Introduction to simulation

Ivo Adan and Jacques Resing

Department of Mathematics and Computer Science
TU Eindhoven

July 4, 2007

Organisation:

- 2 lectures
- studeerwijzer available (with notes, slides, programs, assignments)
- examination of simulation consists of 1 take home assignment
- assignments are done in groups of 2

Topics:

- Modeling of *discrete-event systems*
- Programming
- Ouput analysis

Discrete-event systems:
State changes at (random) discrete points in time

Examples:

- Manufacturing systems
  Completion times of jobs at machines, machine break-downs
- Inventory systems
  Arrival times of customer demand, replenishments
- Communication systems
  Arrival times of messages at communication links

Basic approach to modeling

- Identify the issues to be addressed
- Learn about the system
- Choose a modeling approach
- Develop and test the model
- Verify and validate the model
- Experiment with the model
- Present the results

Types of models

- Physical models
- Simulation models
- Analytical models

But why model?

- Understanding
- Improvement
- Optimization
- Decision making

Issues in developing a model

- Complexity versus Simplicity
- Flexibility
- Data requirements
- Transparency

Analytical and simulation capability:
Effective modeling requires both!

Programming tools:

- General purpose languages (C, Java, ...);
  for downloads, see:
  - Cygwin (a UNIX environment for Windows)
  - DJGPP compiler (stand alone C compiler for DOS)
  - GNUplot (for plotting)
- Simulation language χ developed by the Systems Engineering group
- Simulation system Arena

To start with:
Simulation and simple probabilistic problems

Simulation is a perfect tool to develop and sharpen your *intuition* for probabilitic models (see also Tijms' book *Understanding probability*)

In no time probabilistic properties can be illustrated by a simulation experiment and the results can be shown in graphs or tables!

Problems

- coin tossing
- (nearly) birth day problem
- lottery
- breaking matches

Coin tossing

Two players A and B throw a *fair* coin N times.
If Head, then A gets 1 point; otherwise B.

- What happens to the absolute difference in points as N increases?
- What is the probability that one of the players is leading between 50% and 55% of the time? Or more than 95% of the time?
- In case of 20 trials, say, what is the probability of 5 Heads in a row?

Birthday problem

Consider a group of N randomly chosen persons.
What is the probability that at least 2 persons have the same birthday?

Nearly birthday problem

What is the probability that at least 2 persons have their birthday within r days of each other?

Lottery

Each week a very popular lottery in Andorra prints 10^4 tickets. Each tickets has two 4-digit numbers on it, one visible and the other covered. The numbers are randomly distributed over the tickets. If someone, after uncovering the hidden number, finds two identical numbers, he wins a large amount of money.

- What is the average number of winners per week?
- What is the probability of at least one winner?

The same lottery prints 10^7 tickets in Spain. What about the answers to the questions above?

Breaking matches

A match of 1 cm is broken at 2 random points.

- What is the mean length of the smallest part, and the largest part?
- What is the mean value of the quotient of the length of the smallest and largest part?

Generating random numbers

Iterative procedure:

Start with z_0 (seed)
For $n = 1, 2, \ldots$

$$z_n = f(z_{n-1})$$

f is the *pseudo-random generator*

In practice, the following function is often used

$$z_n = az_{n-1}(\text{modulo } m)$$

(with $a = 630360016$, $m = 2^{31} - 1$)

Then $u_n = z_n/m$ is random on $(0, 1)$

Sampling from other distributions

Let U be uniform on $(0, 1)$
Then sampling from

- interval (a, b): $a + (b - a)U$
- integers $1, \ldots, M$: $1 + \lfloor MU \rfloor$
- discrete distribution:
  let $P(X = x_i) = p_i$, $i = 1, \ldots, M$
  if $U \in [\sum_{j=1}^{i-1} p_j, \sum_{j=1}^{i} p_j)$, then $X = x_i$

Array method for sampling from a discrete distribution

Suppose $p_i = k_i/100$, $i = 1, \ldots, M$,
where k_i's are integers with $0 \le k_i \le 100$

Construct array $A[i]$, $i = 1, \ldots, 100$ as follows:
set $A[i] = x_1$ for $i = 1, \ldots, k_1$
set $A[i] = x_2$ for $i = k_1 + 1, \ldots, k_1 + k_2$, etc.

Then, first, sample random index I from $1, \ldots, 100$:
$I = 1 + \lfloor 100U \rfloor$ and set $X = A[I]$

Inverse transform method for sampling from a continuous distribution

Let the random variable X have a continuous and increasing distribution function F. Denote the inverse of F by F^{-1}. Then X can be generated as follows:

- Generate U from $U(0,1)$;
- Return $X = F^{-1}(U)$.

If F is not continuous or increasing, then we have to use the *generalized* inverse function

$$F^{-1}(u) = \min\{x : F(x) \geq u\}.$$

Examples:

- $X = a + (b - a)U$ is uniform on (a, b);
- $X = -\log(U)/\lambda$ is exponential with parameter λ;
- $X = (-\log(U))^{1/a}/\lambda$ is Weibull, par. a and λ.

