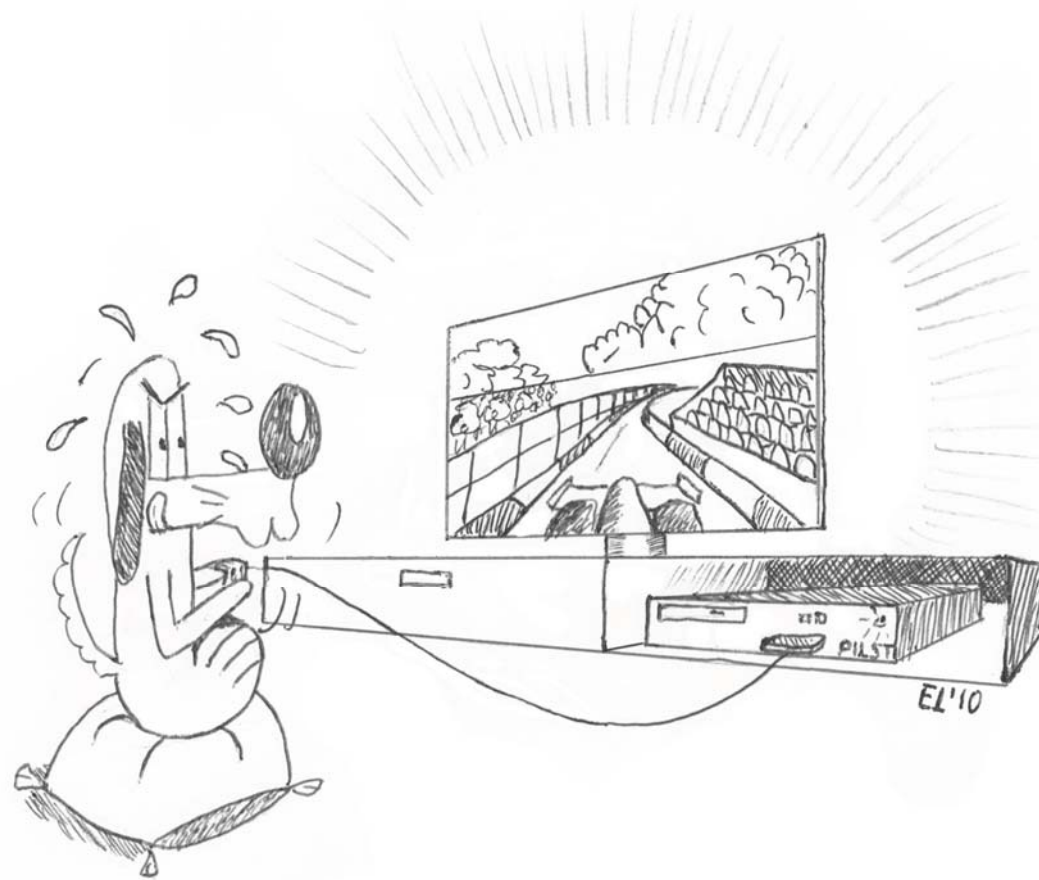


# Simulation



Where real stuff starts

# ToC

- What, transience, stationarity
- How, discrete event, recurrence
  - Accuracy of output
- Random Number Generators
  - How to sample
  - Monte Carlo

# 1 What is a simulation ?

- An experiment in computer
- Important differences
  - ▶ Simulated vs. real time
  - ▶ Serialization of events

# Stationarity

- A simulation can be terminating or not

---

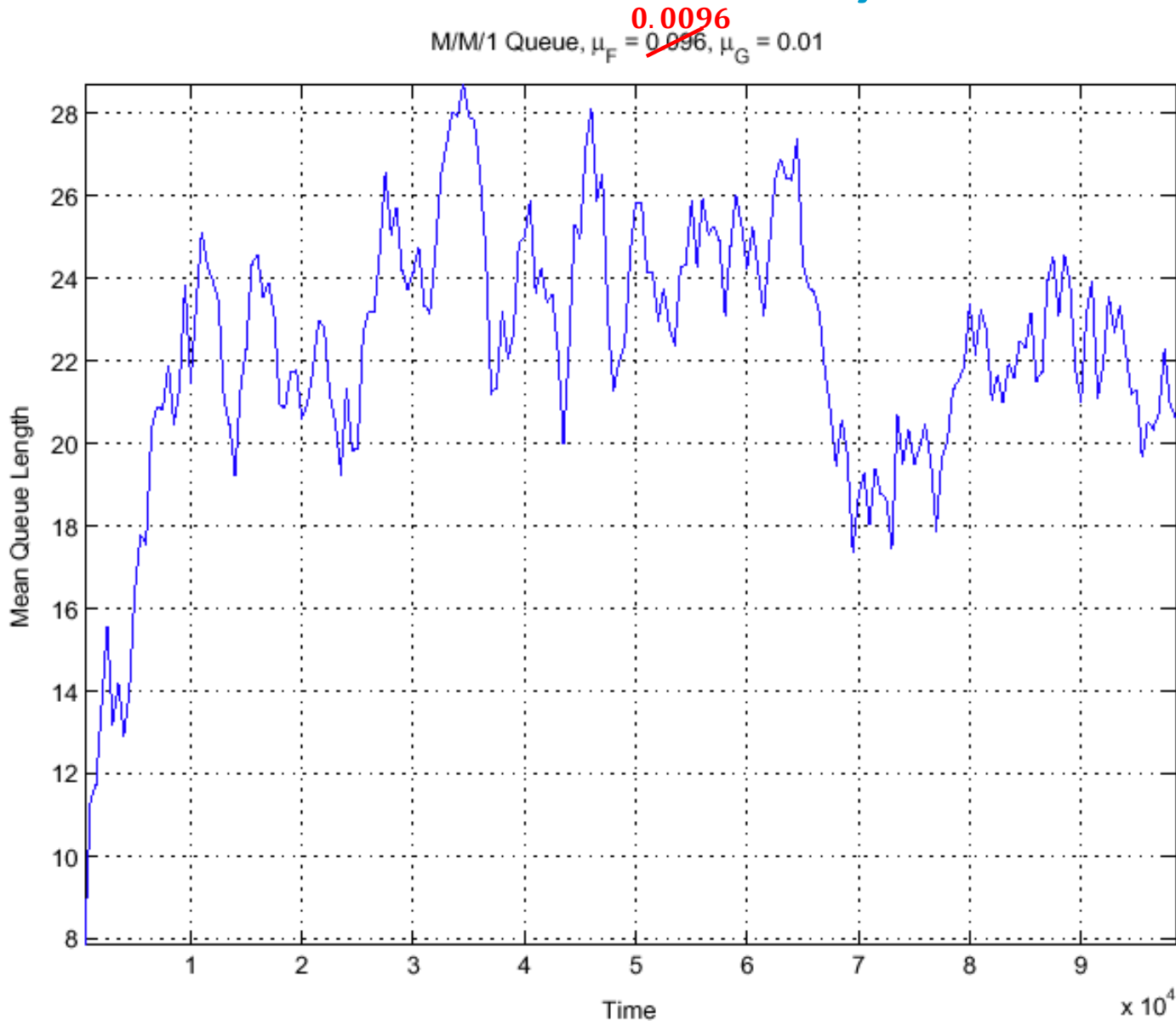
EXAMPLE 6.2: **JOE'S COMPUTER SHOP.** We are interested in evaluating the time it takes to serve  $n$  customers who request a file together at time 0. We run a simulation program that terminates at time  $T_1$  when all users have their request satisfied. This is a terminating simulation; its output is the time  $T_1$ .

- For a terminating simulation you should make sure it is *stationary*

---

EXAMPLE 6.3: **INFORMATION SERVER.** An information server is modelled as a queue. The simulation program starts with an empty queue. Assume the arrival rate of requests is smaller than the server can handle. Due to the fluctuations in the arrival process, we expect some requests to be held in the queue, from time to time. After some simulated time, the queue starts to oscillate between busy periods and idle periods. At the beginning of the simulation, the behaviour is not typical of the stationary regime, but after a short time it becomes so (Figure 6.1 (a)).

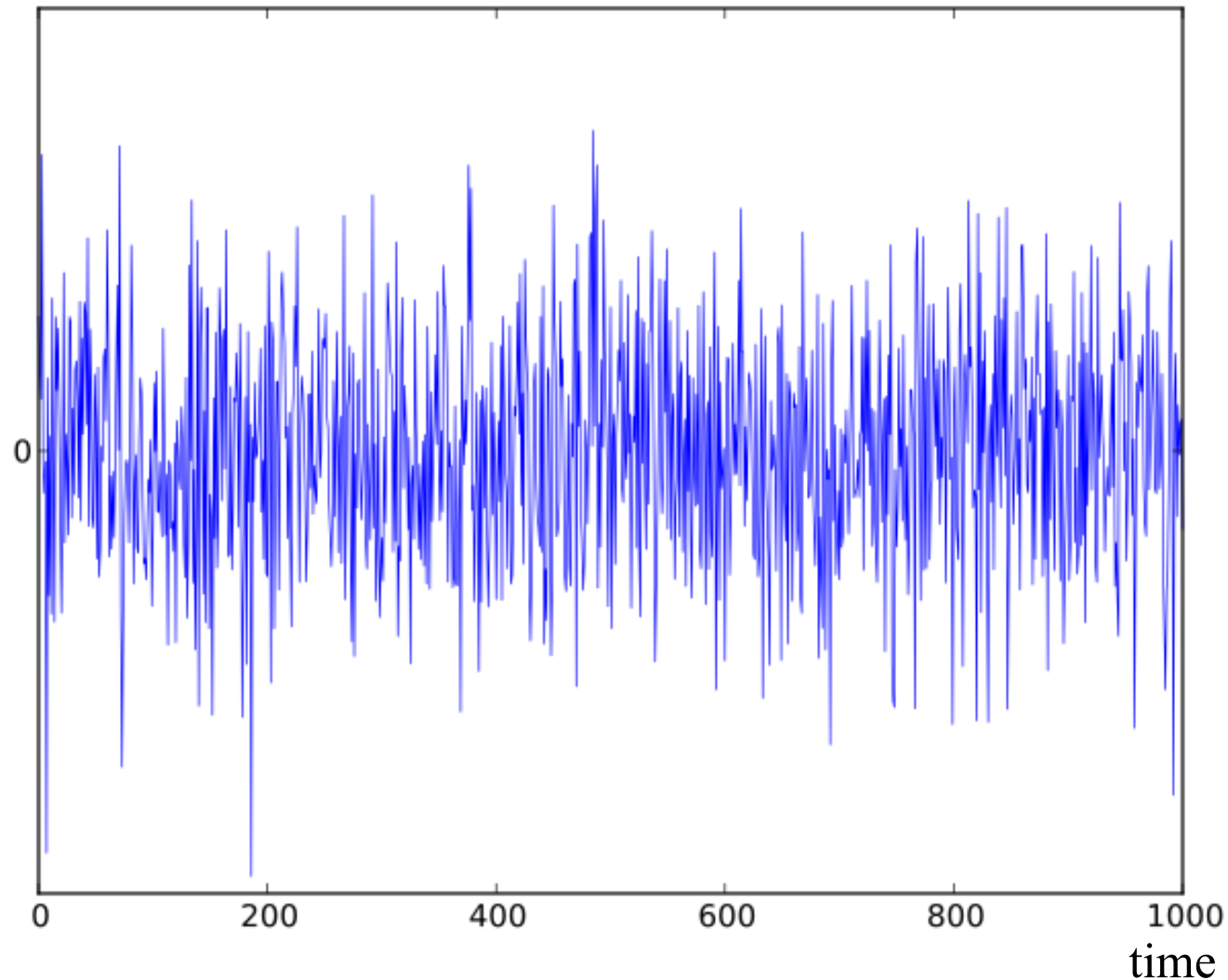
# Information server, scenario 1



(a) Utilization = 0.96

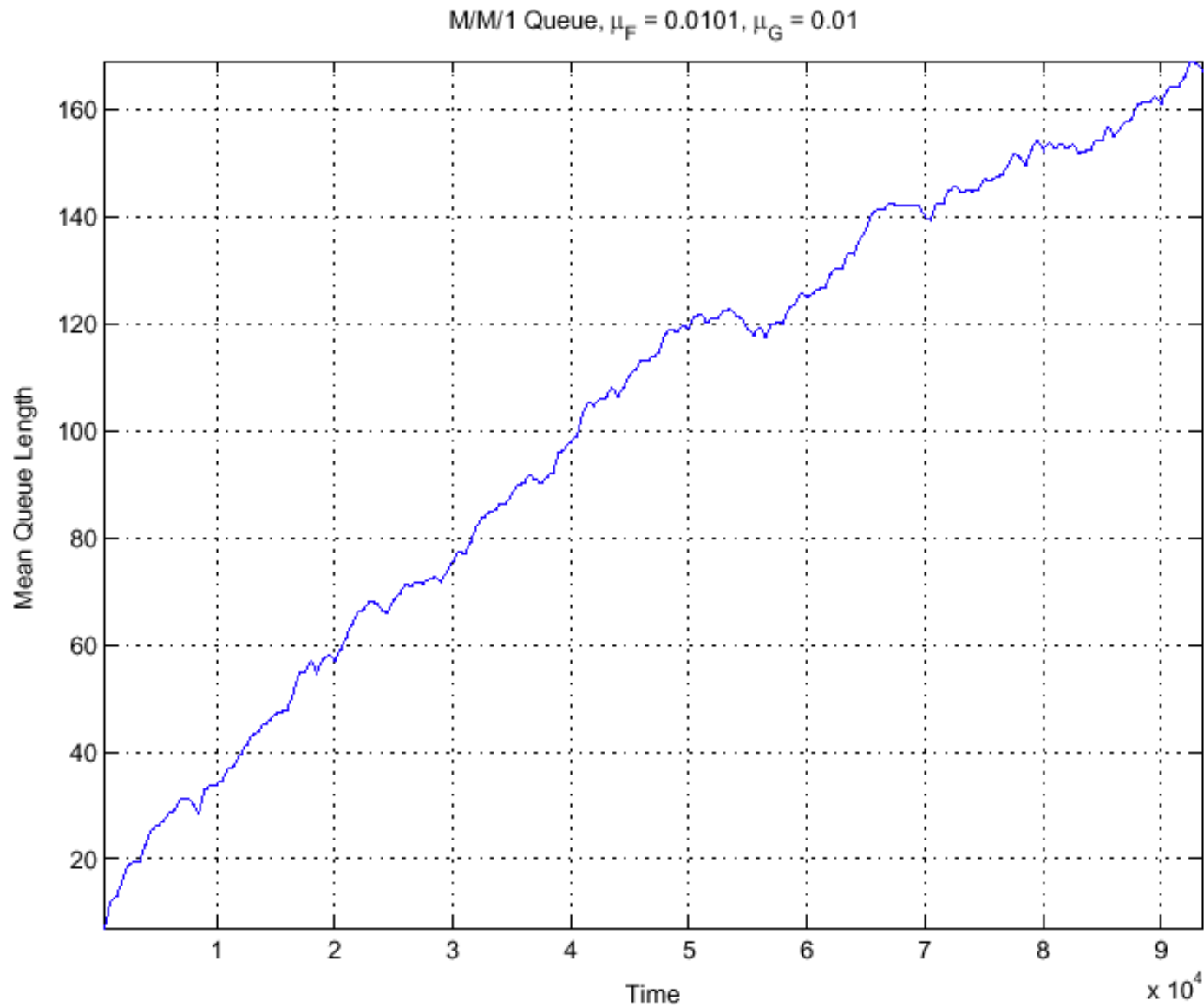
N.B.: Dependency on **previous states** does **not** violate the definition of stationarity.

