9. Simulation

Literature

- I. Mitrani (1982)
  - "Simulation techniques for discrete event systems"
  - Cambridge University Press, Cambridge
- A.M. Law and W. D. Kelton (1982, 1991)
  - "Simulation modeling and analysis"
  - McGraw-Hill, New York

Contents

What is it?

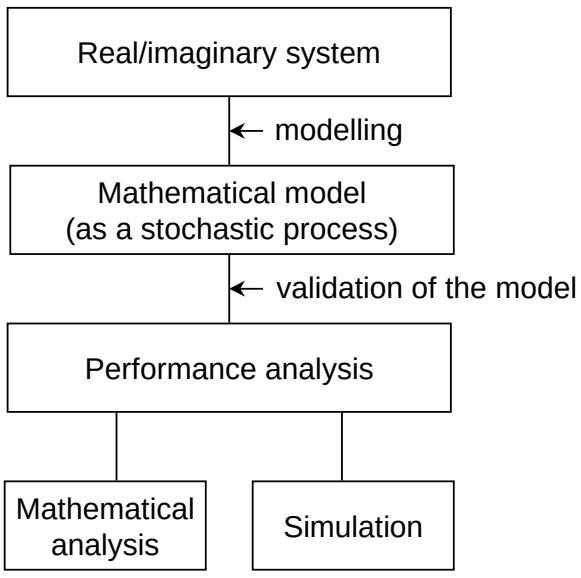
- **Simulation** is (at least from the teletraffic point of view) a statistical method to estimate the performance (or some other important characteristic) of the system under consideration.

- It typically consists of the following four phases:
  - Modelling of the system (real or imaginary) as a dynamic stochastic process
  - Generation of the realizations of this stochastic process (“observations”)
    - Such realizations are called **simulation runs**
  - Collection of data (“measurements”)
  - Statistical analysis of the gathered data, and drawing of the conclusions

Alternative to what?

- In previous lectures, we have got familiar with an alternative way to determine the performance: **mathematical analysis**
- It includes only the following two phases:
  - Modelling of the system as a stochastic process.
    (In this course, we have restricted ourselves to birth-death processes)
  - Solving of the model by means of mathematical analysis
- The modelling phase is common to both of them
- However, the accuracy (faithfulness) of the model that these methods allow can be very different
  - unlike simulation, mathematical analysis typically requires (heavily) restrictive assumptions to be made

Performance analysis of a teletraffic system



Analysis vs. simulation (1)

- **Pros** of analysis
  - Results produced rapidly (after the analysis is made)
  - Exact (accurate) results (for the model)
  - Gives insight
  - Optimization possible (but typically hard)
- **Cons** of analysis
  - Requires restrictive assumptions
    ⇒ the resulting model is typically too simple
    (not capturing all essential features of the system under consideration)
    ⇒ performance analysis of complicated systems impossible
  - Even under these assumptions, the analysis itself may be (extremely) hard

Analysis vs. simulation (2)

- **Pros** of simulation
  - No restrictive assumptions needed (in principle)
    ⇒ performance analysis of complicated systems possible
  - Modelling straightforward
- **Cons** of simulation
  - Production of results time-consuming
    (simulation programs being typically processor intensive)
  - Results inaccurate (however, they can be made as accurate as required by increasing the number of simulation runs, but this takes even more time)
  - Gives not much insight (?)
  - Optimization possible only between very few alternatives (parameter combinations or controls)

Simulation of a stochastic process

- Modelling of the system as a stochastic process
  - This has already been discussed in this course.
  - In the sequel, we will take the model (that is: the stochastic process) for granted. In addition, we will restrict ourselves to simple teletraffic models.
- Generation of the realizations of this stochastic process
  - Generation of random numbers
  - Construction of the realization of the process from event to event (discrete event simulation)
- Collection of data
  - Transient phase vs. steady state (stationarity, equilibrium)
- Statistical analysis and conclusions
  - Point estimators
  - Confidence intervals

