Introduction to network simulation

M. Marot(1)

Michel.Marot@telecom-sudparis.eu

(1) Institut Mines-Télécoms/Télécom SudParis

Outline

- Different types of simulations:
  - Step by step;
  - Event driven;
  - Fluid.
- Validation of a simulation: Le Gall's method.
- Random number generation:
  - Inversion method;
  - Congruential method.

Different types of simulations

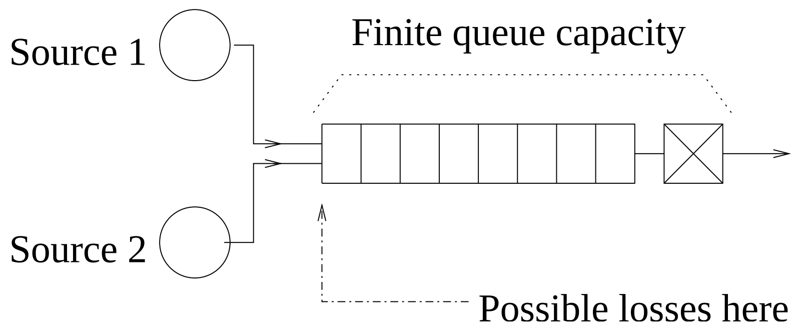
Discrete event simulation

- **Definition**: simulation of phenomena whose evolution is not continuous in time.
- **Example**: simulation of the exchange of the TCP signalisation and TCP segments:
  - A sends a SYN at t_0;
  - B sends SYN+ACK at t_1;
  - A sends ACK at t_2;
  - etc.

  $=>$ All these events are **discrete events**.
- The discrete event simulation is the opposite of the fluid simulation where the evolution of the system is continuous during time. E.g.: simulation of a fluid flow.
- Two kinds of discrete event simulations:
  - synchronous approach;
  - asynchronous approach.

Discrete event simulation: synchronous approach

- The **step by step simulation** consists in making evolve time by constant steps.
- **Example**: simulation of a finite capacity queue with two input Bernoulli sources, in order to study the loss rate:



- Let us assume:
  - A synchronous network carrying fixed size packets;
  - Service time = 1 time unit;
  - The emission rate (probability of having a cell emitted in a time unit) is:
    - $p[1]$ for the 1st source;
    - $p[2]$ for the 2ond source;
  - The maximal duration of the simulation is TMAX;
  - A function drand48() exists, which returns a random value uniformly distributed between 0 an 1.

Discrete event simulation: synchronous approach

```
$Nblosses := 0$
$Nbsent := 0;$
$size := 0;$
**for** t **from** 1 **to** TMAX **do**
   **if** $size > 0$ **then**
      $size := size - 1;$
      {The element in the head of the queue is serviced}
   {The queue is now filled}
   **for** i **from** 1 **to** 2 **do**
      **if** drand48()$\leq p[i]$ **then**
         $Nbsent := Nbsent + 1;$
         **if** size = MaxSize **then**
            $Nblosses := Nblosses + 1;$
         **else**
            $size := size + 1;$
print(”The global loss rate is:”, Nblosses/Nbsent);
```

Discrete event simulation: synchronous approach

- Remark 1:
  - The first source is systematically processed before the second one, it should then experience less losses than the second source.
  - For fairness reasons, one could process at each step of the loop either the first or the second source in a random fashion.
- Remark 2:
  - If $p[1]$ and $p[2]$ are very small, this program is very complex because, in this case, most of the time it runs only tests, without any instruction. That is, most of the time it does simulate nothing !
  - Actually, most of the processings executed by the simulation program consist in simulating an empty system, which presents no interest at all since there is no loss when the system is empty.

$=>$ Problem of the step by step simulation: every moment of time is simulated, periodically, even if nothing occurs.

BUT The main interest of the step by step simulation: simple to implement (no need for schedulers)!

Discrete event simulation: asynchronous approach

- The **asynchronous approach**, or **event driven simulation**, consists in making evolve the time only when there is an interesting event.
- Since, in the previous example, the times when the queue is empty present no interest, it is only simulated:
  - The end of services after a unit of time;
  - The arrivals from one of both files.
- That supposes, when a source sends a customer, to be able to predict the next departure time.
- In this example, it is possible because if a cell is emitted with probability $p[i]$, the time between two customer emissions is geometrically distributed with parameter $\frac{1}{p[i]}$. It suffices then to draw a random variable geometrically distributed with parameter 1/p to know the next departure date.