Unfortunately, for many distribution functions we do not have an easy-to-use (closed-form) expression for the inverse of F.

Simulation of Coin tossing

```
n = 0
points_A = 0
points_B = 0

while n < N do
  if random < 0.5
  then points_A = points_A + 1
  else points_B = points_B + 1
  n = n + 1
  print points_A - points_B
end
```

C-code of simulating coin tossing

```
#include <stdlib.h>

 /* globals */
long    seed;          /* seed of random generator */

main()
{
    int     n,         /* number of current trial */
            N,         /* total number of trials */
            points_A, /* number of points of player A */
            points_B; /* number of points of player B */

    seed = 1;
    srand48(seed);    /* initialization random generator */

    printf("Number of trials: ");
    scanf("%d", &N);  /* get input */

    n = 0;            /* initialization */
    points_A = 0;
    points_B = 0;

    while (n < N) {
        if (drand48() < 0.5)          /* coin tossing */
            points_A = points_A + 1; /* it is Head */
        else
            points_B = points_B + 1;
        n = n + 1;
        printf("%d\n", points_A - points_B);
    }
}
```

Java-code of simulating coin tossing

```
import java.util.Random;

public class CoinToss {

  /* total number of trials */
  protected int N;
  /* probability on head */
  protected double p;
  /* random number generator */
  protected Random rand;

  /**
   * Constructs a CoinToss object.
   * The probability on head is taken 0.5.
   * @param N the number of realisations
   */

  public CoinToss(int N) {
    this(N, 0.5);
  }

  /**
   * Constructs a CoinToss object.
   * The probability on head is 0.5.
   * @param N the number of realisations
   * @param seed the random seed for the random number generator
   */

  public CoinToss(int N, long seed) {
    this(N, 0.5, seed);
  }
```

```
/**
 * Constructs a CoinToss object.
 * The probability on head is 0.5.
 * @param N the number of realisations
 * @param p the probability on head.
 */

public CoinToss(int N, double p) {
  this.N = N;
  this.p = p;
  this.rand = new Random();
}

/**
 * Constructs a CoinToss object.
 * @param N the number of realisations
 * @param p the probability on head.
 * @param seed the random seed for the random number generator
 */

public CoinToss(int N, double p, long seed) {
  this.N = N;
  this.p = p;
  this.rand = new Random(seed);
}
```

```
  /**
   * Tosses a coin <i>N</i> times, and prints
   * <i>number of heads - number of tails</i>
   */

  public void printRealisation() {
    int n = 0;        /* initialization */
    int points_A = 0;
    int points_B = 0;

    while (n < N) {
      if (rand.nextDouble() < p)             /* coin tossing */
        points_A = points_A + 1;        /* it is Head */
      else
        points_B = points_B + 1;
      n = n + 1;
      System.out.println("" + (points_A - points_B));
    }

  }

  public static void main(String[] arg) {
    CoinToss c = new CoinToss(1000, 0.5);
    c.printRealisation();
  }

}
```

Oscilations become bigger and bigger (grow as $\sqrt{N}$)

Fraction of time one of the players is leading

Let $P(\alpha, \beta)$ be the probability that one of the players is leading between $100\alpha\%$ and $100\beta\%$ of the time

To determine $P(\alpha, \beta)$ do the experiment "Throw N times with a coin" many times; an experiment is successful if one of the players is leading between $100\alpha\%$ and $100\beta\%$ of the time

Then

$$P(\alpha, \beta) \approx \frac{\text{number of successful experiments}}{\text{total number of experiments}}$$

Simulation of M experiments

```
success = 0

for run = 1 to M do
  coin_tossing
  if alpha < time_A / N < beta
  or alpha < time_B / N < beta
  then success = success + 1
end

print success / M
```

Simulation of coin tossing

```
n = 0
points_A = 0;
points_B = 0;
time_A = 0

while n < N do
  if random < 0.5
  then points_A = points_A + 1
  else points_B = points_B + 1
  if points_A - points_B >= 0
  then time_A = time_A + 1
  n = n + 1
end

time_B = N - time_A
```

Results for $M = 10^3$, $N = 10^4$ and seed = 1

(α, β)	$P(\alpha, \beta)$
(0.50,0.55)	0.047
(0.50,0.60)	0.104
(0.90,1.00)	0.426
(0.95,1.00)	0.288
(0.98,1.00)	0.178

Successive Heads

Let $P(k)$ be the probability of at least k succesive Heads in case of 20 trials

To determine $P(k)$ do the experiment “Throw 20 times with a coin” many times; an experiment is successful is a row of at least k Heads appears

Then

$$P(k) \approx \frac{\text{number of successful experiments}}{\text{total number of experiments}}$$

Simulation of M experiments

```
success = 0

for run = 1 to M do
  coin_tossing
  if k_row = TRUE
  then success = success + 1
end

print success / M
```

Simulation of coin tossing

```
n = 0
nr_Heads = 0;
k_row = FALSE

while n < 20 and not k_row do
  if random < 0.5
  then nr_Heads = nr_Heads + 1
  else nr_Heads = 0
  if nr_Heads >= k
  then k_row = TRUE
  n = n + 1
end
```