# What do independent states look like?



AWGN (Additive White Gaussian Noise) is an example of a stochastic process with i.i.d. outputs  
**No continuity whatsoever between output values**

# Information server, scenario 2



(b) Utilization = 1.01

# Stationarity

## ■ In scenario 1:

- ▶ Transient phase, followed by “typical” (=stationary) phase
- ▶ You want to measure things only in the stationary phase
  - ▶ Otherwise: non reproducible, non typical

## ■ In scenario 2:

- ▶ There is no stationary regime  
“walk to infinity”

you should not do a non terminating simulation with this scenario

### *Intuitive Definition*

**A stationary simulation is such that you gain no information about its age by analyzing it.**



# Definition of Stationarity

**STATIONARITY** We assume that we are observing the output of a simulation, which we interpret as a sample of a stochastic process  $S(t)$ . Time  $t$  is either discrete or continuous. This process is *stationary* if for any  $n$ , any sequence of times  $t_1 < t_2 < \dots < t_n$  and any time shift  $u$  the joint distribution of  $(S(t_1 + u), S(t_2 + u), \dots, S(t_n + u))$  is independent of  $u$ .

- i.e. simulation does not get “old”

N.B.: stationary does not preclude dependencies between the states at different times,  $S_{t_1}, S_{t_2}, \dots, S_{t_k}$ .

N.B.: Both notations, i.e.,  $X_t$  and  $S_t$ , are used in the textbook.

# Classical Cases

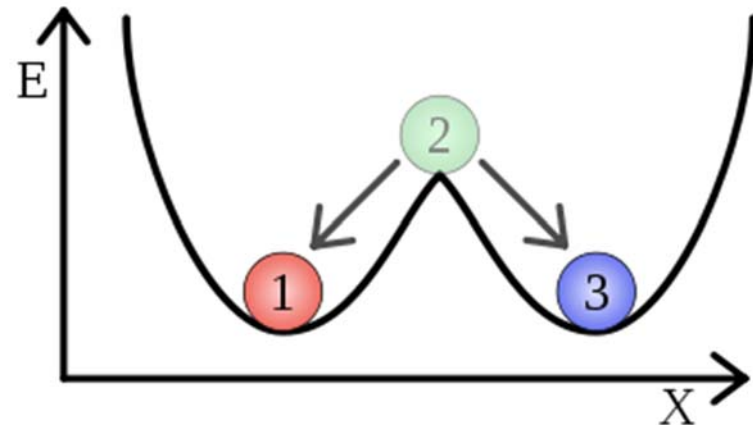
## ■ Markov models

- ▶ State  $X_t$  is sufficient to draw the future of the simulation
- ▶ Quite common case for all simulations

## ■ For a Markov model, over a discrete state space (**NOT necessarily finite**)

- ▶ If you run the simulation long enough it will either walk to infinity (unstable) or converge to stationary
  - ▶ Ex: queue with  $\rho > 1$ : unstable
  - ▶ queue with  $\rho < 1$ : becomes stationary after transient
- ▶ If the state space is **strongly connected** (any state can be reached from any state) then there is 0 or 1 stationary regime
  - ▶ Ex: queue  $\rightarrow$  **either unstable or stable!**
- ▶ Else, there may be several distinct stationary regimes (**Non-ergodic**)
  - ▶ Ex: system with failure modes

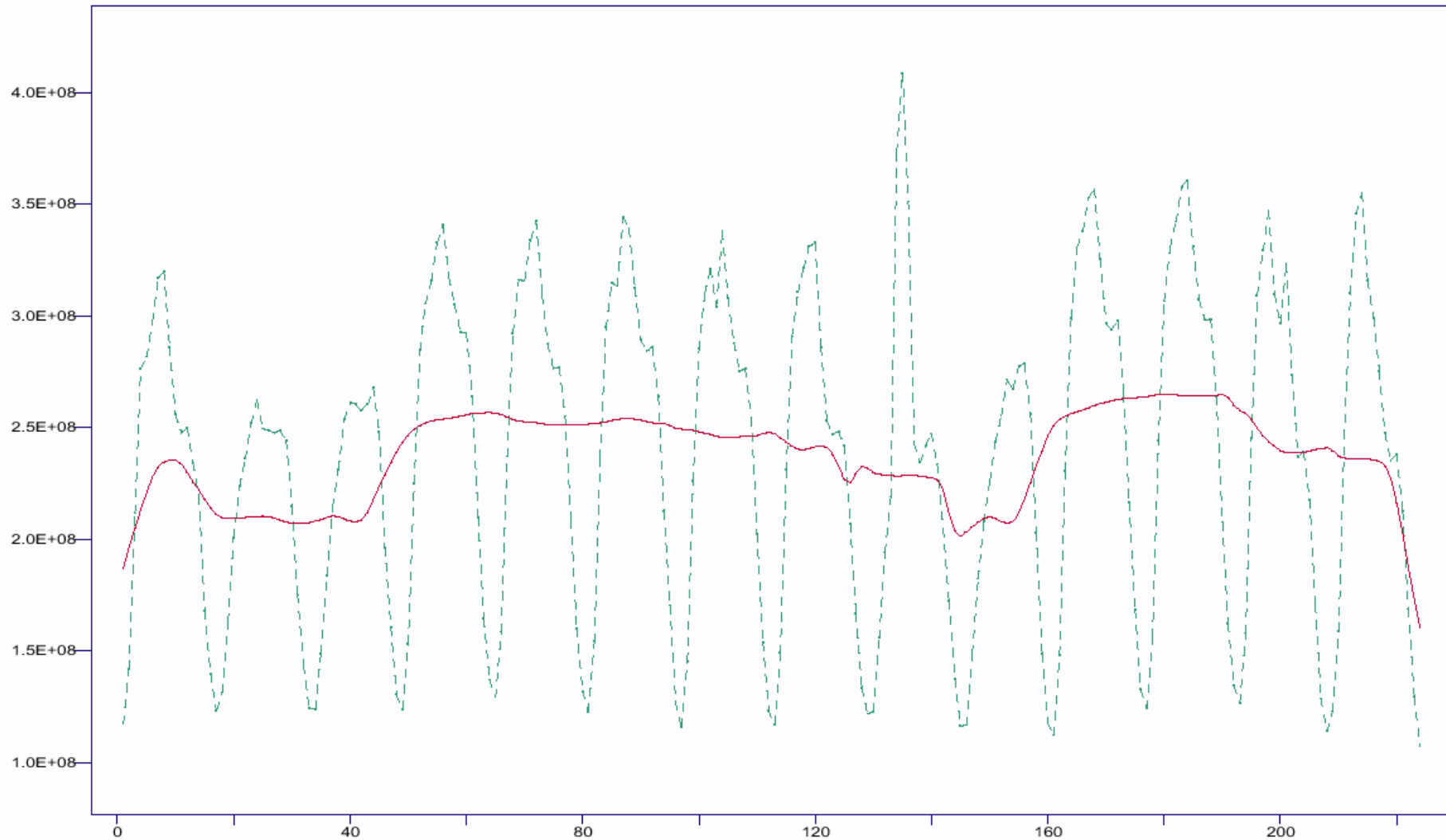
**Bistable system:  
Non-ergodic yet stationary**



# Stationarity and Transience

- Knowing whether a model is stationary is sometimes a hard problem
    - ▶ We will see important models where this is solved
    - ▶ Ex: Solvable for single queues, **not readily solvable** for networks of queues
  - Reasoning about your system may give you indications
    - ▶ Do you expect growth ?
    - ▶ Do you expect seasonality ?
  - Once you believe your model is stationary, you should handle transients
    - ▶ Remove (how ? Look at your output and guess)
    - ▶ Sometimes it is possible to avoid transients at all.
- “*Perfect Simulation*” technique in Chapter 7

# Non-Stationary (Time Dependent Inputs)



N.B.: Dependency on **time** leads to the violation of the definition of stationarity.

# Typical Reasons For Non Stationarity

## ■ Obvious **dependency on time**

- ▶ Seasonality, growth
- ▶ Can be ignored at small time scale (minute or second)
  - ▶ By defining the state of the simulation  $X_t$  on a **coarser** time scale

## ■ Instability: **Explosion**

- ▶ Queue with utilization factor  $>1$

## ■ Instability: **Freezing Simulation**

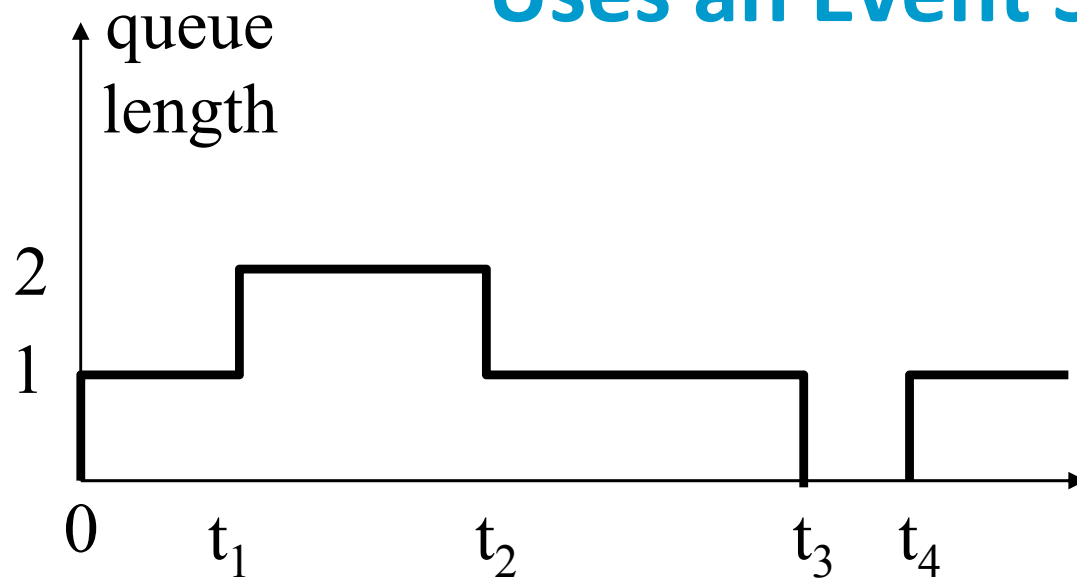
- ▶ System becomes slower with time (**aging**)
- ▶ Typically because there are rare events of large impact (« Kings »)  
The longer the simulation, the larger the largest king
- ▶ Ex: time between regeneration points has **infinite** mean
  
- ▶ We'll come back to this in the chapter « Importance of the View Point »

# 2 Simulation Techniques

- Discrete Event Simulations (aka DES)
- Recurrences
  - ▶ Stochastic recurrences

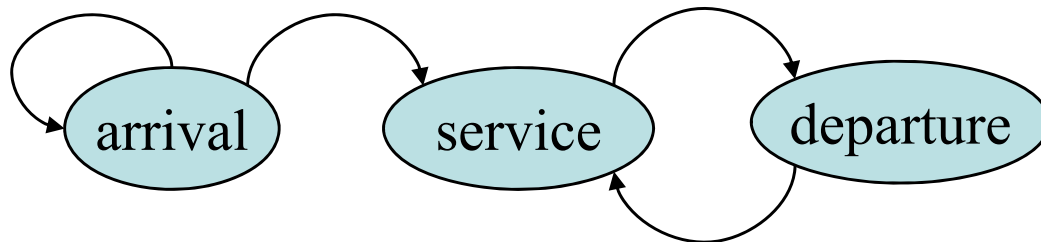
# Discrete event simulation

## Uses an Event Scheduler



### ■ Example:

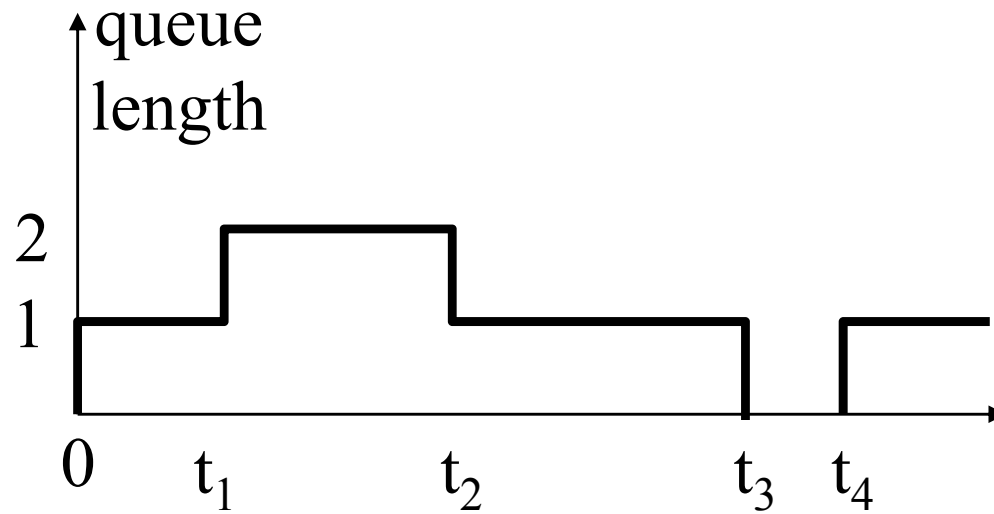
- ▶ Information system modelled as a single server queue
- ▶ Three event classes
  - ▶ arrival
  - ▶ service
  - ▶ departure
- ▶ One event scheduler (global system clock)