Implementation

- Simulation is typically implemented as a computer program
- Simulation program generally comprises the following phases (excluding modelling and conclusions)
  - Generation of the realizations of the stochastic process
  - Collection of data
  - Statistical analysis of the gathered data
- Simulation program can be implemented by
  - a general-purpose programming language, e.g. C or C++
    - most flexible, but tedious and prone to programming errors
  - utilizing simulation-specific program libraries, e.g. CNCL
  - utilizing simulation-specific software, e.g. OPNET
    - most rapid and reliable (depending on s/w), but rigid

Simulation types

- What we have described above, is
  - **discrete** (event-based), **dynamic** (evolving in time) and **stochastic** (random components) simulation
- This is called **discrete event simulation**
  - This is also what we will consider later on in this lecture
- Other types:
  - **continuous** simulation: state and parameter spaces of the process are continuous; description of the system typically by differential equations, e.g. simulation of the trajectory of an aircraft
  - **static** simulation: time plays no role, e.g. numerical integration of a multi-dimensional integral by Monte Carlo method
  - **deterministic** simulation: no random components, e.g. the first example above

Contents

Generation of the realizations of the traffic process

- Assume that the modelling part of simulation has been done
  - So the stochastic process describing the evolution of the system is known
- Next step is to generate realizations of this process.
  - For this, we have to:
    - Generate a realization (value) for all the random variables affecting the evolution of the process (taking properly into account all the (statistical) dependencies between these variables)
    - Using these generated values, construct the realization of the process
  - These two parts are **overlapping** (thus, anything else but sequential)
  - Realizations for random variables are generated by utilizing (**pseudo**) **random number generators**
  - The realization of the process is constructed from event to event (**discrete event simulation**)

Discrete event simulation (1)

- Idea: simulation evolves **from event to event**
  - If nothing happens during an interval, we can just skip it!
- **Event classes**:
  - **basic events** modifying (somehow) the state of the system
    - e.g. arrivals and departures of customers in a simple teletraffic model
  - **extra events** related to the data collection
    - including the event for stopping the simulation run
  - Inside these event classes, there are various **event types**
- Event identification:
  - occurrence time (when) and
  - event type (what)

Discrete event simulation (2)

- Events are organized as an **event list**
  - Events in this list are ordered (ascendingly) by the occurrence time
    - first: the event occurring next
  - Events are handled one-by-one (in this order)
- **Simulation clock** tells the occurrence time of the next event
  - progressing by jumps
- **System state** tells the current state of the system

Discrete event simulation (3)

- General algorithm for a single **simulation run**:
  1 Initialization
    - simulation clock = 0
    - system state = given initial value
    - for each event type, generate next event (whenever possible)
    - construct the event list from these events
  2 Event handling
    - simulation clock = occurrence time of the next event
    - handle the event including
      - generation of new events and their addition to the event list
      - updating of the system state
    - delete the event from the event list
  3 Stopping test
    - if positive, then stop the simulation run; otherwise return to 2

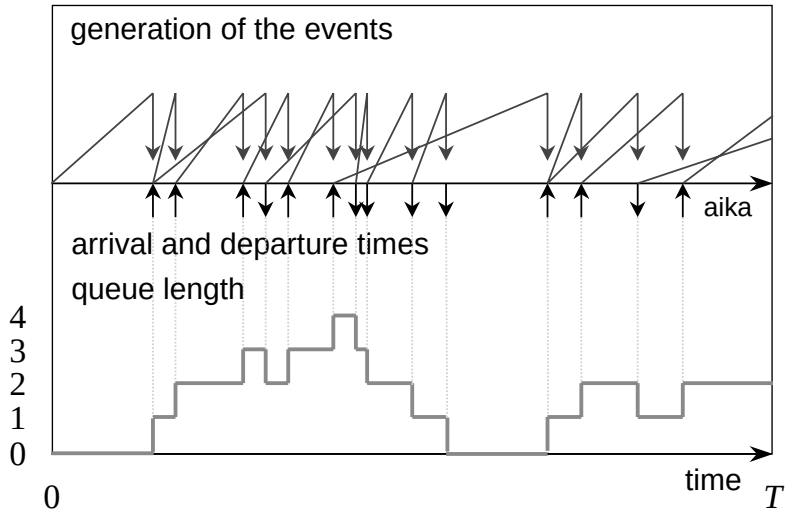
Example (1)