Results for $M = 10^3$ and seed = 1

k	$P(k)$
1	1.000
2	0.984
3	0.761
4	0.455
5	0.251
6	0.124
7	0.049

Birthday problem

Let $P(N)$ be the probability that at least two persons have the same birthday in a group of size N

To determine $P(N)$ do the experiment "Take a group of N randomly chosen persons and get their birthdays" many times; an experiment is successful is at least two persons have the same birthday

Then

$$P(N) \approx \frac{\text{number of successful experiments}}{\text{total number of experiments}}$$

Simulation of birthday problem

```
success = 0

for run = 1 to M do
  take_group of size N
  if same_birthday = TRUE
  then success = success + 1
end

print success / M
```

Simulation of taking random group

```
n = 0
for i = 1 to 365 do birthday[i] = FALSE
same_birthday = FALSE

while n < N and not same_birthday do
  new = 1 + trunc(random * 365)
  if birthday[new] = TRUE
  then same_birthday = TRUE
  else birthday[new] = TRUE
  n = n + 1
end
```

Results for $M = 10^3$ and seed = 1

N	$P(N)$
10	0.126
15	0.269
20	0.422
25	0.572
30	0.693
40	0.893
50	0.974

Simulation of the nearly birthday problem

```
n = 0
for i = 1 to 365 do birthday[i] = FALSE
nearly_same_birthday = FALSE

while n < N and not nearly_same_birthday do
  new = 1 + trunc(random * 365)
  for i = new - r to new + r do
    if birthday[i] = TRUE
    then nearly_same_birthday = TRUE
  birthday[new] = TRUE
  n = n + 1
end
```

Results for $M = 10^3$ and seed = 1

N	r	$P(N)$
10	0	0.126
	1	0.324
	2	0.476
	7	0.878
20	0	0.422
	1	0.811
30	0	0.693
	1	0.971

Lottery problem

Tickets are numbered $1, \ldots, N$

To print covered numbers, first generate a *random permutation of* $1, \ldots, N$ and then print first number on first ticket, second one on second ticket, and so on; hence:

Number of winners is equal to number of numbers that stay on their position after this permutation

Average number of winners in a lottery

$$\approx \frac{\text{total number of winners in all experiments}}{\text{total number of experiments}}$$

Generating a random permutation of $1, \ldots, N$

1. Initialize $t = N$ and $A[i] = i$ for $i = 1, \ldots, N$;
2. Generate a random number u between 0 and 1;
3. Set $k = 1 + \lfloor tu \rfloor$; swap values of $A[k]$ and $A[t]$;
4. Set $t = t - 1$;
   If $t > 1$, then return to step 2;
   otherwise stop and $A[1], \ldots, A[N]$ yields a permutation.

Complexity is $O(N)$

Results for $M = 10^3$, $N = 10^4$ and seed = 1

Average numbers of winners is 1.005

k	$P(k \text{ winners})$
0	0.351
1	0.385
2	0.185
3	0.066
4	0.013

Results for $M = 10^3$, $N = 10^5$ and seed = 1

Average numbers of winners is 1.047

k	$P(k \text{ winners})$
0	0.350
1	0.380
2	0.172
3	0.074
4	0.020

Observe that number of winners is equal to number of *loops of length* 1

Further, consider $1, \ldots, s$;
then length of loop generated by 1 is *uniform on* $1, \ldots, s$

Loops of a random permutation of $1, \ldots, N$

1. Initialize $t = N$ and $W = 0$;
2. Generate a random number u between 0 and 1;
3. Set $l = 1 + \lfloor tu \rfloor$; if $l = 1$, then $W = W + 1$;
4. Set $t = t - l$; if $t \geq 1$, return to step 2;
   otherwise stop and W yields number of loops of length 1.

Complexity is $O(\log N)$

Results for $M = 10^5$, $N = 10^7$ and seed = 1

Average numbers of winners is 1.001

k	$P(k \text{ winners})$
0	0.367
1	0.370
2	0.183
3	0.061
4	0.016

Breaking matches in N parts

Generate N positions (cracks) $u_1, \ldots, u_N$ in $(0, 1)$, and order these positions in increasing order yielding $u_{(1)}, \ldots, u_{(N)}$ (order statistics); then lengths of parts (or spacings)

$$
\begin{aligned}
d_1 &= u_{(1)}, \\
d_2 &= u_{(2)} - u_{(1)}, \\
&\ldots, \\
d_N &= u_{(N)} - u_{(N-1)}, \\
d_{N+1} &= 1 - u_{(N)}
\end{aligned}
$$

But ordering is not efficient!