Discrete event simulation: asynchronous approach

- **Concept of scheduler**:
  - The events occur in a random order: at a given time, the sequence of the sent cells can alternate a cell from the 1st source, then a cell of the 2ond source, and so on, and then it can change (e.g.: two cells from the 1st source can successively arrive, followed by a cell from the 2ond source).

$=>$ These events can no more be processed in a loop as previously.

  - Each time an event is planned (e.g.: when a source sends a cell, it plannes the departure of the next cell), a tuple (event_date, event_nature) is inserted in a list sorted by increased event dates. The simulation consists then in executing the events of this scheduler in the order of the increasing times.

Discrete event simulation: synchronous approach

- **Algo. event driven simulation:**

```
$Nblosses := 0$;
$Nbsent := 0$;
$e1 := malloc(event)$;
$e2 := malloc(event)$;
$e1.date := 0$
$e1.type := arrival\_source1$
$e2.date := 0$
$e2.type := arrival\_source2$
Insert_event(scheduler, e1);
Insert_event(scheduler, e2);
**while** $t < TMAX$ **do**
  $Evt := extract\_event(scheduler)$;
  $t := Evt.date$;
  **if** $Evt.type = departure$ **then**
    procedure_departure
  **else if** $Evt.type = arrival\_source1$ OR $Evt.type = arrival\_source2$ **then**
    procedure_arrival(Evt)
print(”The global loss rate is:”, Nblosses/Nbsent);
```

Discrete event simulation: synchronous approach

- **procedure_arrival:**

```
$Nbsent := Nbsent + 1;$
**if** size = MaxSize **then**
   $Nblosses := Nblosses + 1;$
**else**
   $size := size + 1;$
   **if** $size = 1$ **then**
      $e := malloc(event); e.date := t + 1; e.type := departure;$
      Insert_event(scheduler,e);
$e := malloc(event);$
**if** Evt.type = arrival_source1 **then**
   $e.date := t + GEO\left(\frac{1}{p[1]}\right);$ $\{$GEO(x) returns a 1/x mean geometrically
   distributed random value$\}$
   $e.type := arrival\_source1$
**else**
   $e.date := t + GEO\left(\frac{1}{p[2]}\right);$
   $e.type := arrival\_source2;$
Insert_event(scheduler,e);
```

Discrete event simulation: synchronous approach

- **procedure_departure:**

```
$size := size - 1;$
if $size > 0$ then
  $e := malloc(event);$
  $e.date := t + 1;$ {New service of duration 1 time unit}
  $e.type := departure;$
  Insert_event(scheduler,e);
```

- **Caution:** some performance criteria are estimated by averages of events (e.g.: loss rates), that is by computing means of occurences of events, while others are estimated by temporal means (e.g.: queue lengths). Ex.:

$$L = \int_{t=0}^{TMAX} \frac{L(t)}{T} dt = \sum_{t_i=0}^{TMAX} L(t_i) \times \frac{t_{i+1} - t_i}{T},$$

  where t_i is the date of the i^{th} event.

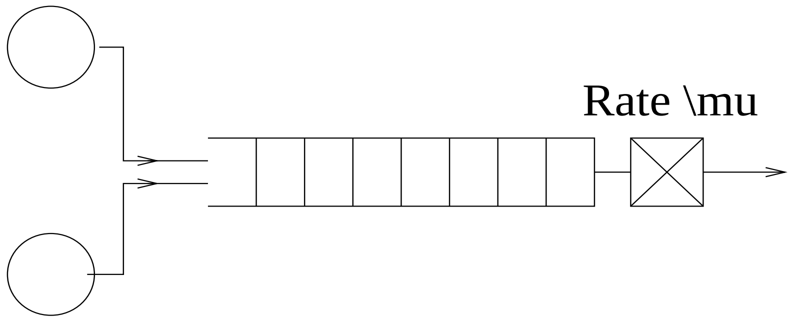
- In discrete event simulation, temporal means must be well computed by weighting the events by their durations.