- Task: Simulate the M/M/1 queue (more precisely: the evolution of the queue length process) from time 0 to time T assuming that the queue is empty at time 0 and omitting any data collection
  - System state (at time *t*) = queue length X_t
    - initial value: $X_0 = 0$
  - Basic events:
    - customer arrivals
    - customer departures
  - Extra event:
    - stopping of the simulation run at time T

Example (2)

- Initialization:
  - initialize the system state: $X_0 = 0$
  - generate the time till the first arrival time from the $\mathrm{Exp}(\lambda)$ distribution
- Handling of an arrival event (occurring at some time *t*):
  - if $X_t = 0$, then
    generate the time till the next departure from the $\mathrm{Exp}(\mu)$ distribution
  - generate the time till the next arrival from the $\mathrm{Exp}(\lambda)$ distribution
  - update the system state: $X_t = X_t + 1$
- Handling of a departure event (occuring at some time t):
  - update the system state: $X_t = X_t - 1$
  - if $X_t > 0$, then
    generate the time till the next departure from the $\mathrm{Exp}(\mu)$ distribution
- Stopping test: $t > T$

Example (3)



Contents

Generation of realizations of random variables

- Based on random number generators
- First step:
  - generation of independent $\mathrm{U}(0,1)$ distributed random variables
- Step from the $\mathrm{U}(0,1)$ distribution to the desired distribution:
  - **rescaling** ($\Rightarrow \mathrm{U}(a,b)$)
  - **discretization** ($\Rightarrow$ Bernoulli(p), Bin(n,p), Poisson(a), Geom(p))
  - **inverse transform** ($\Rightarrow \mathrm{Exp}(\lambda)$)
  - **other transforms** ($\Rightarrow \mathrm{N}(0,1) \Rightarrow \mathrm{N}(\mu,\sigma^2)$)
  - **acceptance-rejection method** (for any continuous random variable defined in a finite interval whose density function is bounded)
    - two independent $\mathrm{U}(0,1)$ distributed random variables needed

Random number generator

- **Random number generator** is an algorithm generating (pseudo) random integers Z_i in some interval $0,1,\ldots,m\text{-}1$
  - The resulting numbers Z_i are called (**pseudo**) **random numbers**
  - The sequence generated is **always** periodic (goal: this period to be as long as possible)
  - Strictly speaking, the numbers generated are not “random” at all, in the sense of being unpredictable (thus: pseudo)
  - In practice, however, the numbers “appear” to be IID with uniform distribution, provided that the random number generator is designed carefully
- Validition of a random number generator can be based on empirical (statistical) and theoretical tests:
  - uniformity of the generated empirical distribution
  - independence of the generated random numbers

Random number generator types

- Linear congruential generator
  - most simple
  - next random number is based on just the current one: $Z_{i+1} = f(Z_i)$
- Multiplicative congruential generator
  - a special case of the first type
- Additive congruential generators
- Shuffling, etc.

Linear congruential generator (LCG)

- **Linear congruential generator** (LCG) uses the following algorithm to generate random numbers belonging to $\{0,1,\ldots,m\text{-}1\}$:

$$Z_{i+1} = (aZ_i + c) \bmod m$$

  - Here a, c and m are fixed non-negative integers ($a < m$, $c < m$)
  - In addition, the starting value (**seed**) $Z_0 < m$ should be specified
- Remarks:
  - Parameters a, c and m should be chosen with care, otherwise the result can be very poor
  - By a right choice of parameters, it is possible to achieve the full period m
    - e.g. $m = 2^b$, c odd, $a = 4k + 1$

Multiplicative congruential generator (MCG)

- **Multiplicative congruential generator** (MCG) uses the following algorithm to generate random numbers belonging to $\{0,1,\ldots,m\text{-}1\}$:

$$Z_{i+1} = (aZ_i) \bmod m$$

  - Here a and m are fixed non-negative integers ($a < m$)
  - In addition, the starting value (seed) $Z_0 < m$ should be specified
- Remarks:
  - MCG is clearly a special case of LCG: $c = 0$
  - Parameters a and m should (still) be chosen with care
  - In this case, it is not possible to achieve the full period m
    - e.g. if $m = 2^b$, then the maximum period is 2^{b-2}
  - However, for m **prime**, period m-1 is possible (by a proper choice of a)
    - PMMLCG = prime modulus multiplicative LCG
    - e.g. $m = 2^{31}$-1 and $a = 16{,}807$ (or $630{,}360{,}016$)