Events and their dependencies

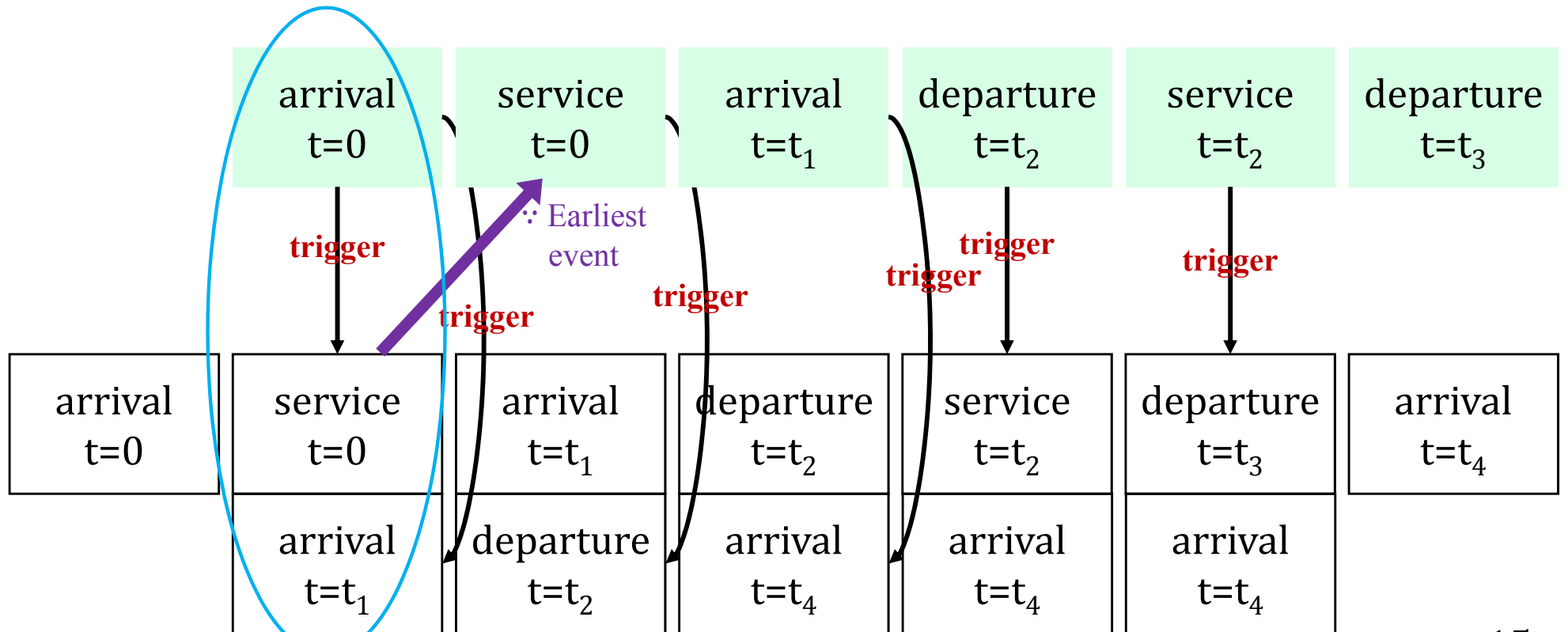
*(How events are triggered, i.e., which events should be added into the event scheduler)*

# Scheduler: Timeline



*initialization*      *Step 1*      *Step 2*      *Step 3*      *Step 4*      *Step 5*      *Step 6*

*State of scheduler*      *Event being executed*





# Statistical Counters

- Assume we want to output: mean queue length and mean response time.  
How do we do this ?

Statistics Counters: `queueLengthCtr` is  $\int_0^t q(s)ds$  where  $q(s)$  is the value of `buffer.length` at time  $s$  and  $t$  is the current time. At the end of the simulation, the mean queue length is `queueLengthCtr/T` where  $T$  is the simulation finish time.

The counter `responseTimeCtr` holds  $\sum_{m=1}^m R_m$  where  $R_m$  is the response time for the  $m$ th request and  $n$  is the value of `nbRequests` at the current time. At the end of the simulation, the mean response time is `responseTimeCtr/N` where  $N$  is the value of `nbRequests`.

- Note the difference between
  - ▶ **Event based** statistic : response time
  - ▶ **Time based** statistic : mean queue length

$R_m$  : the period of time  
from the arrival **event** of request  $m$   
to its departure **event**.

# A Classical Organization of Simulation Code

- Events contain specific code
- A main loop advances the state of the scheduler
- Example: in the code of a *departure* event (the queue is leaving the system)

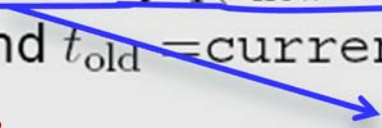
Departure: Update Event Based Counters. Let  $c$  be the request at the head of buffer. Increment `responseTimeCtr` by  $d - a$ , where  $d$  is this event's `firingTime` and  $a$  is the arrival time of the request  $c$ . Increment `nbRequests` by 1.

Execute Event's Actions. Remove the request  $c$  from `buffer` and delete it.

Schedule Follow-Up Events. If `buffer` is not empty after the removal, create a new event of class `Service`, with `firingTime` equal to this event's `firingTime`, and insert it into `eventScheduler`.

## ■ Main program

### Insertion of one arrival event at time=0

- Bootstrapping. Create a new event of class `Arrival` with `firingTime` equal to 0 and insert it into `eventScheduler`.
- Execute Events. While the simulation stopping condition is not fulfilled, do the following.
  - Increment Time Based Counters. Let  $e$  be the **Earliest!** first event in `eventScheduler`. Increment `queueLengthCtr` by  $q(t_{\text{new}} - t_{\text{old}})$  where  $q = \text{buffer.length}$ ,  $t_{\text{new}} = e.\text{firingTime}$  and  $t_{\text{old}} = \text{currentTime}$ .  

  - Execute  $e$ . Either “Arrival” or “Service” or “Departure”
  - Set `currentTime` to `e.firingTime`
  - Delete  $e$
- Termination. Compute the final statistics:  
 $\text{meanQueueLength} = \text{queueLengthCtr} / \text{currentTime}$   
 $\text{meanResponseTime} = \text{responseTimeCtr} / \text{nbRequests}$

**Total simulated time reached?**

**Earliest!**

Integration of queue length over time

# Stochastic Recurrence

- An alternative to discrete event simulation
  - ▶ faster but requires more work on the model
  - ▶ not always applicable
- Defined by iteration:

$$\begin{cases} X_0 = x_0 \\ X_{n+1} = f(X_n, Z_n) \end{cases} \begin{array}{l} \text{(0) No event scheduler : } X_n \rightarrow X_{n+1} \\ \text{(1) Compactified evolution of state} \\ \text{(2) Hard to tweak for further extensions} \end{array}$$

where  $X_n$  is the state of the system at the  $n$ th transition (For any realization,  $X_n$  is in some possibly complicated state space  $\mathcal{X}$ ),  $x_0$  is a fixed, given state in  $\mathcal{X}$ ,  $Z_n$  is some stochastic process that can be simulated (for example a sequence of iid random variables, or a Markov chain), and  $f$  is a deterministic mapping.  $Z_n$  can be viewed as an environmental stochastic process.

The simulated time  $T_n$  at which the  $n$ th transition occurs is assumed to be included in the state variable  $X_n$ .

# Example: random waypoint mobility model

EXAMPLE 6.5: **RANDOM WAYPOINT.**

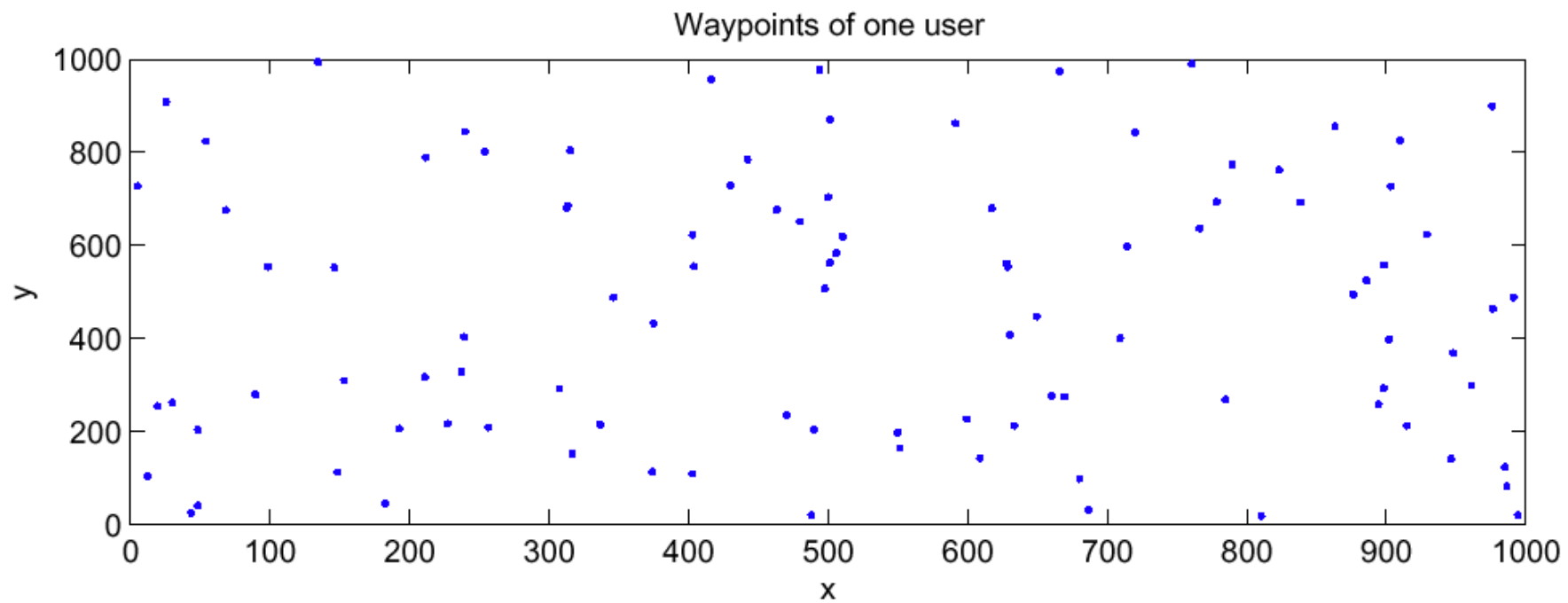
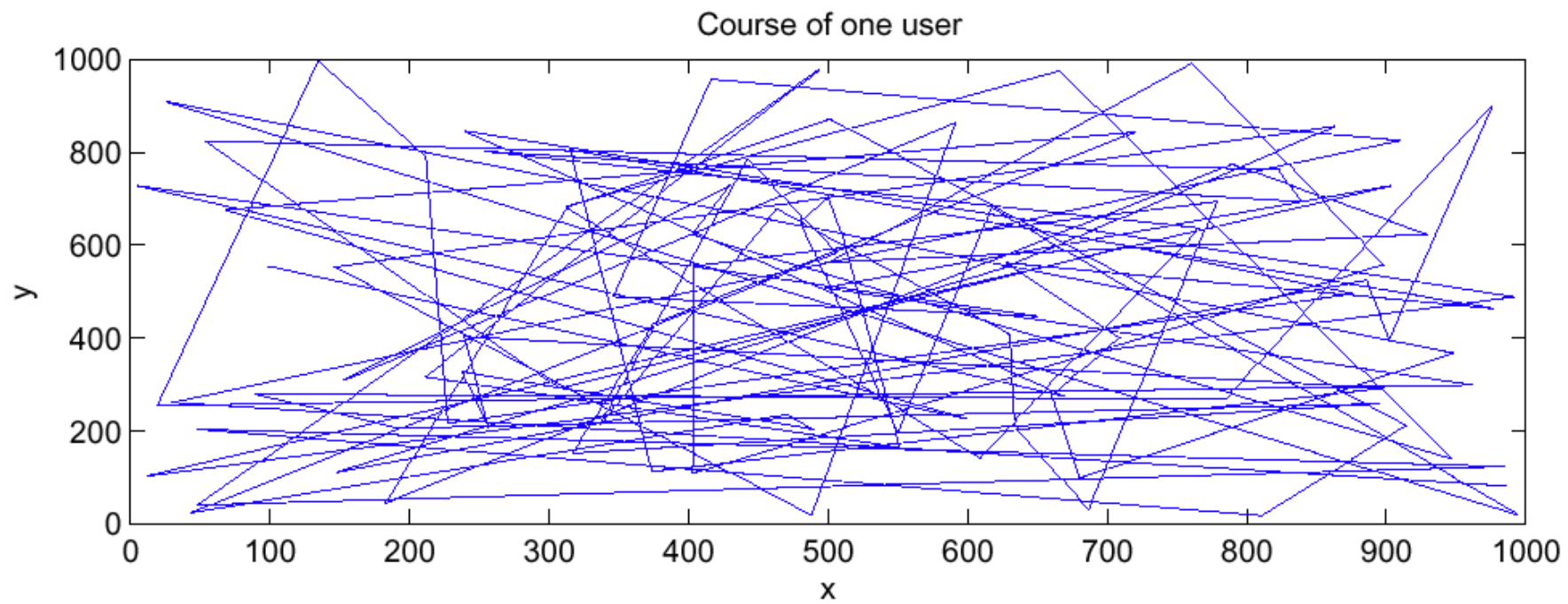
adopted in *Mobile Ad-Hoc Networks* for decades.

The *random waypoint* is a model for a mobile point, and can be used to simulate the mobility pattern in Example 6.1. It is defined as follows. The state variable is  $X_n = (M_n, T_n)$  where  $M_n$  is the position of the mobile at the  $n$ th transition (the  $n$ th “waypoint”) and  $T_n$  is the time at which this destination is reached. The point  $M_n$  is chosen at random, uniformly in a given convex area  $\mathcal{A}$ . The speed at which the mobile travels to the next waypoint is also chosen at random uniformly in  $[v_{\min}, v_{\max}]$ .