Fluid simulation

- Let us suppose that we wants to simulate two TCP connections through a finite size B buffer.
- We want to simulate TCP Vegas:
  - Same processings as TCP Reno except:
  - For each received packet,
    - RTT_{min} is updated;
    - $Diff := \lambda_{expected} - \lambda_{real} \qquad \left(= \frac{W}{RTT_{min}} - \frac{W}{RTT}\right)$, actually, $\lambda_{real} = \frac{W}{RTT}$ is exactly the Little's law.
    - **if** $Diff \times RTT_{min} < \alpha$ **then** $W := W + 1$ **endif**;
    - **if** $Diff \times RTT_{min} > \beta$ **then** $W := W - 1$ **endif**.
  - Typically, the orders of magnitude of α and β are $1, 2, 3, \ldots$;
  - If a loss occurs: $ssthresh$ is assigned $\frac{W}{2}$ and W is assigned 1, and the same process is restarted.

Fluid simulation

- **Fluid model of this system:**



  - μ is the rate of the server;
  - τ_i is the latency of the connection i.
  - The connections are characterised by their rates $\lambda(t)$, as if they were fluids.

- $\lambda(\mathbf{t})$**:** The rate of the connection is given by
  - Either the rate of the connection is greater than μ and then $\lambda(t) = \mu$,
  - or it is smaller and then there is not waiting, the response time is thus equal to the latency τ_i and Little gives thus: $\lambda(t) = \frac{W(t)}{\tau}$ This alternative, without any other possible cas, is exactly the fluid assumption.

$$=> \quad \lambda(t) = \min\left(\mu, \frac{W(t)}{\tau}\right)$$

Fluid simulation

- **RTT(t):** Little gives $RTT(\lambda) = \frac{W(t)}{\lambda(t)}$

  $=> \quad RTT(t) = \max\left(\frac{W(t)}{\mu}, \tau\right)$

- **W(t):** the evolution of the congestion window is given by the behaviour of TCP Vegas:

  $$\begin{cases} W(t) = W_{max} \Rightarrow W(t^+) = \gamma W(t) \qquad \text{(the transient} \\ \text{regime of the convergence from W=1 to W=sstresh is ignored)} \\ \dfrac{dW(t)}{dt} = \dfrac{-\frac{1}{2}\left[sgn\left(Diff(t) \times \tau - \alpha\right) + sgn\left(Diff(t) \times \tau - \beta\right)\right]}{RTT(t)} \end{cases}$$

  where $Diff(t) \times \tau = \left(\dfrac{W(t)}{\tau} - \dfrac{W(t)}{RTT(t)}\right)\tau = W(t)\left(1 - \dfrac{\tau}{RTT(t)}\right)$

- **buffer(t):** either $\sum_i \dfrac{W_i(t)}{\tau_i} < \mu$ and then $Buffer(t) = 0$

  or $\mu = \sum_i \dfrac{W_i(t)}{\frac{buffer(t)}{\mu} + \tau_i}$.

Fluid simulation

- The simulation consists then in calculating these quantities steps by steps by small time increments (they are equal to 1 in the following, but it would be better to take $INC = 10^{-2}$ for instance).
- **Algo. find_buffer(**$\mu, \forall i : \tau_i, \forall i : W_i(t)$**):**

  {Its solves the equation $\mu = \sum_i \frac{W_i(t)}{\frac{Buffer(t+1)}{\mu} + \tau_i}$ in $Buffer(t+1)$}.

```
$result := \sum_i \frac{W_i(t)}{\frac{Sup}{\mu} + \tau_i}$
**if** $result < \mu$ **then**
  **for** $i := 0$ **to** $Precision$ **do**
    $test := \frac{inf+sup}{2}$
    $result := \sum_i \frac{W_i(t)}{\frac{test}{\mu} + \tau_i}$
    **if** $result < \mu$ **then**
      $sup := test$
    **else**
      $inf := test$
$Buffer(t+1) := sup$
```