Uniform spacings

Let $D_1, \ldots, D_{N+1}$ be uniform spacings on $(0, 1)$;

Let $X_1, \ldots, X_{N+1}$ be exponentials with mean 1, and set

$$S_N = \sum_{i=1}^{N+1} X_i;$$

Then

$$(D_1, \ldots, D_{N+1}) \text{ and } (X_1/S_{N+1}, \ldots, X_{N+1}/S_{N+1})$$

have exactly the same distribution; in other words, uniform spacings are normalized exponentials

Sampling from the exponential distribution

If U is uniform on $(0,1)$, then the random variable

$$X = -\log(1-U)/\mu$$

is exponential with parameter μ; since

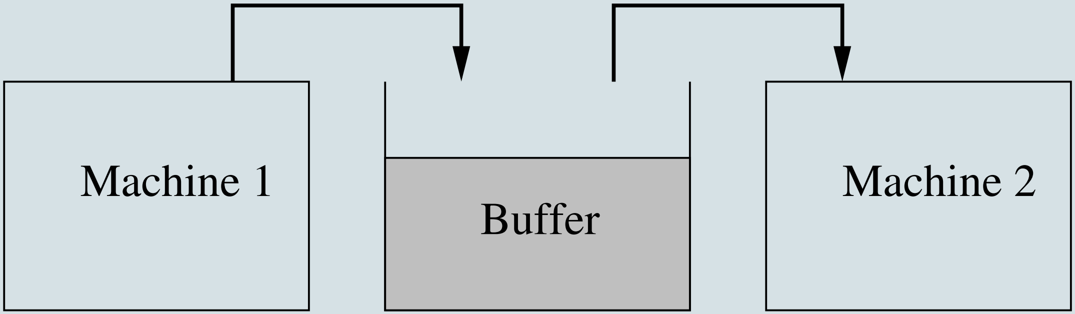
$$\begin{aligned} P(X \leq x) &= P(-\log(1-U)/\mu \leq x) \\ &= P(\log(1-U) \geq -\mu x) \\ &= P(1-U \geq e^{-\mu x}) \\ &= P(U \leq 1-e^{-\mu x}) \\ &= 1-e^{-\mu x} \end{aligned}$$

Results for $M = 10^4$ and seed = 1

cracks	E(small)	E(large)	E(small/large)
1	0.248	0.752	0.383
2	0.109	0.614	0.211
3	0.062	0.523	0.139
4	0.039	0.459	0.101
5	0.027	0.411	0.077

Example of a discrete-event system: Two-machine production line

Machine 1 produces material and puts it into the buffer,
machine 2 takes the material out the buffer.
The material is a *fluid* flowing in and out the buffer.

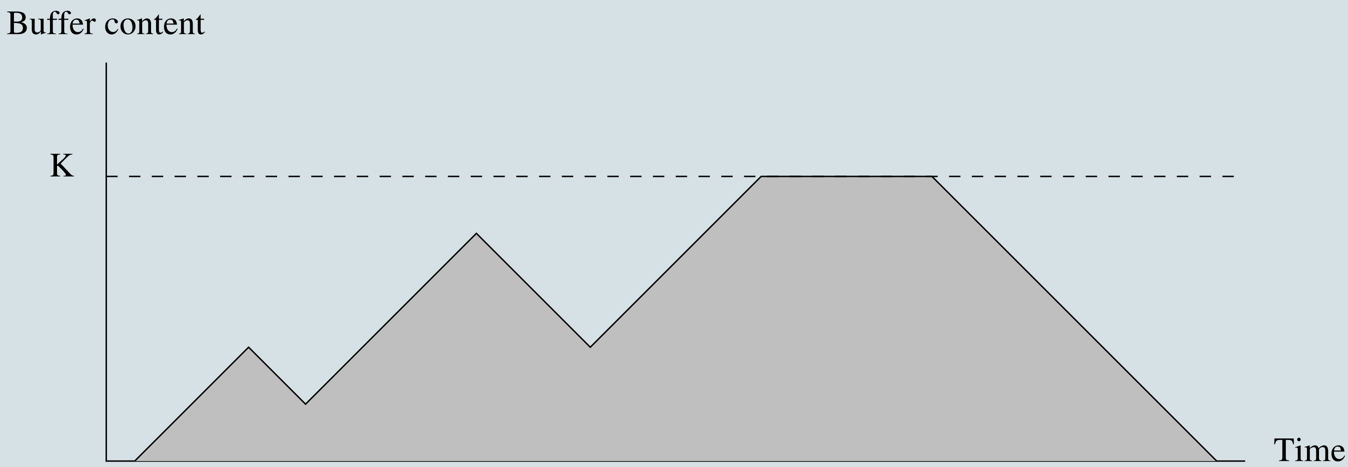


Fluid flow model

The production rate of machine i is r_i ($i = 1, 2$).
We assume that $r_1 > r_2$ (otherwise no buffer needed).
Machine 2 is perfect (never fails), but machine 1 is subject to breakdowns; the mean up time is $E(U)$ and the mean down time is $E(D)$.
The size of the buffer is K.
When the buffer is full, the production rate of machine 1 slows down to r_2.

Questions:

- What is the throughput (average production rate) TH?
- How does the throughput depend on the buffer size K?