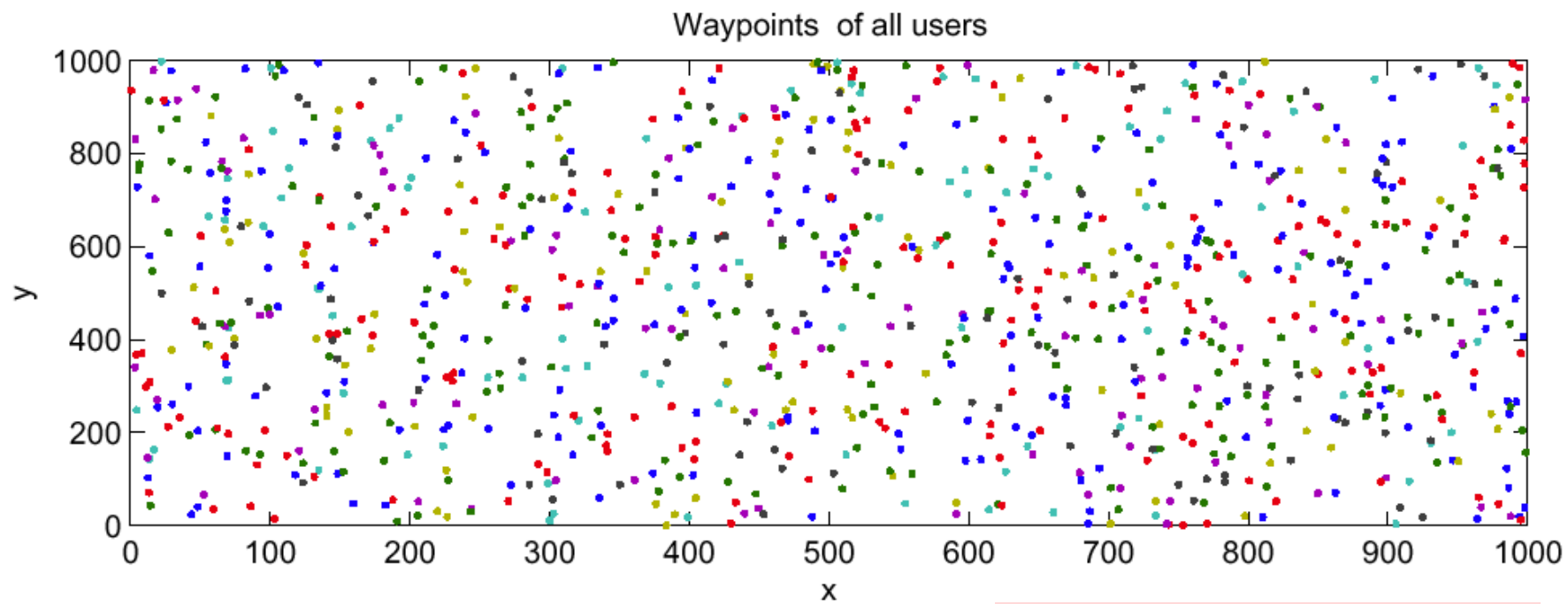
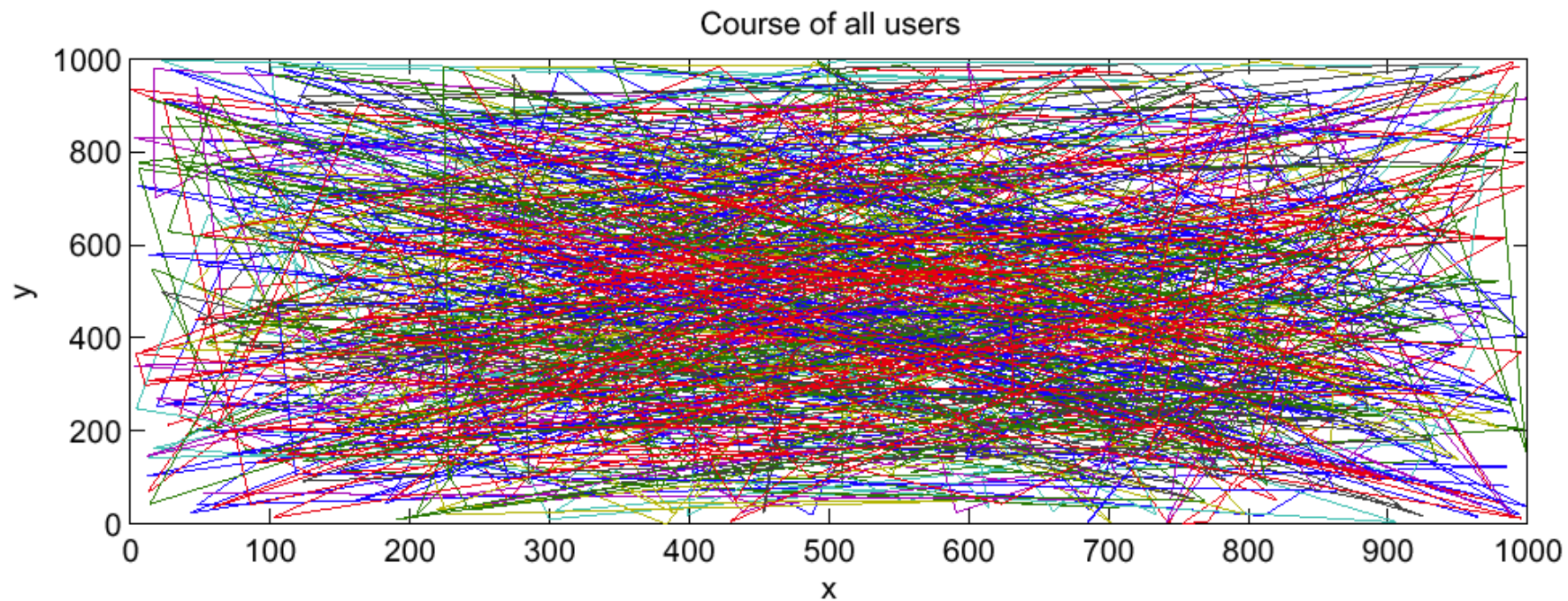
The random waypoint model can be cast as a stochastic recurrence by letting  $Z_n = (M_{n+1}, V_{n+1})$ , where  $M_{n+1}, V_{n+1}$  are independent i.i.d. sequences, such that  $M_{n+1}$  is uniformly distributed in  $\mathcal{A}$  and  $V_{n+1}$  in  $[v_{\min}, v_{\max}]$ . We have then the stochastic recurrence

$$X_{n+1} := (M_{n+1}, T_{n+1}) = \left( M_{n+1}, T_n + \frac{\|M_{n+1} - M_n\|}{V_n} \right)$$

For each **single** mobile!



(a) 1 mobile



(b) 10 mobiles

NB: Event average!

# Queuing System implemented as Stochastic Recurrence

Let  $X_n$  represent the state of the simulator just after an arrival or a departure, as follows: reckoned as a convention

$$X_n = (t_n, b_n, q_n, a_n, d_n)$$

with  $t_n$  = the simulated time at which this transition<sup>n</sup> occurs,  $b_n$  = `buffer.length`,  $q_n$  = `queueLengthCtr` (both just after the transition),  $a_n$  = the time interval from this transition to the next arrival and  $d_n$  = the time interval from this transition to the next departure.

Let  $Z_n$  be a couple of two random numbers, drawn independently of anything else, with distribution uniform in  $(0, 1)$ . ( $z_1, z_2$ )

The recurrence is defined by  $f((t, b, q, a, d), (z_1, z_2)) = (t', b', q', a', d')$  with



if  $a < d$  // this transition is an arrival

$$\Delta = a$$

$$t' = t + a$$

$$b' = b + 1$$

$$q' = q + b\Delta$$

$$a' = F^{-1}(z_1)$$

$$\text{if } b == 0 \text{ then } d' = G^{-1}(z_2) \text{ else } d' = d - \Delta$$

else // this transition is a departure

$$\Delta = d$$

$$t' = t + d$$

$$b' = b - 1$$

$$q' = q + b\Delta$$

$$a' = a - \Delta$$

$$\text{if } b' > 0 \text{ then } d' = G^{-1}(z_1) \text{ else } d' = \infty$$

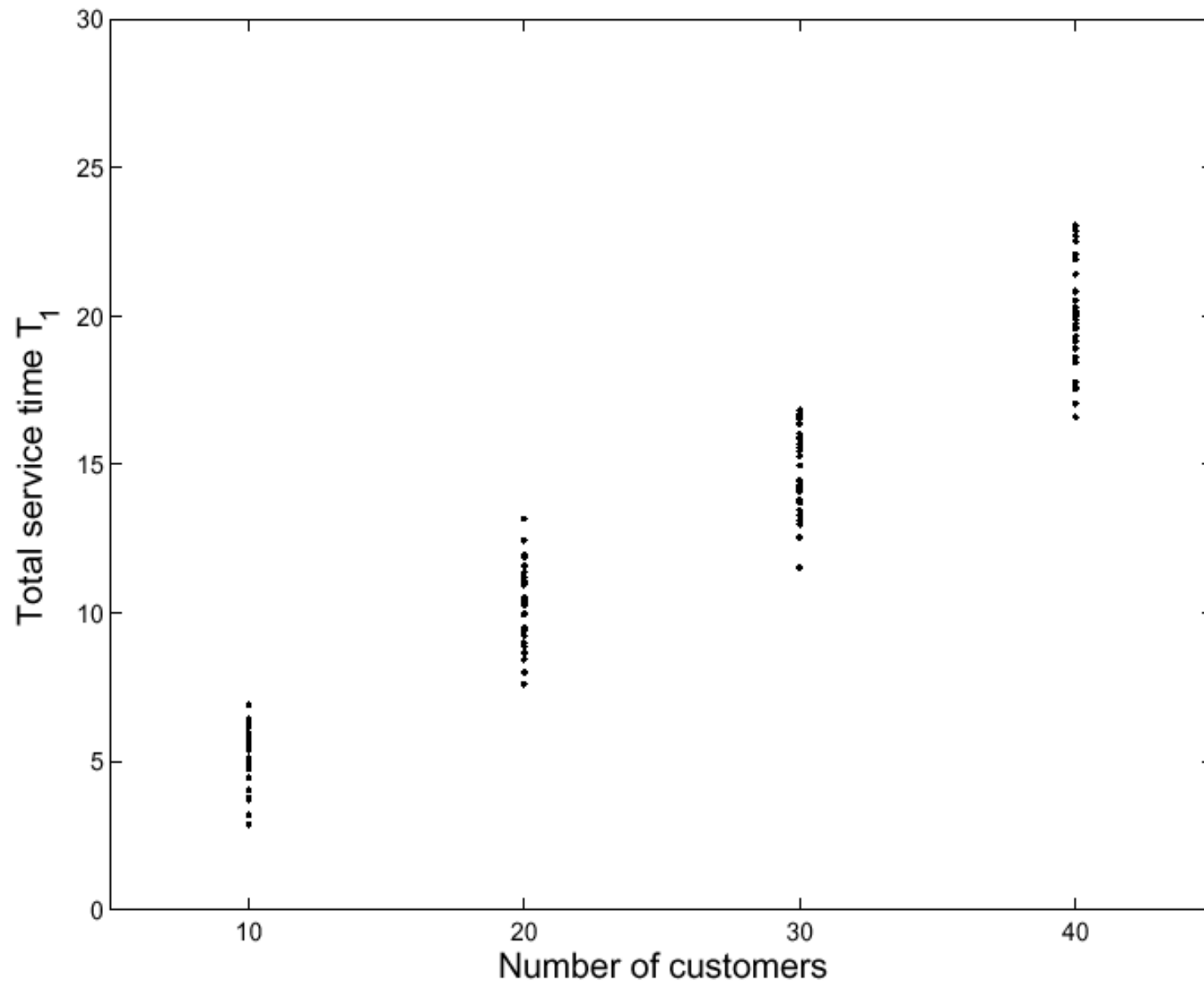
You should brood over a few steps ahead to determine the next state variables, rather than throwing each event to the scheduler in DES case.

**Too compact to be amended for new functionalities!**

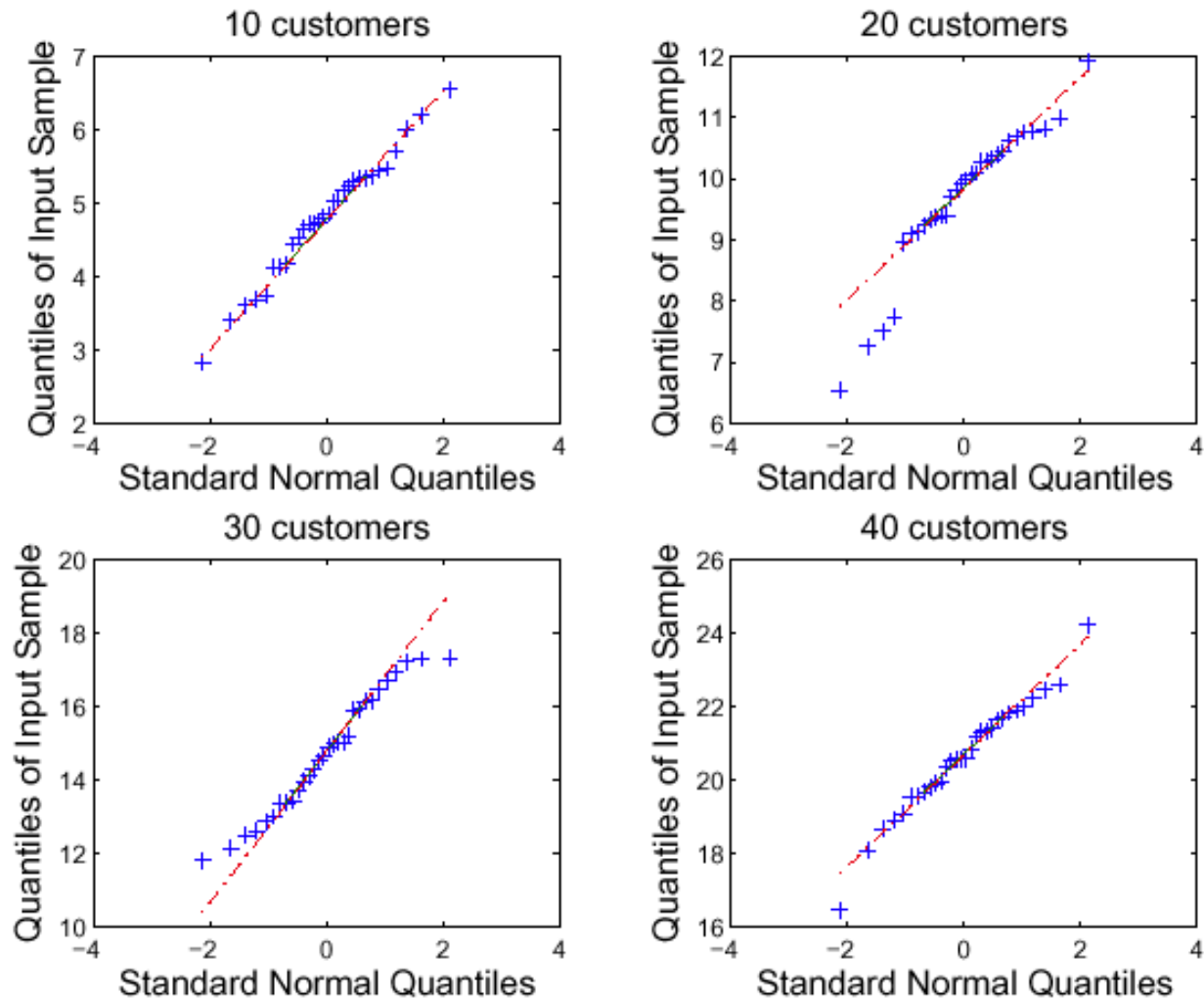
# 3 Accuracy of Simulation Output

- A stochastic simulation produces a random output, we need **confidence intervals**
- Method of choice: independent replications
- Remove transients
  - ▶ For non terminating simulations
- Be careful to have **truly random seeds**
  - ▶ Suppress any dependency between seeds
  - ▶ Ex: **use computer time as seed**