Fluid simulation

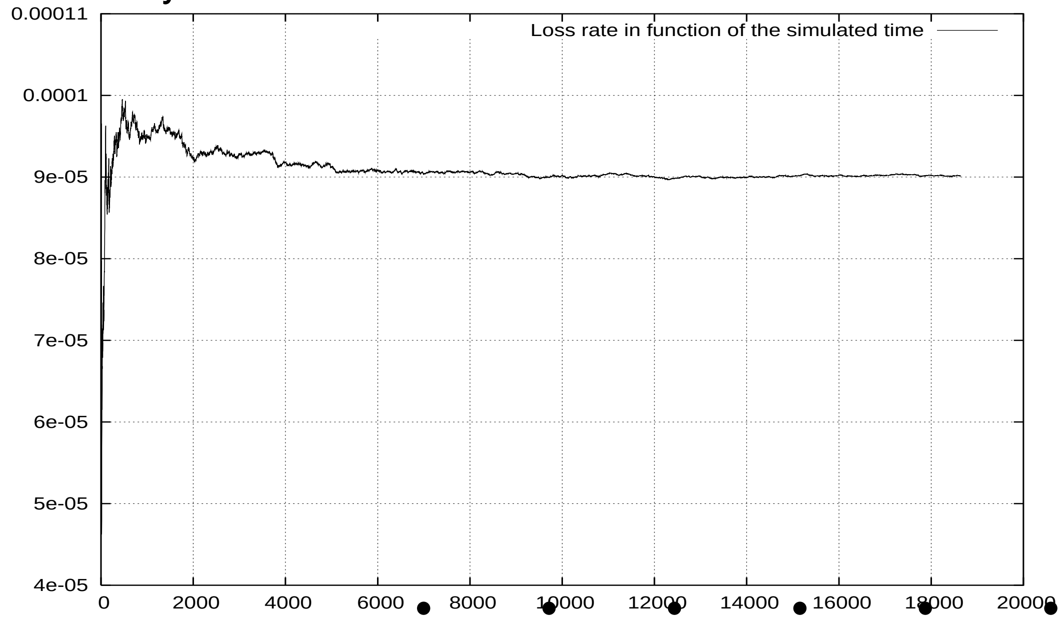
- **Algo. fluidsimulation:**

```
while TRUE do
  $inf := 0; sup := B;$
  if $\sum_i \frac{W_i(t)}{\tau_i} < \mu$ then
    $Buffer(t+1) := inf;$
  else
    find_buffer($\mu, \forall i : \tau_i, \forall i : W_i(t)$)
  while $Buffer(t) \geq B$ do {eventually gives $W_i(t+1)$}
    Multiply $W_i$ by $\gamma$
  $RTT_i(t+1) := \tau_i + \frac{Buffer(t+1)}{\mu}$
  if $W_i(t+1)\left(1 - \frac{\tau_i}{RTT_i(t+1)}\right) < \alpha$ then
    $W_i(t+1) := W_i(t) + \frac{1}{RTT_i(t)}$
  else if $W_i(t+1)\left(1 - \frac{\tau_i}{RTT_i(t+1)}\right) > \beta$ then
    $W_i(t+1) := W_i(t) - \frac{1}{RTT_i(t)}$
  else
    $W_i(t+1) := W_i(t)$
```