Time path realization of the buffer content

Applications

Chemical processes
Machine 1 produces a standard substance that is used by machine 2 for the production of a range of products. When machine 2 changes from one product to another it needs to be cleaned. Switching off machine 1 is costly, so the buffer allows machine 1 to continu production.
How large should the buffer be?

Of course, in this application, machine 1 instead of 2 is perfect.

Data communication

In communication networks standard packages called cells are sent from one switch to another. In a switch incoming packages are 'multiplexed' on one outgoing line. If temporarily the number of incoming cells exceeds the capacity of the outgoing line, the excess inflow is buffered. Once the buffer is full, an incoming cell will be lost.

How large should the buffer be such that the loss probability is sufficiently small?

Production of discrete items

Items are produced on two consecutive workstations. The first one is a robot, the second one is manned and somewhat slower. Unfortunately the robot is not fully reliable. Occasionally it breaks down. A buffer enables the manned station to continu while the robot is being repaired.

What is a good size of the buffer?

Zero buffer

Fraction of time machine 1 is working is equal to $E(U)/(E(U)+E(D))$; hence

$$TH = r_2 \cdot \frac{E(U)}{E(U)+E(D)}$$

Infinite buffer

Average production rate of machine 1 is equal to

$$r_1 \cdot \frac{E(U)}{E(U)+E(D)}$$

Hence

$$TH = \min\left\{r_1 \cdot \frac{E(U)}{E(U)+E(D)}, r_2\right\}$$

Finite buffer

Assume *exponential* up and down times;
let $1/\lambda = E(U)$ and $1/\mu = E(D)$.

The system can be described by a continuous-time Markov process with states (i, x) where i is the state of the first machine ($i = 1$ means that machine 1 is up, $i = 0$ means that it is down) and x is the buffer content ($0 \leq x \leq K$).

Define $F(i, x)$ as the (steady state) probability that machine 1 is in state i and that the buffer content is less or equal to x; then

$$TH = r_2 \cdot (1 - F(0, 0))$$

Balance equations

$$\mu F(0,x) = \lambda F(1,x) + r_2 F'(0,x)$$
$$\lambda F(1,x) + (r_1 - r_2) F'(1,x) = \mu F(0,x)$$

or in vector-matrix notation

$$F'(x) = AF(x)$$

where

$$F(x) = \begin{pmatrix} F(0,x) \\ F(1,x) \end{pmatrix}$$

$$A = \begin{pmatrix} \mu/r_2 & -\lambda/r_2 \\ \mu/(r_1 - r_2) & \lambda/(r_1 - r_2) \end{pmatrix}$$

The solution is given by

$$F(x) = C_1 v_1 e^{\sigma_1 x} + C_2 v_2 e^{\sigma_2 x}$$

where σ_1 and σ_2 are the eigenvalues of A, and v_1 and v_2 are the correspon-ding eigenvectors. Here

$$\sigma_1 = 0, \qquad \sigma_2 = \frac{\mu}{r_2} - \frac{\lambda}{r_1 - r_2}$$

$$v_1 = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}, \qquad v_2 = \begin{pmatrix} r_1 - r_2 \\ r_2 \end{pmatrix}$$

The coefficients C_1 and C_2 follow from the *boundary conditions*

$$F(1,0)=0, \qquad F(0,K)=\frac{\lambda}{\lambda+\mu}$$

yielding

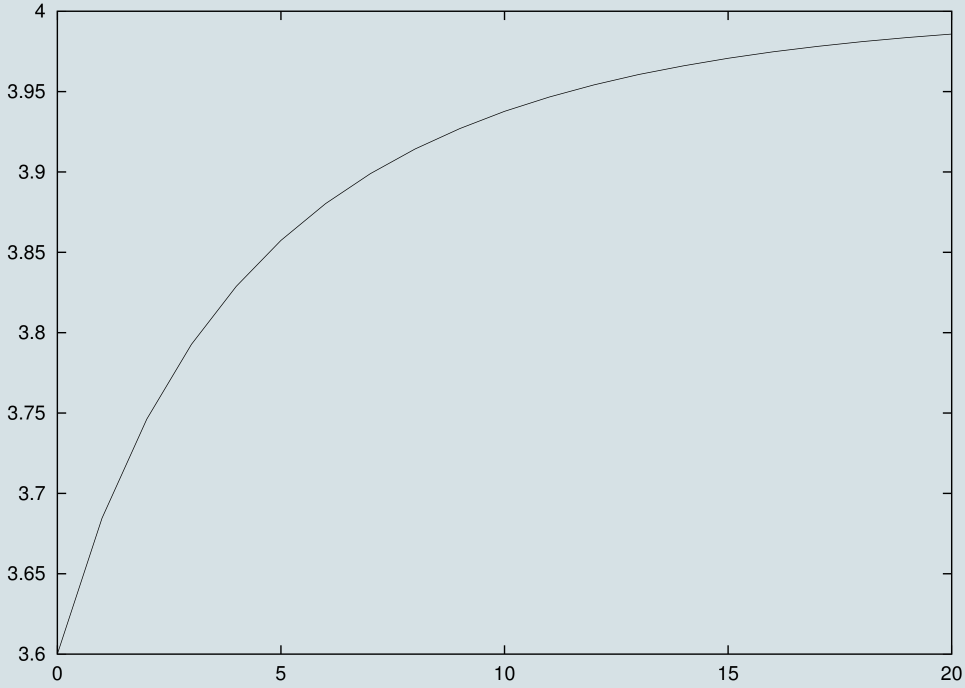
$$C_1 = r_2 \cdot \frac{\lambda}{\lambda+\mu} \cdot \left(\lambda r_2 - \mu(r_1-r_2)e^{\sigma_2 K}\right)^{-1}$$