# Results of 30 Independent Replications



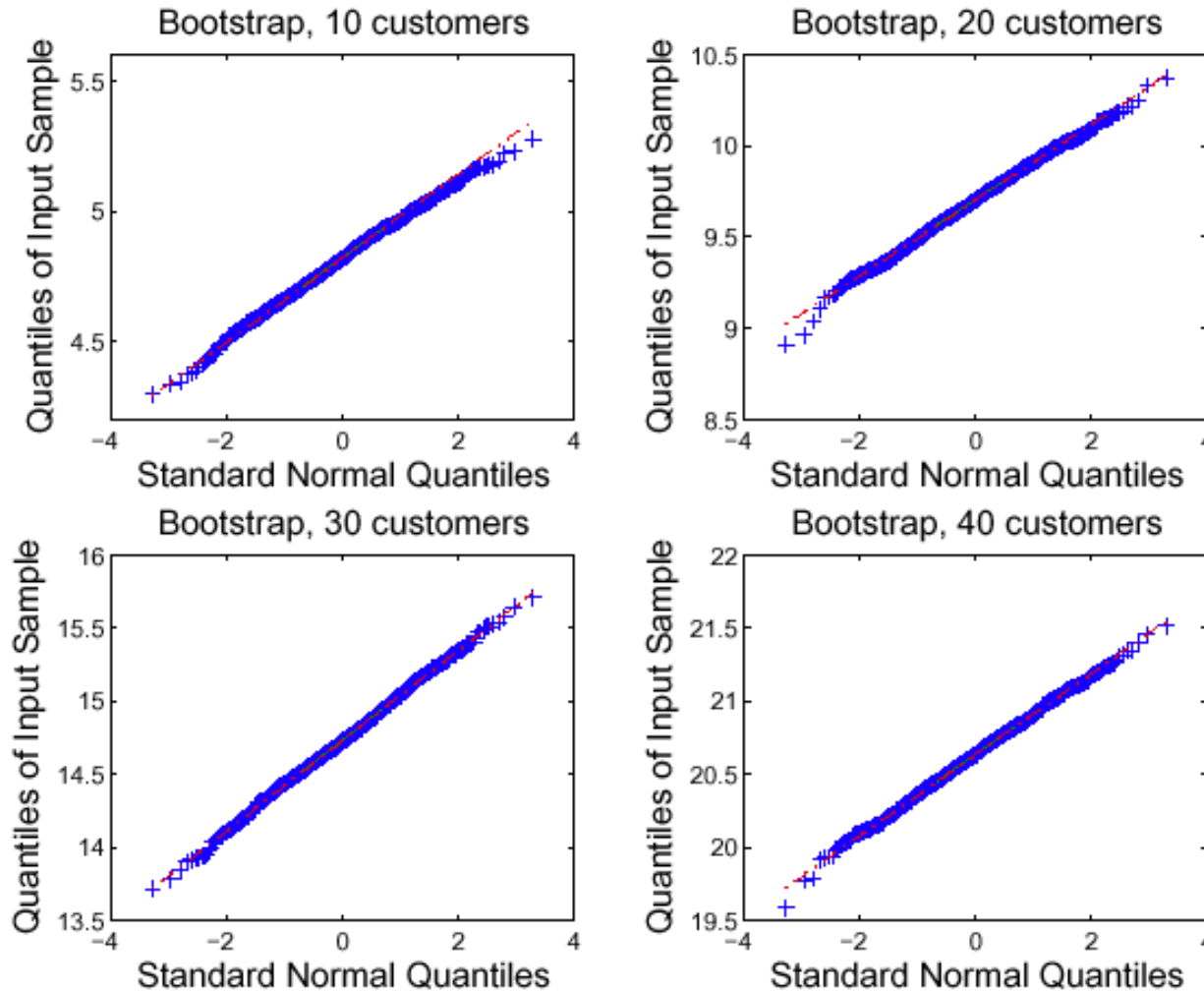
# Do They Look Normal ?



# Bootstrap Replicates

## General Bootstrap Method:

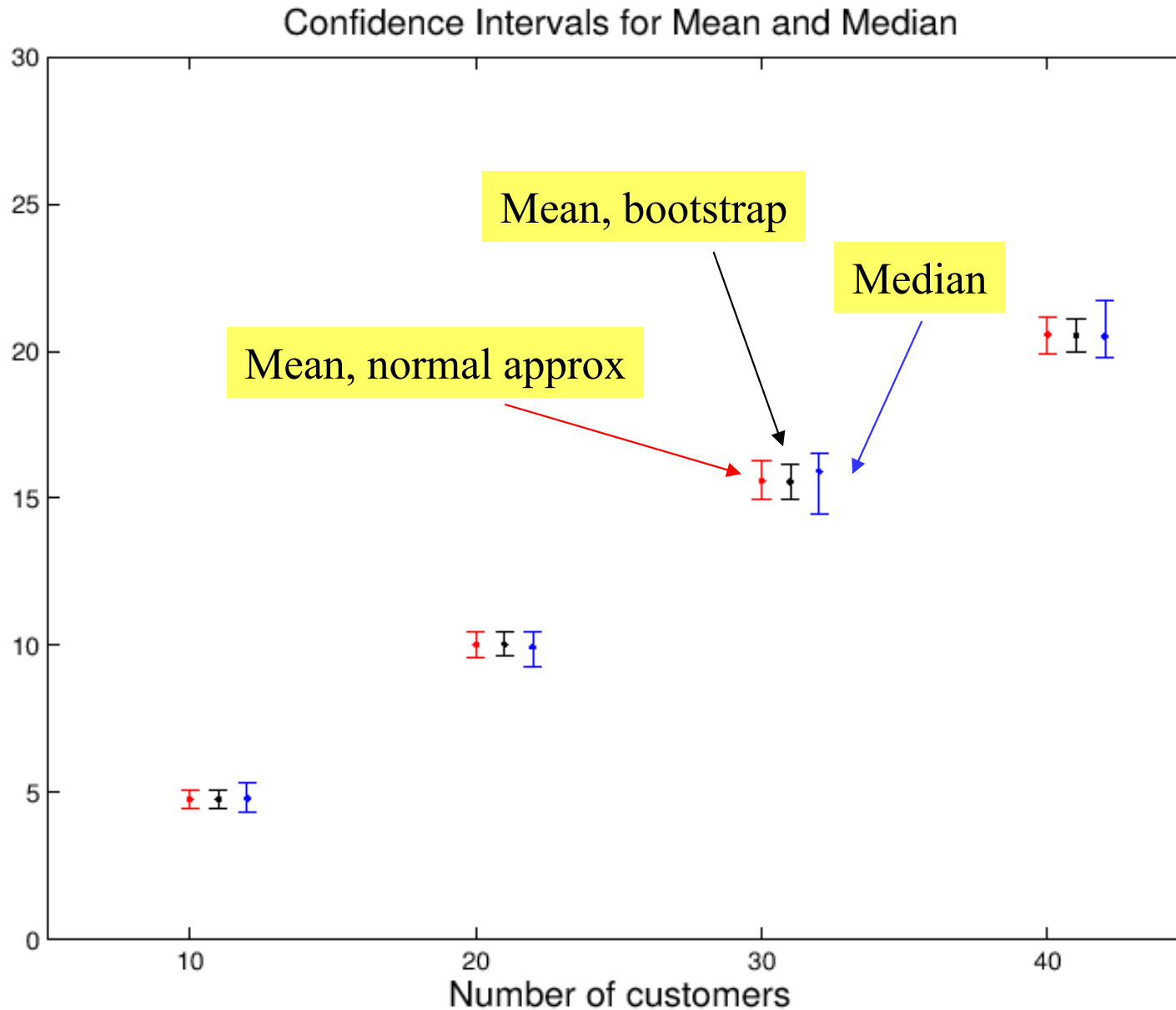
Replicates are computed based on sampling with replacement as in **Bootstrap Percentile Method** in Chapter 2. To compute prediction intervals of those replicates, various methods including **Percentile** can be used.



## **When to use?:**

To see if an estimate (e.g., mean) is normal, rather than the raw data samples themselves.

# Confidence Intervals



**“Bootstrap” turns out to be slightly superior to normal approximation case.**

# 5 Random Number Generator

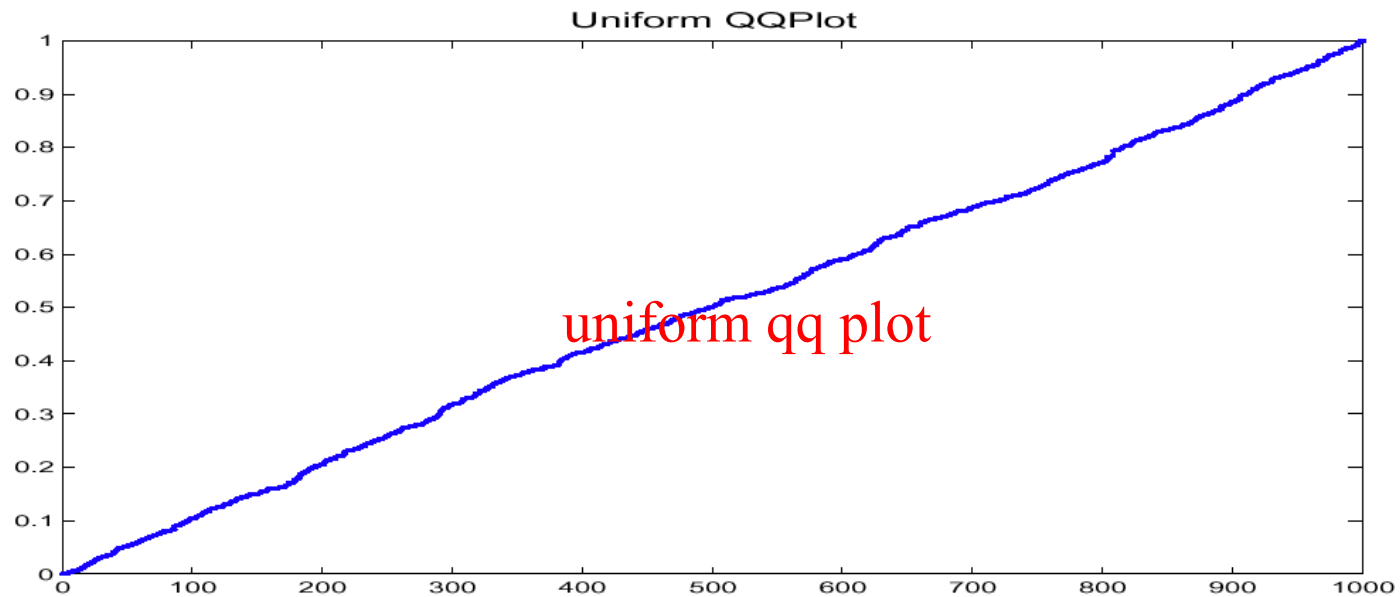
- A stochastic simulation does not use truly random numbers but pseudo-random numbers
  - ▶ Produces a random number »  $U(0,1)$
  - ▶ Example (obsolete but commonly used, e.g. the default one in ns2)

---

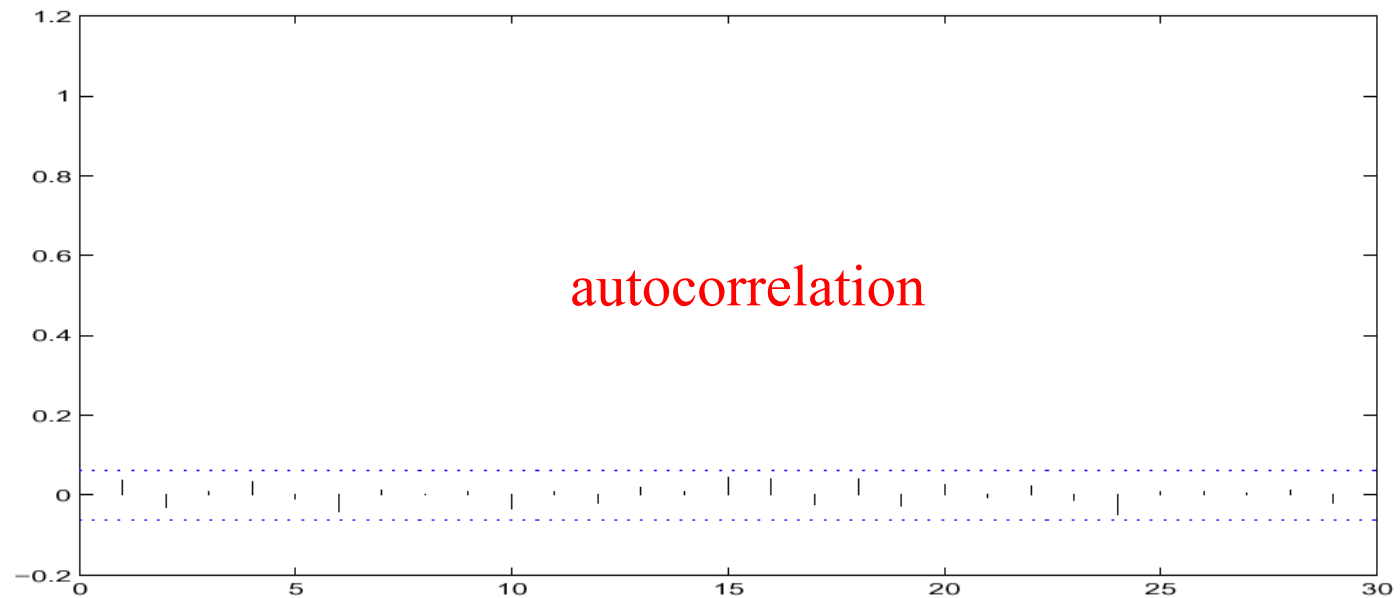
EXAMPLE 6.8: **LINEAR CONGRUENCE.** A widespread generator (for example the default in ns2) has  $a = 16'807$  and  $m = 2^{31} - 1$ . The sequence is  $x_n = \frac{sa^n \bmod m}{m}$  where  $s$  is the seed.  $m$  is a prime number, and the smallest exponent  $h$  such that  $a^h = 1 \bmod m$  is  $m - 1$ . It follows that for any value of the seed  $s$ , the period of  $x_n$  is exactly  $m - 1$ . Figure 6.5 shows that the sequence  $x_n$  indeed looks random.