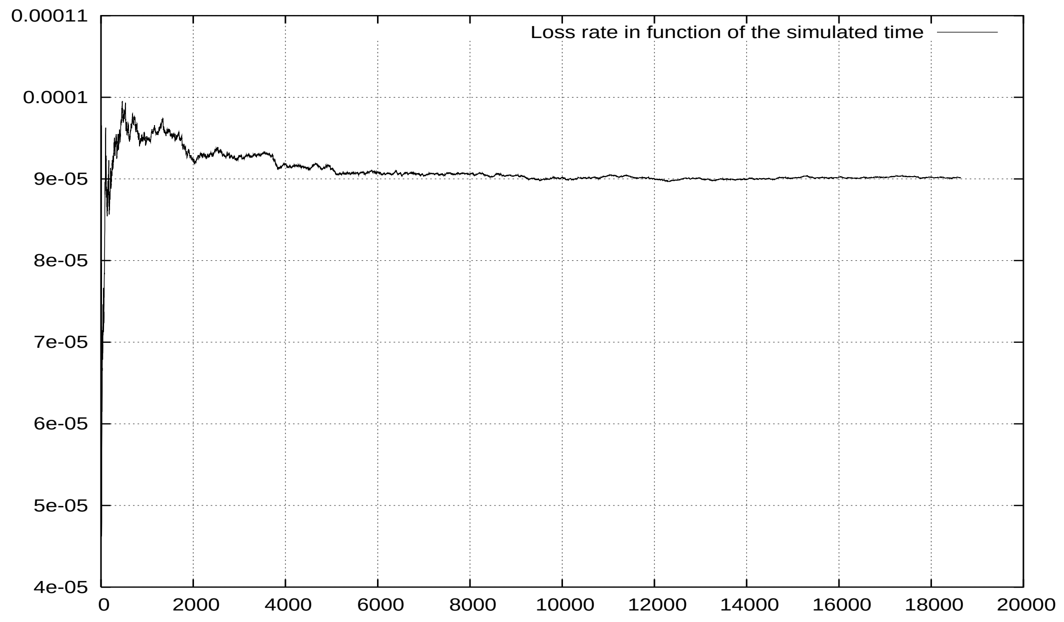
Validity of the simulation

Validity of the simulation

- Simulation is an experiment which consists in estimating a quantity (e.g. loss rate, response time, etc.).
- If we are interest by the loss rate, a variable, Nbsent, is incremented each time a packet is sent, another one, Nblosses, is incremented each time a loss occurs, and the ratio $\frac{Nblosses}{Nbsent}$ is observed.
- As the simulation progresses, this quantity must converge towards a given value, function of the parameters of system:



Validity of the simulation



- If the simulation is sufficiently long, a good estimation is obtained because the simulation has time to converge, otherwise the result is totally random !
- On the above example, if the simulation is stopped at $t = 1000s$, the estimated loss rate is false.
- **Problem:** HOW TO DETERMINE THE SIMULATION LENGTH ?

Validity of the simulation: law of large numbers

- **First method:** law of large numbers.
- Assume we want estimate a quantity μ by an empirical mean $\frac{1}{N}\sum_{i=1}^{N} X_i$.
- What is the confidence interval $\mathcal{I}$ of the mean μ (i.e.: such as $P(\mu \in \mathcal{I}) = 1-\eta$) for a confidence coefficient $1-\eta$?
- If the X_i are independant, with same mean and same standard-deviation σ,
$$\frac{1}{N}\sum_{i=1}^{N} X_i \rightsquigarrow \mathcal{N}\left(\mu, \frac{\sigma}{\sqrt{N}}\right)$$
- Thus, $\frac{\frac{1}{N}\sum_{i=1}^{N} X_i - \mu}{\sigma/\sqrt{N}} \rightsquigarrow \mathcal{N}(0,1)$, and so $P\left(\left|\frac{\frac{1}{N}\sum_{i=1}^{N} X_i - \mu}{\sigma/\sqrt{N}}\right| < t\right) = \Pi(t)$, where $\Pi(t)$ is the cumulative distribution function of the $\mathcal{N}(0,1)$ distribution.
- With a probability $1-\eta$, $\mathcal{I} = \left[\frac{1}{N}\sum_{i=1}^{N} X_i - \frac{t\sigma}{\sqrt{N}}; \frac{1}{N}\sum_{i=1}^{N} X_i + \frac{t\sigma}{\sqrt{N}}\right]$
- Unfortunately, this is true if the X_i are independant, which is rarely the case in simulations involving queues mechanisms (errors occur in bursts, long delays occur in sequences,...).

Validity of the simulation: Le Gall's method (1967)

- **Second method:** Le Gall's method. Let us suppose:
  - We want estimate a quantity Z with the estimator W_t; $\lim_{t\to+\infty} E\left[Z_t\right] = Z$ (i.e.: the estimator is without bias, for example an average: $Z_t = \dfrac{1}{T}\displaystyle\int_0^T Y(t)dt$);
  - An absolute error ε is tolerated with a confidence coefficient $1-\eta$, that is we want t such as

$$P\left(|Z_t - Z| < \varepsilon\right) > 1 - \eta.$$

- We can apply Tschebischev inequality: $\left(P\left(|Z_t - Z| \geq \varepsilon\right) \leq \dfrac{E\left[|Z_t - Z|^\alpha\right]}{\varepsilon^\alpha}\right)$

$$\left(P\left(|Z_t - Z| \geq \varepsilon\right) \leq \frac{E\left[|Z_t - Z|^2\right]}{\varepsilon^2}\right)$$