$$C_2 = -\mu \cdot \frac{\lambda}{\lambda+\mu} \cdot \left(\lambda r_2 - \mu(r_1-r_2)e^{\sigma_2 K}\right)^{-1}$$

Hence

$$TH = r_2 \cdot \frac{\mu}{\lambda+\mu} \cdot \frac{\lambda r_1 - (\lambda+\mu)(r_1-r_2)e^{\sigma_2 K}}{\lambda r_2 - \mu(r_1-r_2)e^{\sigma_2 K}}$$

Example: $\lambda = 1/9$, $\mu = 1$, $r_1 = 5$ and $r_2 = 4$



The throughput as a function of the buffer size

We assumed exponentially distributed up and down times;
what about other (general) distributions?

You may use phase-type distributions;
then a Markov process description is still feasible, but the analysis becomes (much) more complicated.

Let us develop a simulation model!

Simulation model

System behavior only significantly changes when machine 1 breaks down or when it has been repaired. In the simulation we jump from one event to another, and calculate the buffer content at these moments (in between the behavior of the buffer content is known). Based on the information obtained we can estimate the throughput.

Initialization

```
t = 0 {time}
b = 0 {buffercontent at time t
       we assume that at t=0 the buffer is empty
       and that the machine has just been repaired}
empty = 0 {total time upto time t
           that buffer is empty}
```

Main program

```
while (t < runlength)
do
    u = up_time
    t = t + u
    b = min(b+u*(r1-r2), K)
    d = down_time
    t = t + d
    if (b - d*r2 < 0)
    then empty = empty + d - b/r2
    b = max(b-d*r2, 0)
end
```

Output

```
TH = r2 * (1 - empty/t)
```

Questions:

- How do we obtain appropriate input for the simulation model?
- How accurate is the outcome of a simulation experiment?
- What is a good choice for the runlength of a simulation experiment?
- What is the effect of the initial conditions on the outcome of a simulation experiment?

Input of a simulation

Specifying distributions of random variables (e.g., interarrival times, processing times) and assigning parameter values can be based on:

- Historical numerical data
- Expert opinion

In practice, there is sometimes real data available, but often the only information of random variables that is available is their mean and standard deviation.

Empirical data can be used to:

- construct empirical distribution functions and generate samples from them during the simulation;
- fit theoretical distributions and then generate samples from the fitted distributions.

Fitting a distribution

Methods to determine the parameters of a distribution:

- Maximum likelihood estimation
- Moment fitting

Maximum likelihood estimation

Let $f(x;\theta)$ denote the probability density function with unknown parameter (vector) θ.
Let $X = (X_1, \ldots, X_n)$ denote a vector of *i.i.d.* observations from f.
Then

$$L(\theta, X) = \prod_{i=1}^{n} f(X_i, \theta)$$

is the *likelihood function* and $\hat{\theta}$ satisfying

$$L(\hat{\theta}, X) = \sup_{\theta} L(\theta, X)$$

is the *maximum likelihood estimator* of θ.

Examples:

- **Exponential distribution**

$$f(x,\mu) = \mu e^{-\mu x}$$

  Then

$$\frac{1}{\hat{\mu}} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- **Uniform** (a,b)

$$f(x,a,b) = \frac{1}{b-a}$$

  Then

$$\hat{a} = \min X_i, \qquad \hat{b} = \max X_i.$$

But for many distributions $\hat{\theta}$ has to be calculated numerically.

Moment fitting

Obtain an approximating distribution by fitting a *phase-type distribution* on the mean, $E(X)$, and the coefficient of variation,

$$c_X = \frac{\sigma_X}{E(X)},$$

of a given positive random variable X, by using the following simple approach.

Coefficient of variation less than 1

If $0 < c_X < 1$, then fit an $E_{k-1,k}$ distribution as follows. If

$$\frac{1}{k} \leq c_X^2 \leq \frac{1}{k-1},$$

for certain $k = 2, 3, \ldots$, then the approximating distribution is with probability p (resp. $1 - p$) the sum of $k - 1$ (resp. k) independent exponentials with common mean $1/\mu$. By choosing