- ▶ Output appears to be random (see next slides)

# The Linear Congruential Generator of ns2



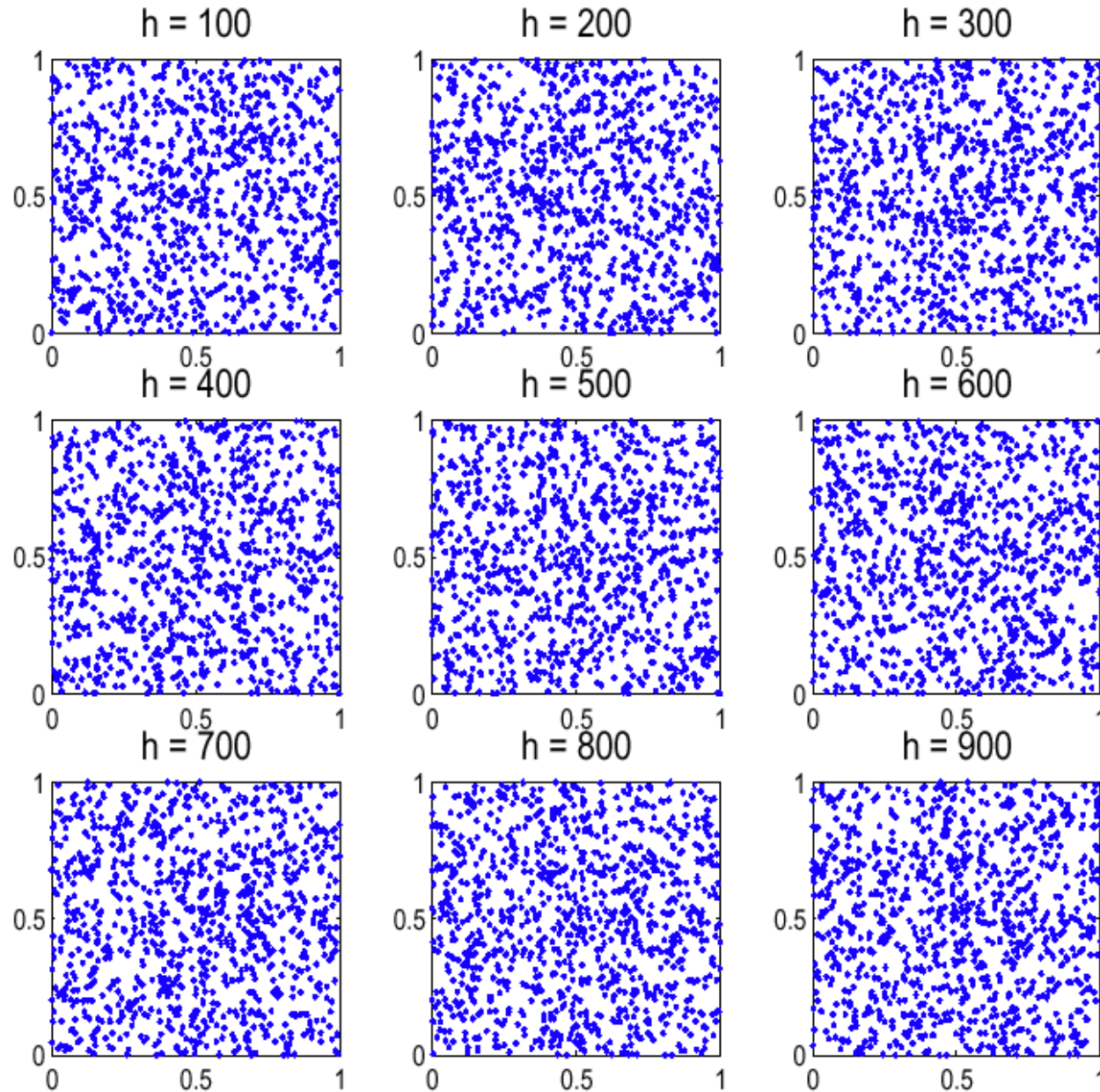
(a)





# Lag Diagram, 1000 points

**Lag Diagram: Scatterplot of data pair  $(X_i, X_{i+h})$**



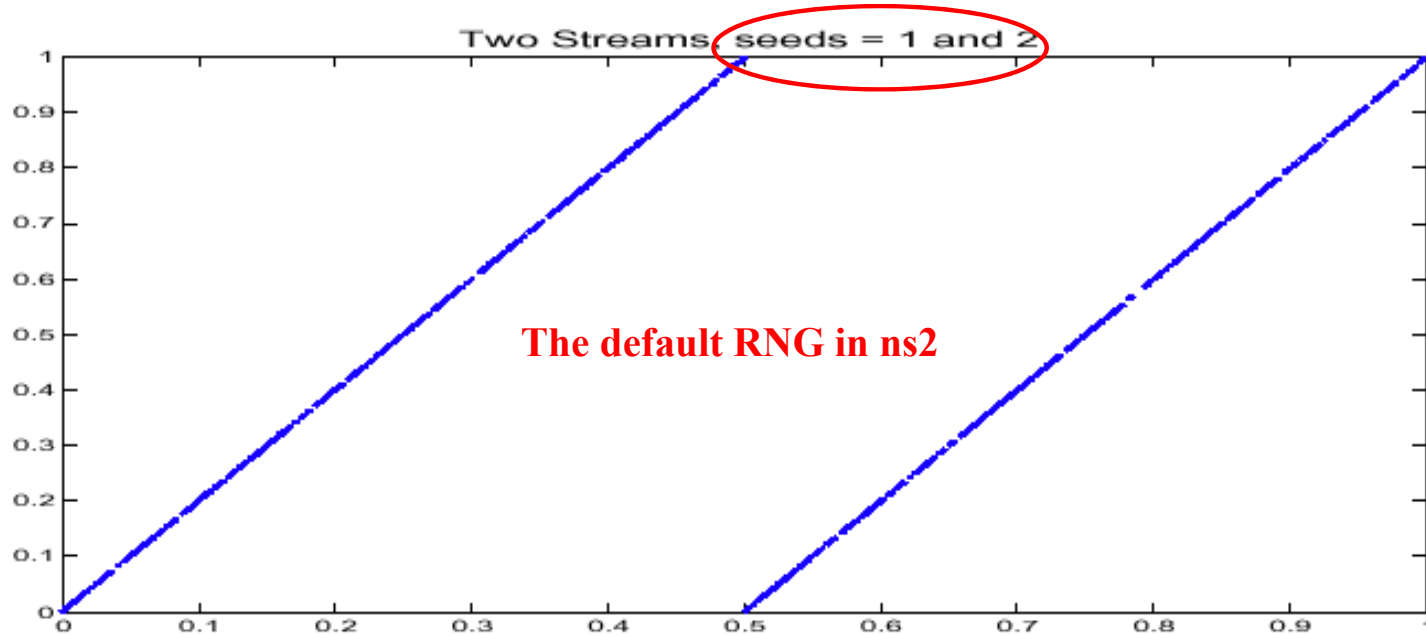
**How *uniformly* the pair of random numbers are scattered.**

# Period of RNG

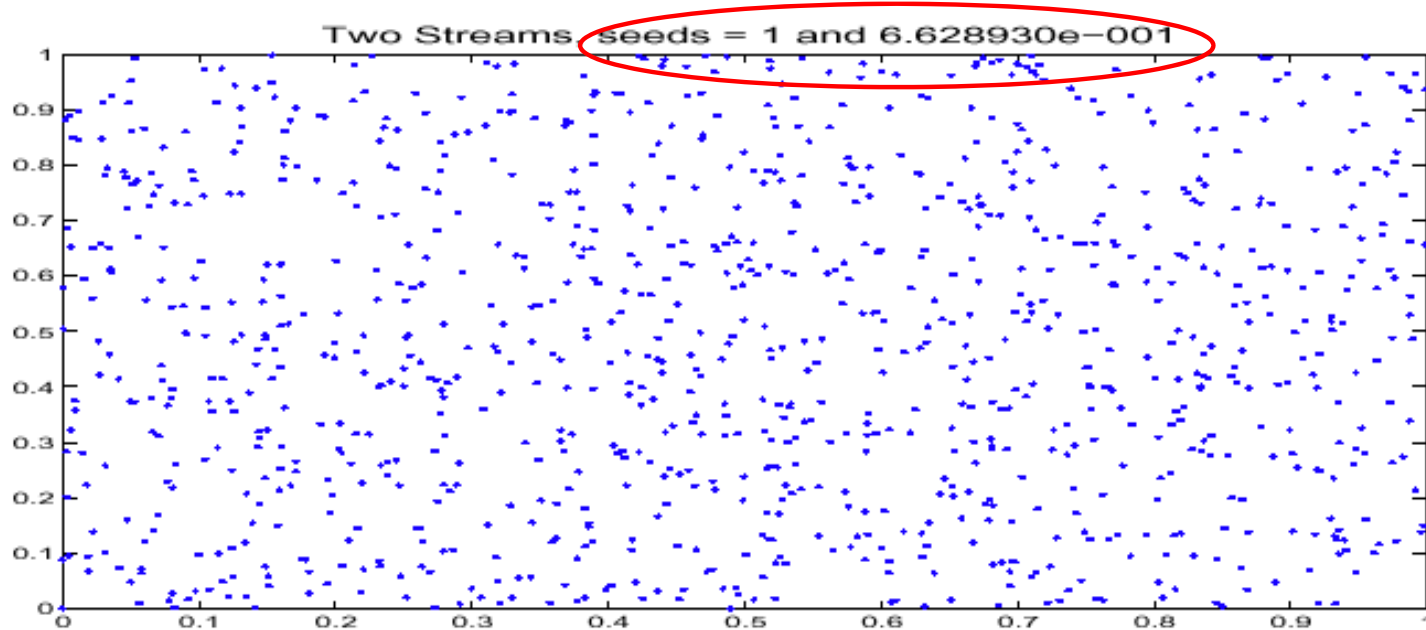
- RNG is in fact **periodic**
  - ▶ Period of which is formidably large
  - ▶ Period should be much larger than maximum number of **uses**

The period of a random number generator should be much smaller than the number of times it is called in a simulation. The generator in Example 6.8 has a period of ca.  $2 \times 10^9$ , which may be too small for very large simulations. There are other generators with much longer periods, for example the “Mersenne Twister” [67] with a period of  $2^{19937} - 1$ . They use other chaotic sequences and combinations of them.

# Two Parallel Streams with too simple a RNG

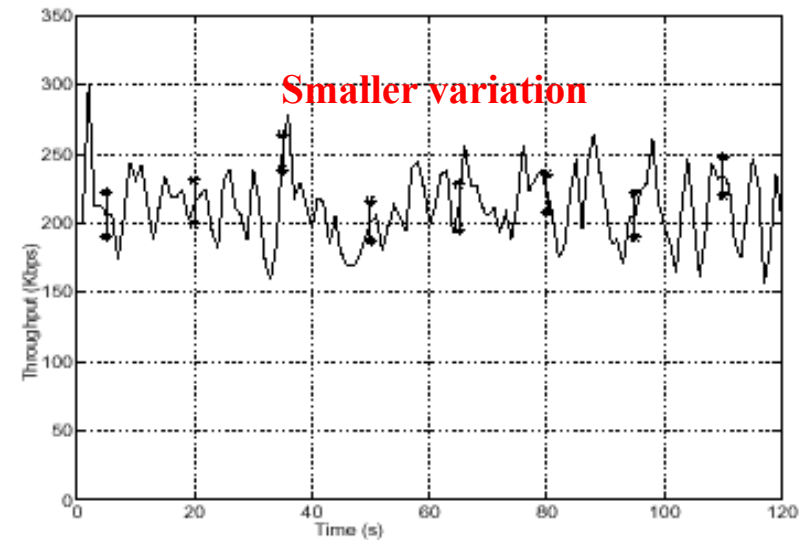
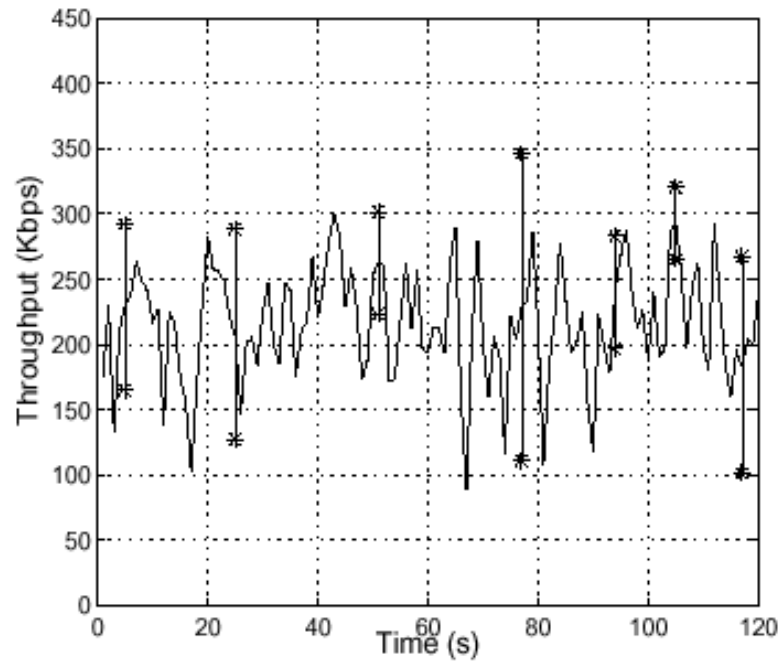


(a)



# Impact of RNG

## Throughput of TCP connections over a wireless ad-hoc network



(a) Linear Congruence with  $a = 16'807$  and  $m = 2^{31} - 1$  and (b) L'Ecuyer's generator[LecuyerSimConf-01]

# Take Home Message

- Be careful to have **a RNG that has a period orders of magnitude larger than what you will ever use in the simulation**
- **Serialize** the use of the RNG rather than parallel streams
  - ▶ Reuse only one seed again and again
    - ▶ to prevent any possible correlation between different streams of RNG.

# 6 Sampling From A Distribution

- Problem:
  - ▶ Given a distribution  $F(\cdot)$ , and a (uniform) RNG, produce some sample  $X$  that follows this distribution
- A common task in simulation
- MATLAB does it for us most of the time, but not always
- Two generic methods
  - ▶ CDF inversion
  - ▶ Rejection sampling

# CDF Inversion

- Applies to real or integer valued RV
- **The general theory**

THEOREM 6.6.1. *Let  $F$  be the CDF of a random variable  $X$  with values in  $\mathbb{R}$ . Define the **pseudo-inverse**,  $F^{-1}$  of  $F$  by*

$$F^{-1}(p) = \sup\{x : F(x) \leq p\}$$

*Let  $U$  be a sample of a random variable with uniform distribution on  $(0, 1)$ ;  $F^{-1}(U)$  is a sample of  $X$ .*

**Twisting a uniform distribution to an arbitrary one**

EXAMPLE 6.10: **EXPONENTIAL RANDOM VARIABLE.** The CDF of the *exponential distribution* with parameter  $\lambda$  is  $F(x) = 1 - e^{-\lambda x}$ . The pseudo-inverse (which in this case is the plain inverse) is obtained by solving the equation

$$1 - e^{-\lambda x} = p$$

where  $x$  is the unknown. The solution is  $x = -\frac{\ln(1-p)}{\lambda}$ . Thus a sample  $X$  of the exponential distribution is obtained by letting  $X = -\frac{\ln(1-U)}{\lambda}$ , or, since  $U$  and  $1 - U$  have the same distribution:

$$X = -\frac{\ln(U)}{\lambda} \tag{6.9}$$

where  $U$  is the output of the random number generator.