U(0,1) distribution

- Let Z denote a (pseudo) random number belonging to $\{0,1,\ldots,m\text{-}1\}$
- Then (approximately)

$$U = \frac{Z}{m} \approx \mathrm{U}(0,1)$$

U(a,b) distribution

- Let $U \sim \mathrm{U}(0,1)$
- Then

$$X = a + (b-a)U \sim \mathrm{U}(a,b)$$

- This is called the **rescaling** method

Discretization method

- Let $U \sim \mathrm{U}(0,1)$
- Assume that Y is a **discrete** random variable
  - with value set $S = \{0,1,\ldots,n\}$ or $S = \{0,1,2,\ldots\}$
- Denote: $F(x) = P\{Y \le x\}$
- Then

$$X = \min\{x \in S \mid F(x) \ge U\} \sim Y$$

- This is called the **discretization** method (cf. inverse transform method)
- Example: $\mathrm{Bernoulli}(p)$ distribution

$$X = \begin{cases} 0, & \text{if } U \le 1-p \\ 1, & \text{if } U > 1-p \end{cases} \sim \mathrm{Bernoulli}(p)$$

Inverse transform method

- Let $U \sim \mathrm{U}(0,1)$
- Assume that Y is a **continuous** random variable
- Assume further that $F(x) = P\{Y \le x\}$ is strictly increasing
- Let $F^{-1}(y)$ denote the inverse of the function $F(x)$
- Then

$$X = F^{-1}(U) \sim Y$$

- This is called the **inverse transform** method
- Proof: Since $P\{U \le u\} = u$ for all u, we have

$$P\{X \le x\} = P\{F^{-1}(U) \le x\} = P\{U \le F(x)\} = F(x)$$

Exp(λ) distribution

- Let $U \sim \mathrm{U}(0,1)$
  - Then also $1\text{-}U \sim \mathrm{U}(0,1)$
- Let $Y \sim \mathrm{Exp}(\lambda)$
  - $F(x) = P\{Y \le x\} = 1\text{-}e^{-\lambda x}$ is strictly increasing
  - The inverse transform is $F^{-1}(y) = -(1/\lambda)\log(1\text{-}y)$
- Thus, by the inverse transform method,

$$X = F^{-1}(1-U) = -\frac{1}{\lambda}\log(U) \sim \mathrm{Exp}(\lambda)$$

N(0,1) distribution

- Let $U_1 \sim \mathrm{U}(0,1)$ and $U_2 \sim \mathrm{U}(0,1)$ be **independent**
- Then, by so called Box-Müller method,
  the following two (transformed) random variables are **independent** and identically distributed obeying the $\mathrm{N}(0,1)$ distribution:

$$X_1 = \sqrt{-2\log(U_1)}\sin(2\pi U_2) \sim N(0,1)$$
$$X_2 = \sqrt{-2\log(U_1)}\cos(2\pi U_2) \sim N(0,1)$$

N(μ,σ²) distribution

- Let $X \sim \mathrm{N}(0,1)$
- Then, by the rescaling method,

$$Y = \mu + \sigma X \sim N(\mu, \sigma^2)$$

Contents

Collection of data

- Our starting point was that simulation is needed to estimate the value, say α, of some performance parameter
  - This parameter may be related to the **transient** or the **steady-state** behaviour of the system.
  - Examples 1 & 2 (transient phase characteristics)
    - average waiting time of the first k customers in an M/M/1 queue assuming that the system is empty in the beginning
    - average queue length in an M/M/1 queue during the interval $[0,T]$ assuming that the system is empty in the beginning
  - Example 3 (steady-state characteristics)
    - the average waiting time in an M/M/1 queue in equilibrium
- Each simulation run yields one sample, say X, describing somehow the parameter under consideration
- For drawing statistically reliable conclusions, multiple samples, $X_1,\ldots,X_n$, are needed (preferably IID)