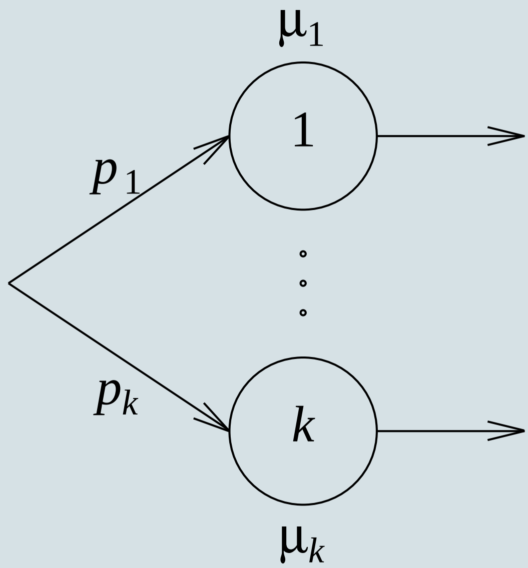
$$p = \frac{1}{1 + c_X^2}[kc_X^2 - \{k(1 + c_X^2) - k^2 c_X^2\}^{1/2}], \qquad \mu = \frac{k - p}{E(X)},$$

the $E_{k-1,k}$ distribution matches $E(X)$ and c_X.

Coefficient of variation greater than 1

In case $c_X \geq 1$, fit a $H_2(p_1, p_2; \mu_1, \mu_2)$ distribution.

Phase diagram for the $H_k(p_1, \dots, p_k; \mu_1, \dots, \mu_k)$ distribution:



But the H_2 distribution is not uniquely determined by its first two moments. In applications, the H_2 distribution with *balanced means* is often used. This means that the normalization

$$\frac{p_1}{\mu_1} = \frac{p_2}{\mu_2}$$

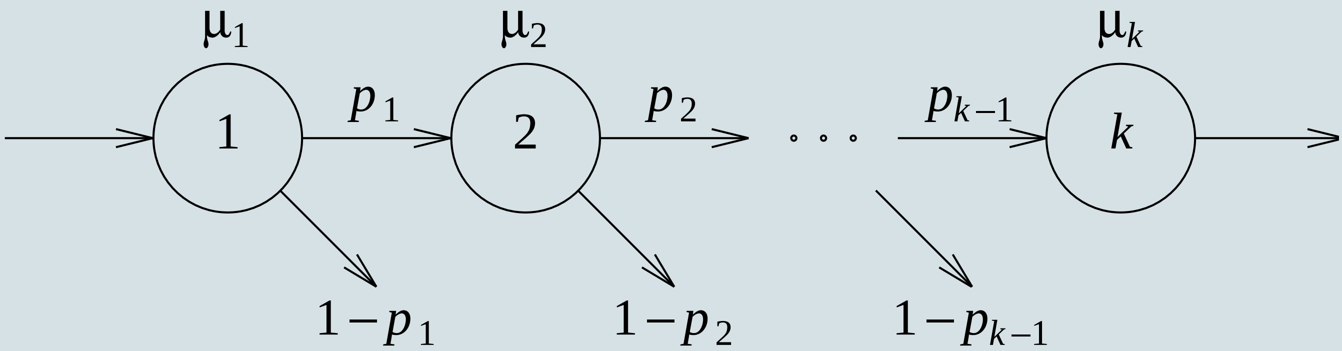
is used. The parameters of the H_2 distribution with balanced means and fitting $E(X)$ and c_X (≥ 1) are given by

$$p_1 = \frac{1}{2}\left(1 + \sqrt{\frac{c_X^2 - 1}{c_X^2 + 1}}\right), \qquad p_2 = 1 - p_1,$$

$$\mu_1 = \frac{2p_1}{E(X)}, \qquad \mu_1 = \frac{2p_2}{E(X)}.$$

In case $c_X^2 \geq 0.5$ one can also use a Coxian-2 distribution for a two-moment fit.

Phase diagram for the Coxian-k distribution:



The following parameter set for the Coxian-2 is suggested:

$$\mu_1 = \frac{2}{E(X)}\left(1+\sqrt{\frac{c_X^2-1/2}{c_X^2+1}}\right), \quad \mu_2 = \frac{4}{E(X)} - \mu_1, \quad p_1 = \frac{\mu_2}{\mu_1}(\mu_1 E(X) - 1)$$

Phase-type distributions may also naturally arise in practical applications.

Example:
The processing of a job involves performing several tasks, where each task takes an exponential amount of time; then the processing time can be described by an Erlang distribution.

Fitting nonnegative discrete distributions

Let X be a random variable on the non-negative integers with mean EX and coefficient of variation c_X. Then it is possible to fit a discrete distribution on $E(X)$ and c_X using the following families of distributions:

- Mixtures of Binomial distributions;
- Poisson distribution;
- Mixtures of Negative-Binomial distributions;
- Mixtures of geometric distributions.

This fitting procedure is described in Adan, van Eenige and Resing (see Probability in the Engineering and Informational Sciences, 9, 1995, pp 623-632).

Adequacy of fit

- Grapical comparison of fitted and empirical curves;
- Statistical tests (*goodness-of-fit tests*).

Output analysis of a simulation

Confidence intervals

Let $X_1, X_2, \ldots, X_n$ be independent *realizations* of a random variable X with unknown mean μ and unknown variance σ^2.