# Example: integer valued RV

Thus, an integer valued random variable  $N$  can be sampled by:  $N =$  the index  $n$  such that

**With discrete CDF  $F(n) = \sum_{k=0}^n p_k$**

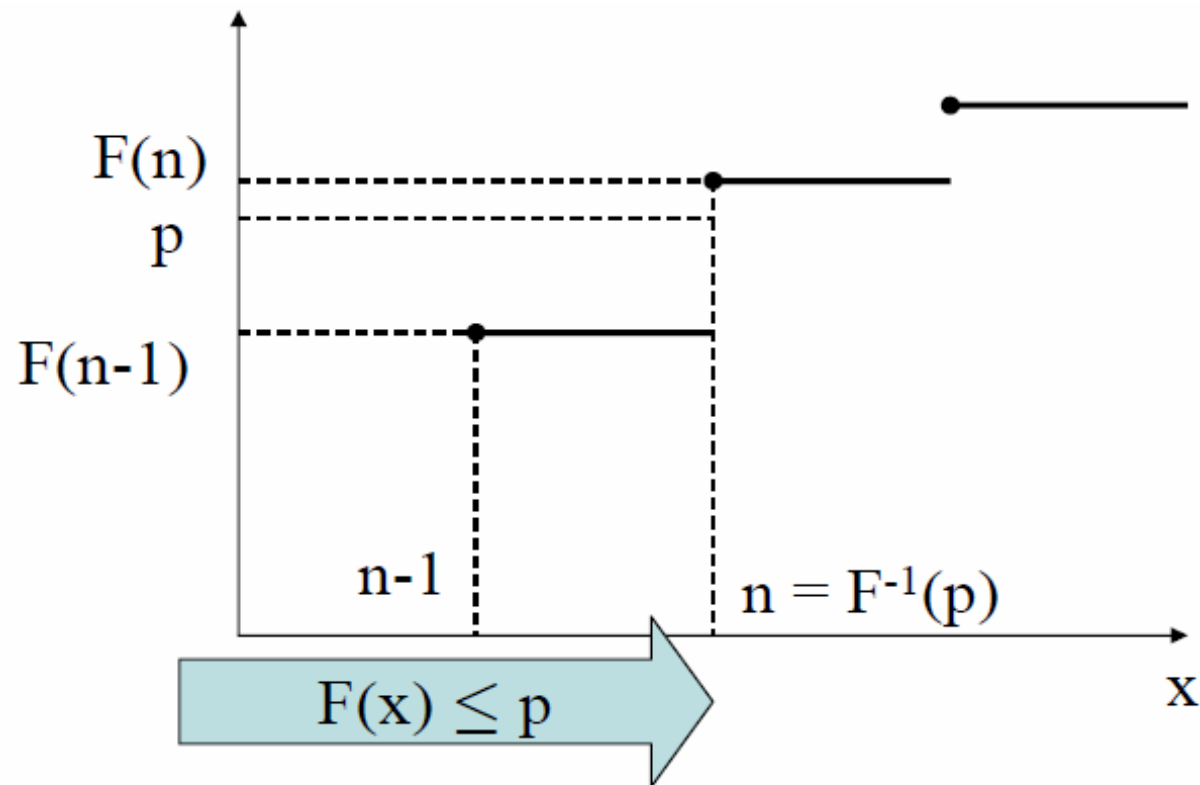


Figure 6.8: Pseudo-Inverse of CDF  $F()$  of an integer-valued random variable  $F(n - 1) \leq U < F(n)$ , where  $U$  is the output of the random generator.

Integer Case:  $F^{-1}(p) = n \Leftrightarrow F(n - 1) \leq p < F(n)$

EXAMPLE 6.11: **GEOMETRIC RANDOM VARIABLE.** Here  $X$  takes integer values  $0, 1, 2, \dots$ . The *geometric distribution* with parameter  $\theta$  satisfies  $\mathbb{P}(X = k) = \theta(1 - \theta)^k$ , thus for  $n \in \mathbb{N}$ :

$$F(n) = \sum_{k=0}^n \theta(1 - \theta)^k = 1 - (1 - \theta)^{n+1}$$

**Plug  $F(n)$  into  
this formula**

by application of Eq.(6.10):  $F^{-1}(p) = n \Leftrightarrow F(n - 1) \leq p < F(n)$

$$F^{-1}(p) = n \Leftrightarrow n \leq \frac{\ln(1 - p)}{\ln(1 - \theta)} < n + 1$$

**The very definition of floor**

hence

$$F^{-1}(p) = \left\lfloor \frac{\ln(1 - p)}{\ln(1 - \theta)} \right\rfloor$$

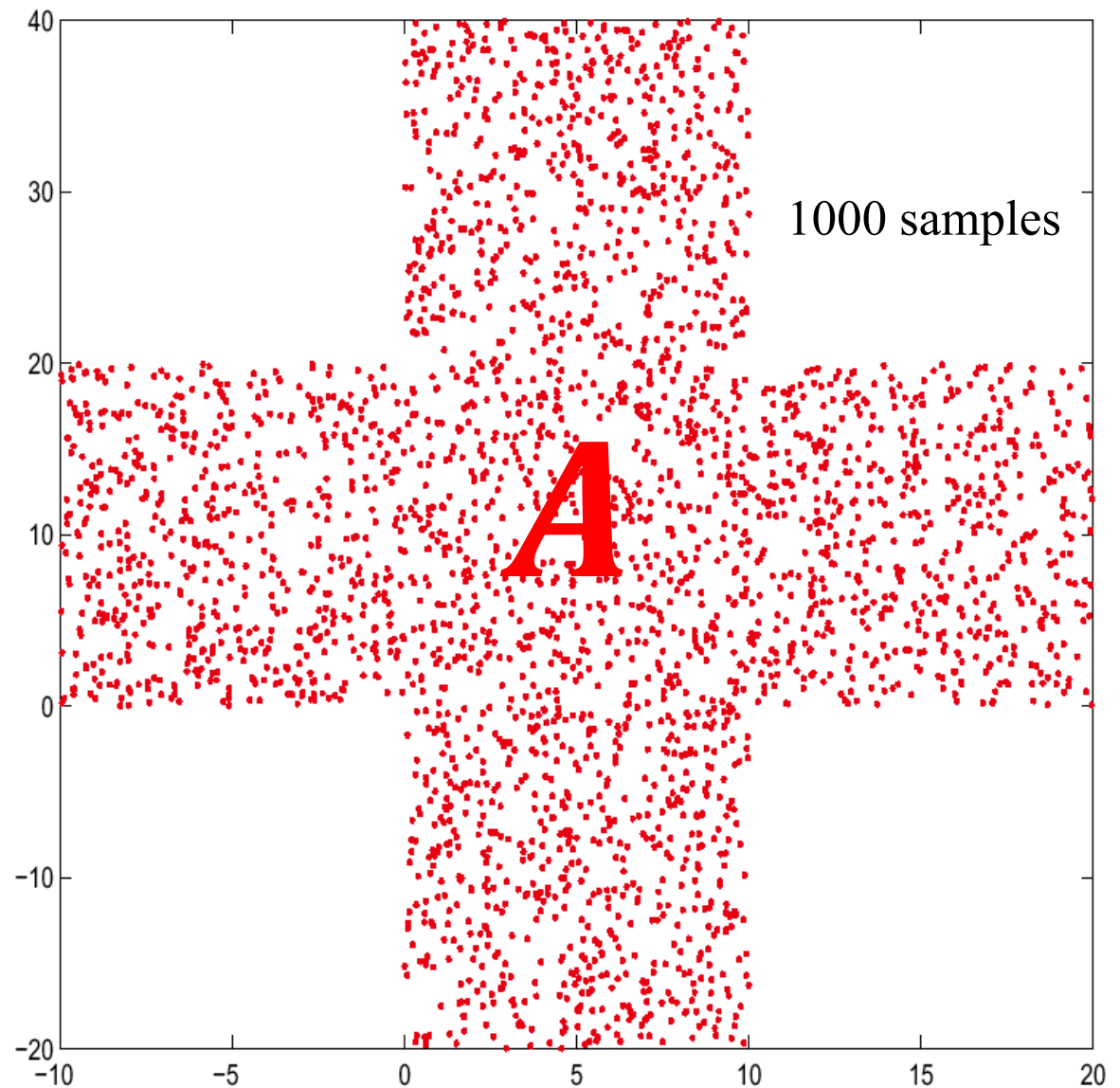
and, since  $U$  and  $1 - U$  have the same distribution, a sample  $X$  of the geometric distribution is

$$X = \left\lfloor \frac{\ln(U)}{\ln(1 - \theta)} \right\rfloor \tag{6.11}$$

# Rejection Sampling

- Applies more generally, also to joint n-dimensional distributions
- Example 1: conditional distribution on this area
- Step 1 :
  - ▶ Can you sample a point uniformly in the bounding rectangle ?
- Step 2 :
  - ▶ How can you go from there to a uniform sample inside the non convex area ?

**Just reject those samples that do not fall in  $A$**



---

<sup>10</sup>The coordinates are independent and uniform: generate two independent samples  $U, V \sim \text{Unif}(0, 1)$ ; the sample is  $((1 - U)x_{\min} + Ux_{\max}, (1 - V)y_{\min} + Vy_{\max})$ .

# Rejection Sampling for Conditional Distribution

- This is the main idea

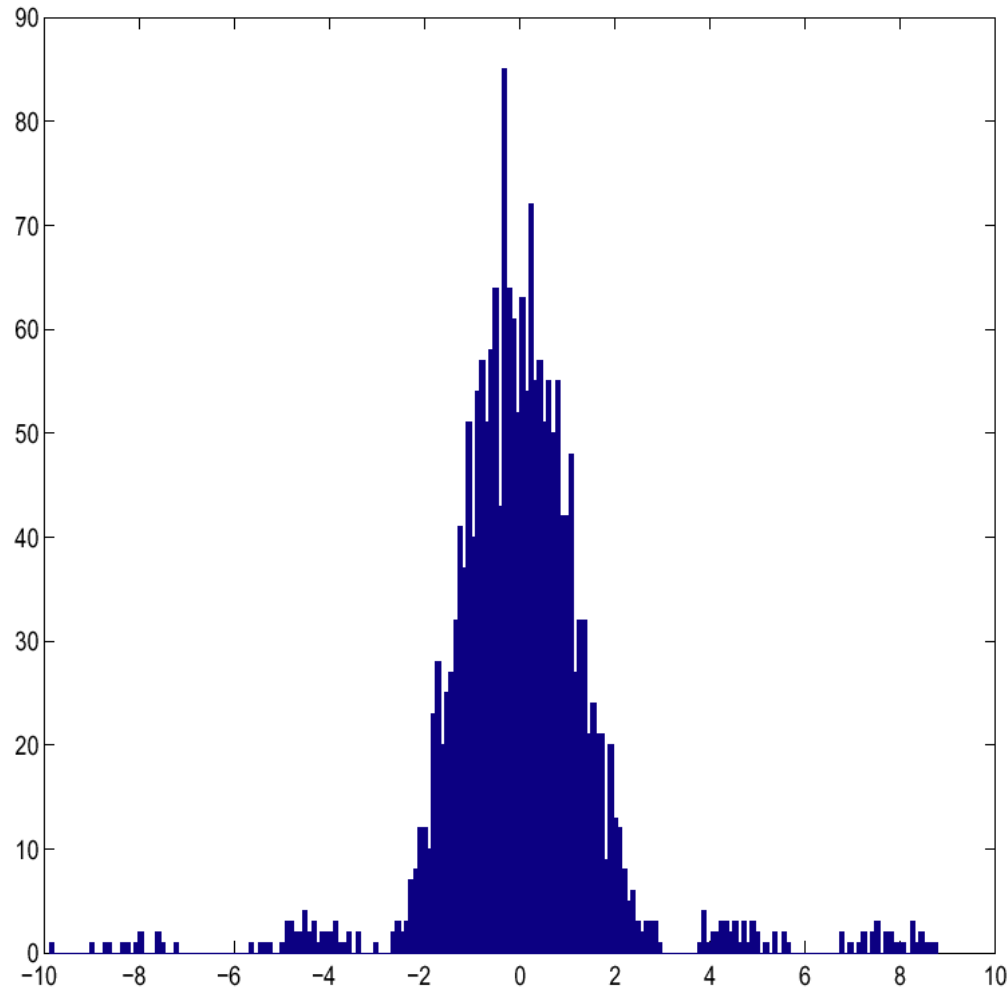
**THEOREM 6.6.2** (Rejection Sampling for a Conditional Distribution). *Let  $X$  be a random variable in some space  $S$  such that the distribution of  $X$  is the conditional distribution of  $\tilde{X}$  given that  $\tilde{Y} \in \mathcal{A}$ , where  $(\tilde{X}, \tilde{Y})$  is a random variable in  $S \times S'$  and  $\mathcal{A}$  is a measurable subset of  $S'$ . A sample of  $X$  is obtained by the following algorithm:*

```
do
    draw a sample of  $(\tilde{X}, \tilde{Y})$ 
until  $\tilde{Y} \in \mathcal{A}$ 
return  $(\tilde{X})$ 
```

*The expected number of iterations of the algorithm is  $\frac{1}{\mathbb{P}(\tilde{Y} \in \mathcal{A})}$ .*

- How can you apply this to the example in the previous slide ?

# A Sample from a Weird Distribution



(a)

Figure 6.10: (a) Empirical histogram (bin size = 10) of 2000 samples of the distribution with density  $f_X(x)$  proportional to  $\frac{\sin^2(x)}{x^2} \mathbf{1}_{\{-a \leq y \leq a\}}$  with  $a = 10$ .

**It's tricky to compute pseudo-inverse  $F^{-1}(\cdot)$ !**

# Rejection Sampling for General Distributions

$f(\cdot) = f_X(\cdot) \times f_Y(\cdot)$  where  $f_Y(\cdot)$  is defined over a compact area up to a proportionality constant.

THEOREM 6.6.3 (Rejection Sampling for Distribution with Density). Consider two random variables  $X, Y$  with values in the same space, that both have densities. Assume that:

- we know a method to draw a sample of  $X$
- the density of  $Y$  is known up to a normalization constant  $K$ :  $f_Y(y) = K f_Y^n(y)$ , where  $f_Y^n$  is a known function
- there exist some  $c > 0$  such that

$$\frac{f_Y^n(x)}{f_X(x)} \leq c$$

A sample of  $Y$  is obtained by the following algorithm:

```
do
    draw independent samples of  $X$  and  $U$ , where  $U \sim \text{Unif}(0, c)$ 
until  $U \leq \frac{f_Y^n(X)}{f_X(X)}$ 
return( $X$ )
```

The expected number of iterations of the algorithm is  $\frac{c}{K}$ .

**ARBITRARY DISTRIBUTION WITH DENSITY** Assume that we want a sample of  $Y$ , which takes values in the bounded interval  $[a, b]$  and has a density  $f_Y = K f_Y^n(y)$ . Assume that  $f_Y^n(y)$  (non normalized density) can easily be computed, but not the normalization constant  $K$  which is unknown. Also assume that we know an upper bound  $M$  on  $f_Y^n$ .

