", &m_2);
printf ("Enter a value for a_1: ");
scanf ("%ld", &a_1);
printf ("Enter a value for a_2: ");
scanf ("%ld", &a_2);
printf ("Enter a value for x_1: ");
scanf ("%ld", &x_1);
printf ("Enter a value for x_2: ");
scanf ("%ld", &x_2);
printf ("Enter how many numbers to generate: ");
scanf ("%ld", &num_rand);
printf ("You entered: %d\n", m_1);
printf ("You entered: %d\n", m_2);

while (index < num_rand) {
  index++;
  x_1 = (a_1 * x_1) % m_1;
  x_2 = (a_2 * x_2) % m_2;
  temp = (fabs((x_1 - x_2) % m_1));
  if (temp > 0) {
    temp = temp / m_1;
  } else {
    temp = (m_1 - 1) / m_1;
  }
  /* formatting output */
  if (index % 5 == 0) {
    printf("%.3ld  Generated random number: %6.4f\n", index, temp);
    printf("%.3ld  x_1: %ld,       x_2: %ld\n", index, x_1, x_2);
    if ( min_range < temp && temp < max_range ) {
      total_samples++;
      if (start_gap_flag) {
        gap_store[gap_size] = gap_store[gap_size] + 1;
      }
      //printf("gap_store[%d]: %d", gap_size, gap_store[gap_size]);
      gap_size = -1;
      start_gap_flag = 1;
      //printf("\n==%.3ld==\n", index);
    }
  } else {
    printf("     Generated random number: %6.4f\n", temp);
    printf("     x_1: %ld,       x_2: %ld\n", x_1, x_2);
if ( min_range < temp && temp < max_range ) {
    total_samples++;
    if (start_gap_flag) {
     gap_store[gap_size] = gap_store[gap_size] + 1;
    }
    //printf("gap_store[%d]: %d", gap_size, gap_store[gap_size]);
    gap_size = -1;
    start_gap_flag = 1;
    //printf("\n==%.3ld==\n", index);
   }
  }
  rand_store[index] = temp;
  gap_size++;
 }
 printf("\n%ld numbers have been generated\n\n", index);
 for(int i = 0; i < (sizeof(gap_store)/sizeof(gap_store[0])); i++) {
  if (gap_store[i] != 0) {
   printf("\n Gap Length of %d, observed count is %d\n", i, gap_store[i] );
   printf(" Gap Length of %d, expected count is %6.6f\n\n", i, ((max_range - min_range) *
pow((1 - (max_range - min_range)), i) * total_samples));
  }
  printf(" Gap Length of %d, expected count is %6.6f\n", i, ((max_range - min_range) *
pow((1 - (max_range - min_range)), i) * total_samples));
  //printf("\n%d ", gap_store[i]);
 }
 return (0);
}
```

A2Q3.c -- Question 3

```
/* --------------------------------------------------------------------------
 * A random variate generator for exponential distribution.
 * MEW = 7, MEW = 170.
 * generate 1000 random variates for each.
 * Histograms in other file.
 * Name              : a2q3.c
 * Author            : Matthew Barnes
 * Language          : ANSI C
 * Compile with      : gcc a2q3.c -lm
 * --------------------------------------------------------------------------
*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define MAX_LEN 1000
/* ============== */
   int main(void)
/* ============== */
{
  long rand_count = 1000;
  double mean = 7;
  double random;
  double rand_num;
  double rand_store[MAX_LEN];
  FILE *fp;

  srand(time(NULL));

  fp = fopen("a2q3.csv", "w");
  if (fp == NULL) {
    printf("Couldn't open file\n");
    return (1);
  }

  for (int i = 0; i < rand_count; i++)
  {
      rand_num = (-1) * (mean * log(1 - ((double) rand() / RAND_MAX)));
      fprintf(fp, "%6.6f\n", rand_num);
      rand_store[i] = rand_num;
  }
  for (int j = 0; j < (sizeof(rand_store)/sizeof(rand_store[0])); j++) {
    printf("  %d:    %6.6f\n", j, rand_store[j]);
  }
  fclose(fp);
  return (0);
}
```