Sample mean

$$\bar{X}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Sample variance

$$S^2(n) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}(n))^2$$

Clearly $\bar{X}(n)$ is an estimator for the unknown mean μ.
How can we construct a confidence interval for μ?

Central limit theorem states that for large n

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}}$$

is approximately a standard normal random variable, and this remains valid if σ is replaced by $S(n)$.
Hence, let $z_\beta = \Phi^{-1}(\beta)$ (e.g., $z_{1-0.025} = 1.96$), then

$$P(-z_{1-\delta/2} \leq \frac{\sum_{i=1}^{n} X_i - n\mu}{S(n)\sqrt{n}} \leq z_{1-\delta/2}) \approx 1-\delta$$

or equivalently

$$P\left(\bar{X}(n) - z_{1-\delta/2}\frac{S(n)}{\sqrt{n}} \leq \mu \leq \bar{X}(n) + z_{1-\delta/2}\frac{S(n)}{\sqrt{n}}\right) \approx 1-\delta$$

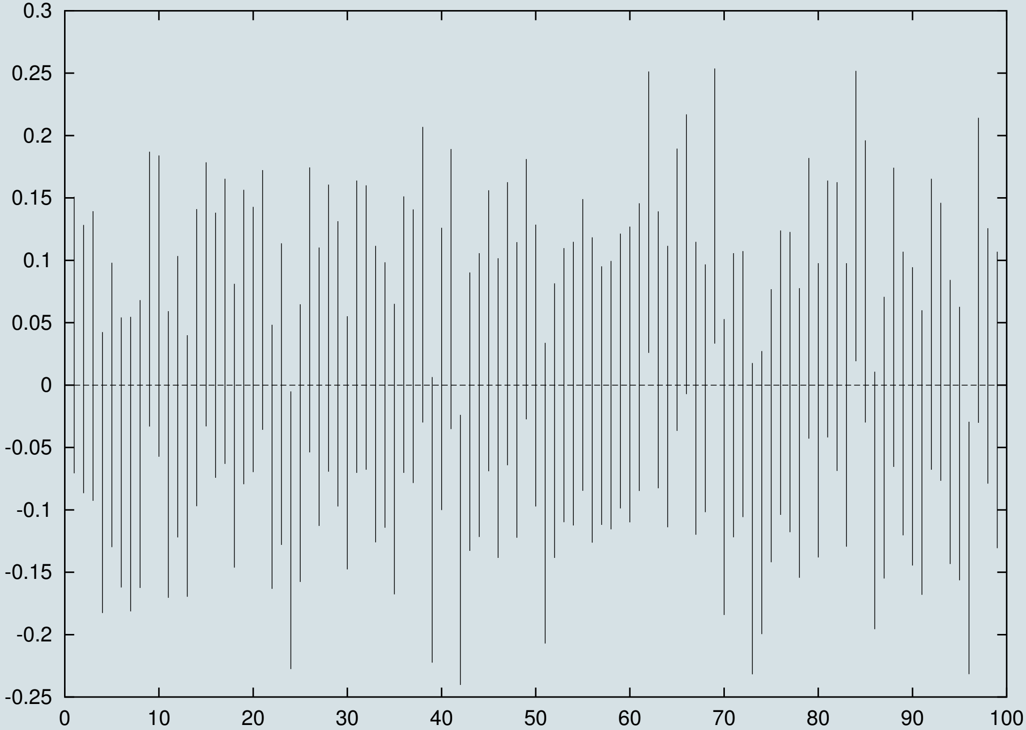
Conclusion:

An approximate $100(1-\delta)\%$ confidence interval for the unknown mean μ is given by

$$\bar{X}(n) \pm z_{1-\delta/2}\frac{S(n)}{\sqrt{n}}$$

As a consequence, to obtain one extra digit of the parameter μ, the required simulation time increases with approximately a factor 100.

100 confidence intervals for the mean of uniform random variable on $(-1, 1)$; each interval is based on 100 observations.



Remark:
If the observations X_i are normally distributed, then

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{S(n)\sqrt{n}}$$

has *for all* n a Student's t distribution with $n-1$ degrees of freedom; so an *exact* confidence interval can be obtained by replacing $z_{1-\delta/2}$ by the corresponding quantile of the t distribution with $n-1$ degrees of freedom.

Remark:
The width of a confidence interval can be reduced by

- increasing the number of observations n;
- decreasing the value of $S(n)$.

The reduction obtained by halving $S(n)$ is the same as the one obtained by producing four times as much observations. Hence, *variance reduction techniques* are important.

Remark:
Recursive computation of the sample mean and variance of the realizations $X_1, \ldots, X_n$ of a random variable X:

$$\bar{X}(n) = \frac{n-1}{n}\bar{X}(n-1) + \frac{1}{n}X_n$$

and

$$S^2(n) = \frac{n-2}{n-1}S^2(n-1) + \frac{1}{n}\left(X_n - \bar{X}(n-1)\right)^2$$

for $n = 2, 3, \ldots$, where

$$\bar{X}(1) = X_1, \qquad S^2(1) = 0.$$