We take  $X$  uniformly distributed over  $[a, b]$  and obtain the sampling method:

```

do
    draw  $X \sim \text{Unif}(a, b)$  and  $U \sim \text{Unif}(0, M)$ 
until  $\underline{U} \leq \underline{f_Y^n(X)}$ 
return( $X$ )

```

**Both  $U$  and  $X$  are random.**

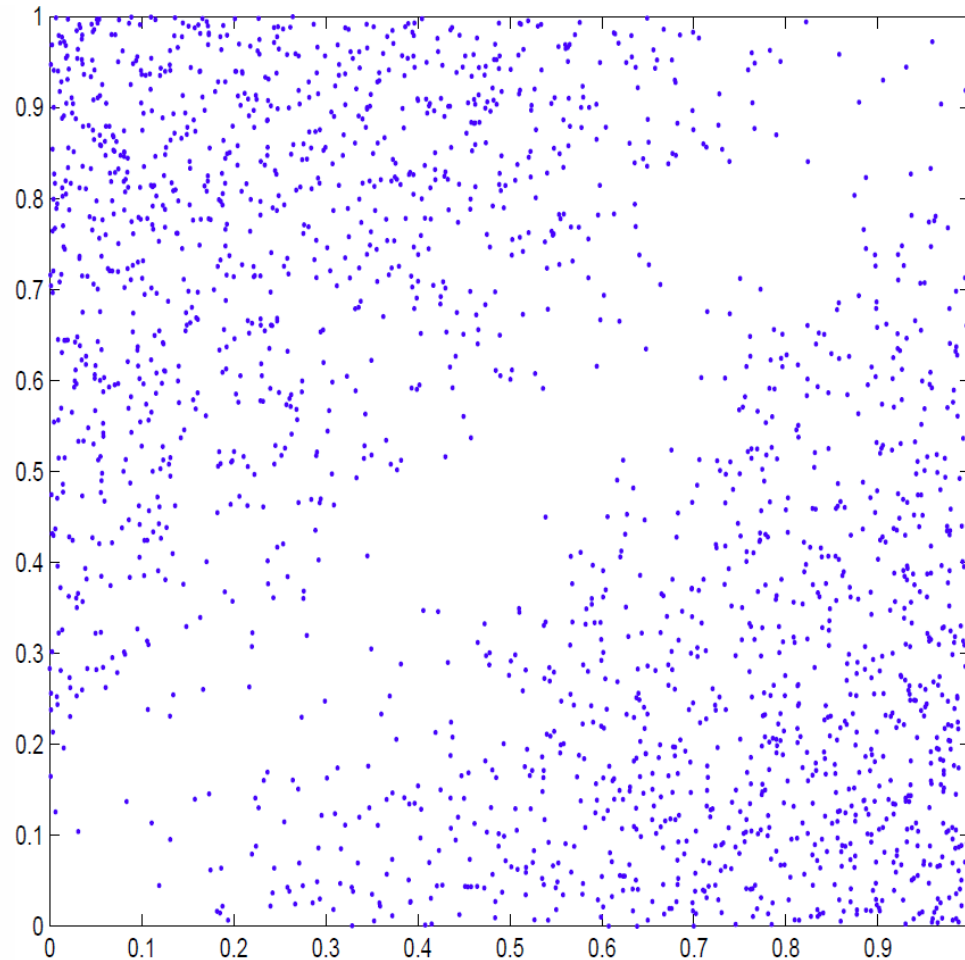
Note that we do *not* need to know the multiplicative constant  $K$ . For example, consider the distribution with density

$$f_Y(y) = K \frac{\sin^2(y)}{y^2} \mathbf{1}_{\{-a \leq y \leq a\}} \quad (6.13)$$

$K$  is hard to compute, but a bound  $M$  on  $f_Y^n$  is easy to find ( $M = 1$ ) (Figure 6.10).

# Another Sample from a Weird Distribution

Density  $\propto |X_1 - X_2|$  in a unit area



(b)

Figure 6.10:(b) 2000 independent samples of the distribution on the rectangle with density  $f_{X_1, X_2}(x_1, x_2)$  proportional to  $|x_1 - x_2|$ .



**EXAMPLE 6.13: A STOCHASTIC GEOMETRY EXAMPLE.** We want to sample the random vector  $(X_1, X_2)$  that takes values in the rectangle  $[0, 1] \times [0, 1]$  and whose distribution has a density proportional to  $|X_1 - X_2|$ . We take  $f_X$  = the uniform density over  $[0, 1] \times [0, 1]$  and  $f_Y^n(x_1, x_2) = |x_1 - x_2|$ . An upper bound on the ratio  $\frac{f_Y^n(x_1, x_2)}{f_X(x_1, x_2)}$  is 1. The sampling algorithm is thus:

```
do
    draw  $X_1, X_2$  and  $U \sim \text{Unif}(0, 1)$ 
until  $U \leq |X_1 - X_2|$ 
return  $(X_1, X_2)$ 
```

Figure 6.10 shows an example. Note that there is no need to know the normalizing constant to apply the sampling algorithm.

## 6.3 Ad-Hoc Methods

- Optimized methods exist for some common distributions
  - ▶ Optimization = reduce computing time
- If implemented in your tool, use them !
- Example: simulating a normal distribution
  - ▶ **Inversion method** is not simple
    - ▶ normal CDF is complicated  $\rightarrow$  no closed form for  $F^{-1}(\cdot)$
  - ▶ **Rejection method** is impossible
  - ▶ But a more efficient method exists, for drawing jointly **2** independent normal RV
- There are also ad-hoc methods for  $n$ -dimensional normal distributions (Gaussian vectors)

# Sample from *correlated* 2d-Normal Vector

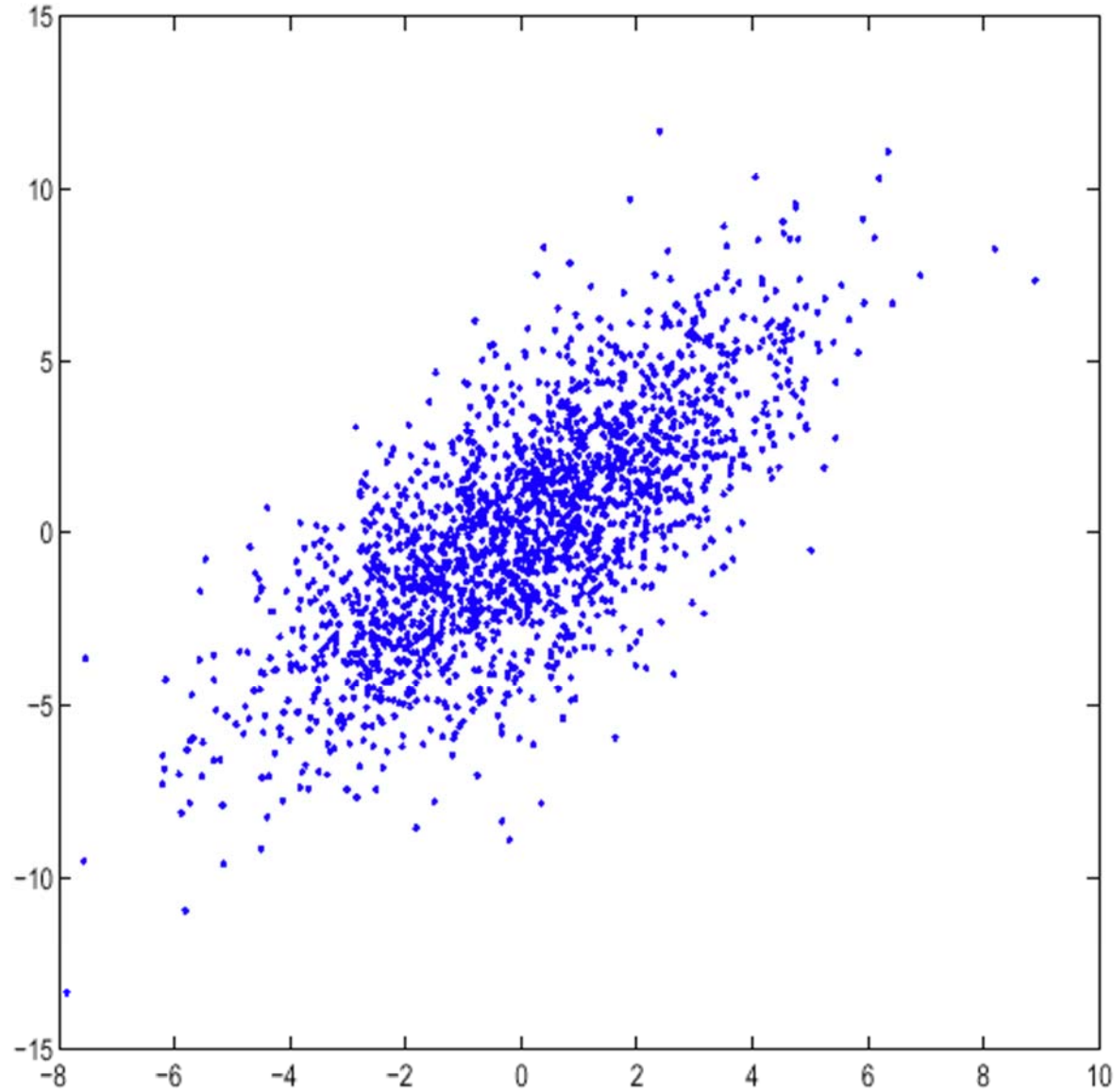


Figure 3.11: 1000 independent samples of the normal vector  $X_1, X_2$  with 0 mean and covariance  $\Omega_{1,1} = \sigma_1^2 = 5$ ,  $\Omega_{1,2} = \Omega_{2,1} = 5$ ,  $\Omega_{2,2} = \sigma_2^2 = 10$ . It is obtained by the transformation  $X = LY$  with  $Y_{iid} \sim N_{0,1}$  and  $L = \sqrt{5}(1, 0; 1/2, \sqrt{3}/2)$ .

Choleski's factorization  $\Omega = LL^T$  is used for 2D normal

# 4 Monte Carlo Simulation

- A simple method to compute integrals of all kinds
- Idea: interpret the integral as  $\beta = \mathbb{E}(\varphi(\vec{X}))$ .
- Assume you can *simulate* as many independent samples of  $X$  as you want

Monte-Carlo simulation consists in generating  $R$  i.i.d. replicates  $\vec{X}^r, r = 1, \dots, R$ . The Monte-Carlo estimate of  $\beta$  is

**Vector**

$$\hat{\beta} = \frac{1}{R} \sum_{r=1}^R \varphi(\vec{X}^r) \quad (6.2)$$

A confidence interval for  $\beta$  can then be computed using the methods in Chapter 2 for a confidence interval for the mean. By adjusting  $R$ , the number of replications, we can control the accuracy of the method, i.e. the width of the confidence interval.

**In Ch. 4, if  $p$ -value is small ( $<0.05$ ), i.e., sample data is abnormal, null hypothesis is rejected.**

This example “estimates”  $p$ -value of a goodness of (distribution) fit test. In the example, all experiments (i.e.,  $n_i$  are known) are finished and you want to test whether the sample data follows a hypothesized distribution.

EXAMPLE 6.7:  $p$ -VALUE OF A TEST. Let  $X_1, \dots, X_n$  be a sequence of i.i.d. random variables that take values in the discrete set  $\{1, 2, \dots, I\}$ . Let  $q_i = \mathbb{P}(X_k = i)$ . Let  $N_i = \sum_{k=1}^n \mathbf{1}_{\{X_k=i\}}$  (number of observation that are equal to  $i$ ). Assume we want to compute

**Probability is a kind of expectation.**

$$p = \mathbb{P} \left( \sum_{i=1}^k N_i \ln \frac{N_i}{nq_i} > a \right) \quad (6.3)$$

where  $a > 0$  is given. This computation arises in the theory of goodness of fit tests, when we want to test whether  $X_i$  does indeed come from the model defined above ( $a$  is then equal to  $\sum_{i=1}^k n_i \ln \frac{n_i}{nq_i}$  where  $n_i$  is our data set). For large values of the sample size  $n$  we can approximate  $\beta$  by a  $\chi^2$  distribution (see Section 4.4), but for small values there is no analytic result.

**We have observed  $1, \dots, I$  respectively  $n_1, \dots, n_k$  times such that  $\sum_{i=1}^k n_i = n$ .  
Is it so abnormal that we can't claim they follow multinomial distribution with  $n, q_1, \dots, q_k$ ?**

**To justify the claim, we have to compute this based on samples  $X_i$  ( $N_i$ ) following multinomial distribution with  $n, q_1, \dots, q_k$  (null hypothesis)**

We use Monte-Carlo simulation to compute  $p$ . We generate  $R$  i.i.d. replicates  $X_1^r, \dots, X_n^r$  of the sequence ( $r = 1, \dots, R$ ). This can be done by using the inversion method described in this chapter. For each replicate  $r$ , let

$$N_i^r = \sum_{k=1}^n \mathbf{1}_{\{X_k^r=i\}} \quad (6.4)$$

The Monte Carlo estimate of  $p$  is

$$\hat{p} = \frac{1}{R} \sum_{r=1}^R \mathbf{1}_{\{\sum_{i=1}^k N_i \ln \frac{N_i}{nq_i} > a\}} \quad (6.5)$$

**Obviously,  $p$ -value can be regarded as a success probability in Theorem 2.2.4.**

Assuming that  $\hat{p}R \geq 6$ , we compute a confidence interval by using the normal approximation in Eq.(2.29). The sample variance is estimated by **of the  $p$ -value**

$$\hat{\sigma} = \sqrt{\frac{\hat{p}(1 - \hat{p})}{R}} \quad (6.6)$$

and a confidence interval at level 0.95 is  $\hat{p} \pm 1.96\hat{\sigma}$ . Assume we want a relative accuracy at least equal to some fixed value  $\epsilon$  (for example  $\epsilon = 0.05$ ). This is achieved if **The ratio of std to mean**

$$\frac{1.96\hat{\sigma}}{\hat{p}} \leq \epsilon \quad (6.7)$$

which is equivalent to

$$R \geq \frac{3.92}{\epsilon^2} \left( \frac{1}{\hat{p}} - 1 \right) \quad (6.8)$$

**Rationale behind (6.8):**

**Run the simulation enough times  $R$  such that the confidence interval for  $\hat{p}$  is no greater than  $[\hat{p} - \hat{p} \cdot \epsilon, \hat{p} + \hat{p} \cdot \epsilon]$ .  
→ fine-tuning the error range of the estimate  $\hat{p}$**

We can test for every value of  $R$  whether Eq.(6.8) is verified and stop the simulation when this happens. Table 6.1 shows some results with  $n = 100$  and  $a = 2.4$ ; we see that  $p$  is equal to 0.19 with an accuracy of 5%; the number of Monte Carlo replicates is proportional to the relative accuracy to the power  $-2$ .

$R$	$\hat{p}$	confidence interval margin
30	0.2667	0.1582
60	0.2500	0.1096
120	0.2333	0.0757
240	0.1917	0.0498
480	0.1979	0.0356
960	0.2010	0.0254
1920	0.1865	0.0174
3840	0.1893	0.0124
7680	0.1931	0.0088

**When  $R=7680$ ,  
relative accuracy of  
5% is achieved at last.**

Table 6.1: Computation of  $p$  in Example 6.7 by Monte Carlo simulation. The parameters of the model are  $I = 4$ ,  $q_1 = 9/16$ ,  $q_2 = q_3 = 3/16$ ,  $q_4 = 1/16$ ,  $n = 100$  and  $a = 2.4$ . The table shows the estimate  $\hat{p}$  of  $p$  with its 95% confidence margin versus the number of Monte-Carlo replicates  $R$ . With 7680 replicates the relative accuracy (margin/ $\hat{p}$ ) is below 5%.



# Take Home Message

- Most hard problems relative to computing a probability or an integral can be solved with Monte Carlo
  - ▶ Brainless but why not
  - ▶ Run time may be large -> importance sampling techniques (Chapter 6.7)

# Conclusion

- Simulating well requires knowing the concepts of
  - ▶ Transience
  - ▶ Confidence intervals
  - ▶ Sampling methods