Transient phase characteristics (1)

- Example 1:
  - Consider e.g. the average waiting time of the first k customers in an M/M/1 queue assuming that the system is empty in the beginning
  - Each simulation run can be stopped when the kth customer enters the service
  - The sample X based on a single simulation run is in this case:

$$X = \frac{1}{k}\sum_{i=1}^{k} W_i$$

    - Here W_i = waiting time of the ith customer in this simulation run
- Multiple IID samples, $X_1,\ldots,X_n$, can be generated by the method of **independent replications:**
  - multiple independent simulation runs (using independent random numbers)

Transient phase characteristics (2)

- Example 2:
  - Consider e.g. the average queue length in an M/M/1 queue during the interval $[0,T]$ assuming that the system is empty in the beginning
  - Each simulation run can be stopped at time T (that is: simulation clock = T)
  - The sample X based on a single simulation run is in this case:

$$X = \frac{1}{T}\int_{0}^{T} Q(t)\,dt$$

    - Here $Q(t)$ = queue length at time t in this simulation run
    - Note that this integral is easy to calculate, since $Q(t)$ is piecewise constant
- Multiple IID samples, $X_1,\ldots,X_n$, can again be generated by the method of independent replications

Steady-state characteristics (1)

- Collection of data in a single simulation run can **typically** (but not always) be done only after a **warm-up** phase (hiding the transient characteristics) resulting in
  - overhead
  - bias in estimation
  - need for determination of a **sufficiently long** warm-up phase
- Multiple samples, $X_1,\ldots,X_n$, may be generated by the following three methods:
  - independent replications
  - batch means
  - regenerative method
- The first two methods require a warm-up phase,
  but the last one (that is: regenerative method) does not

Steady-state characteristics (2)

- Method of **independent replications**:
  - multiple independent simulation runs (using independent random numbers)
  - each simulation run includes the warm-up phase $\Rightarrow$ inefficiency
  - samples IID $\Rightarrow$ accuracy
- Method of **batch means**:
  - one (very) long simulation run divided (artificially) into one warm-up phase and N equal length periods (each of which represents a single simulation run)
  - only one warm-up phase $\Rightarrow$ efficiency
  - samples only approximately IID $\Rightarrow$ inaccuracy, choice of N

Steady-state characteristics (3)

- **Regenerative method**:
  - only possible, if the traffic process is a **regenerative** stochastic process
    - G/G/1 queue (and, thus also M/M/1 and M/G/1) is regenerative (a new cycle starts whenever a new customer arrives in an empty system)
    - all Markov processes are regenerative (a new cycle starts whenever the process enters some fixed state)
  - one (very) long simulation run divided into N IID cycles (each of which represents a single simulation run)
  - **no** warm-up phase $\Rightarrow$ efficiency
  - samples IID $\Rightarrow$ accuracy
  - the problem is that the **cycle lengths**, which are random variables, can be (much too) long $\Rightarrow$ inefficiency

Contents

Parameter estimation

- As mentioned, our starting point was that simulation is needed to estimate the value, say α, of some performance parameter
- Each simulation run yields a (random) sample, say X_i, describing somehow the parameter under consideration
  - Sample X_i is called **unbiased** if $E[X_i] = \alpha$
- Denote the **sample average** by

$$\overline{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i$$

- Assuming that the samples X_i are IID with mean α and variance σ^2, the sample average is **unbiased** and **consistent** estimator of α, since

$$E[\overline{X}_n] = \frac{1}{n}\sum_{i=1}^{n} E[X_i] = \alpha$$

$$D^2[\overline{X}_n] = \frac{1}{n^2}\sum_{i=1}^{n} D^2[X_i] = \frac{1}{n}\sigma^2 \to 0 \quad (\text{as } n \to \infty)$$

Example

- Consider the average waiting time of the first 25 customers in an M/M/1 queue with load ρ = 0.9 assuming that the system is empty in the beginning
  - Theoretical value: α = 2.12
  - Samples X_i from ten simulation runs (n = 10):
    - 1.05, 6.44, 2.65, 0.80, 1.51, 0.55, 2.28, 2.82, 0.41, 1.31
  - Sample average (point estimate for α):

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{10}(1.05 + 6.44 + \ldots + 1.31) = 1.98$$

Confidence interval (1)

- Assume that X_i's are IID with unknown mean α and **known** variance σ^2
- By the Central Limit Theorem (see Lecture 5, Slide 49), for large n,

$$Z := \frac{\overline{X}_n - \alpha}{\sigma / \sqrt{n}} \sim \mathrm{N}(0,1)$$

- Let z_p denote the p-fractile of the $\mathrm{N}(0,1)$ distribution, that is: $P\{Z \le z_p\} = p$, where $Z \sim \mathrm{N}(0,1)$
  - Example: for $\beta = 5\%$, $z_{1-(\beta/2)} = z_{0.975} \approx 1.96 \approx 2$
- The following interval is called the **confidence interval** for the sample average at **confidence level** $1 - \beta$:

$$\overline{X}_n \pm z_{1-\frac{\beta}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$

  - This means that $P\{|\overline{X}_n - \alpha| \le z_{1-(\beta/2)}\, \sigma/\sqrt{n}\} = 1 - \beta$, that is: "with probability $1 - \beta$, the parameter α belongs to this interval"

Confidence interval (2)

- In general, however, the variance σ^2 is unknown (in addition to the mean α)
- It can be estimated by the **sample variance**:

$$S_n^2 := \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - n\overline{X}_n^2\right)$$

- It is possible to prove that the sample variance is an unbiased and consistent estimator of σ^2:

$$E[S_n^2] = \sigma^2$$

$$D^2[S_n^2] \to 0 \quad (\text{as } n \to \infty)$$

Confidence interval (3)

- Assume that X_i's are IID obeying the $\mathrm{N}(\alpha,\sigma^2)$ distribution with unknown mean α and **unknown** variance σ^2
- Then it is possible to show that

$$T := \frac{\overline{X}_n - \alpha}{S_n / \sqrt{n}} \sim \mathrm{Student}(n-1)$$

- Let $t_{n\text{-}1,p}$ denote the p-fractile of the Student(n-1) distribution, that is: $P\{T \le t_{n\text{-}1,p}\} = p$, where $T \sim$ Student(n-1)
  - Example 1: for $n = 10$ and $\beta = 5\%$, $t_{n\text{-}1,1\text{-}(\beta/2)} = t_{9,0.975} \approx 2.26 \approx 2$
  - Example 2: for $n = 100$ and $\beta = 5\%$, $t_{n\text{-}1,1\text{-}(\beta/2)} = t_{99,0.975} \approx 1.98 \approx 2$
- Thus, the confidence interval for the sample average at confidence level $1 - \beta$ is now as follows:

$$\overline{X}_n \pm t_{n-1,1-\frac{\beta}{2}} \cdot \frac{S_n}{\sqrt{n}}$$

Example (continued)

- Consider the average waiting time of the first 25 customers in an M/M/1 queue with load $\rho = 0.9$ assuming that the system is empty in the beginning
  - Theoretical value: $\alpha = 2.12$
  - Samples X_i from ten simulation runs (n = 10):
    - 1.05, 6.44, 2.65, 0.80, 1.51, 0.55, 2.28, 2.82, 0.41, 1.31
  - Sample average = 1.98 and the square root of the sample variance:

$$S_n = \sqrt{\frac{1}{9}\left((1.05-1.98)^2 + \ldots + (1.31-1.98)^2\right)} = 1.78$$

  - So, the confidence interval (that is: interval estimate for α) at confidence level 95% is

$$\overline{X}_n \pm t_{n-1,1-\frac{\beta}{2}} \cdot \frac{S_n}{\sqrt{n}} = 1.98 \pm 2.26 \cdot \frac{1.78}{\sqrt{10}} = 1.98 \pm 1.27 = (0.71, 3.25)$$

Observations

- Simulation results become more accurate (that is: the interval estimate for α becomes narrower) when
  - the number n of simulation runs is increased, or
  - the variance σ^2 of each sample is reduced
- Given the desired accuracy for the simulation results,
  the number of required simulation runs can be determined dynamically

THE END