- For t sufficiently large, $Z \simeq E\left[Z_t\right]$, and so $E\left[|Z_t - Z|^2\right] \simeq VAR\left[Z_t\right]$.
- So, $P\left(|Z_t - Z| \geq \varepsilon\right) \leq \dfrac{VAR\left[Z_t\right]}{\varepsilon^2}$.

Validity of the simulation: Le Gall's method (1967)

$$P\left(|Z_t - Z| \geq \varepsilon\right) \leq \frac{VAR\left[Z_t\right]}{\varepsilon^2} \quad \Leftrightarrow \quad 1 - P\left(|Z_t - Z| < \varepsilon\right) \leq \frac{VAR\left[Z_t\right]}{\varepsilon^2}$$

$$\Leftrightarrow \quad P\left(|Z_t - Z| < \varepsilon\right) \geq 1 - \frac{VAR\left[Z_t\right]}{\varepsilon^2}.$$

- Thus, $1 - \dfrac{VAR\left[Z_t\right]}{\varepsilon^2} = 1 - \eta$.
- And then, $\varepsilon = \dfrac{\sigma_{Z_t}}{\sqrt{\eta}}$.
- For a confidence coefficient of 95%, $\eta = 5\%$ and so, $\boxed{\varepsilon = 4.5\sigma_{Z_t}}$.
- **Conclusion:** To estimate the confidence interval for a duration t, it suffices:
  - to run a simulation of duration $T \gg t$;
  - to split the simulation in k blocks of durations t;
  - to compute $Z_{t,i}$ on the block $i, i = 1, \ldots, k$;
  - to estimate the standard-deviation σ_t of the $Z_{t,i}$;
  - and to take $\varepsilon_t = 4.5\sigma_t$ (**Be careful:** $\varepsilon_T \neq 4.5\sigma_t$).

Validity of the simulation: Le Gall's method (1967)

- **Example:** estimation of a loss rate:

$$\begin{array}{|c|}\hline 0\\0\\1\\1\\0\\\hline 1\\0\\0\\1\\0\\\hline 0\\0\\1\\1\\1\\\hline 0\\0\\\vdots\\\hline\end{array}
\quad
\left.\begin{array}{l}
\left.\begin{array}{l}\\ \\ \\ \\ \\ \end{array}\right\}\text{Block 1 (size n)}\rightarrow Z_1 = \frac{1}{n}\sum_{k=1}^{n} x_i \\
\left.\begin{array}{l}\\ \\ \\ \\ \\ \end{array}\right\}\text{Block 2 (size n)}\rightarrow Z_2 = \frac{1}{n}\sum_{k=1}^{n} x_i \\
\left.\begin{array}{l}\\ \\ \\ \\ \\ \end{array}\right\}\text{Block 3 (size n)}\rightarrow Z_3 = \frac{1}{n}\sum_{k=1}^{n} x_i \\
\qquad\qquad\qquad\vdots
\end{array}\right\}
\rightarrow
\begin{array}{rcl}
\sigma_t & = & \sqrt{\frac{1}{k}\sum_{i=1}^{k} Z_i^2 - \left(\frac{1}{k}\sum_{i=1}^{k} Z_i\right)^2} \\
or, better: & & \\
 & = & \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{k} Z_i^2 - \frac{1}{k}\left(\sum_{i=1}^{k} X_i\right)^2\right]}
\end{array}$$

Validity of the simulation: Le Gall's method (1967)

- The confidence interval for a simulation of duration t (**and not** T) is thus about $4.5 \times \sigma_t$.
- **Remark:** In the case where temporal means are considered, and not means of events, each event must be weighted by its duration and sums must be replaced by integrals.
- **Problem:**
  - We are able to compute the confidence interval for a duration t, but the duration of the simulation is T: what is the confidence interval for T ?
  - If we want an absolute error of $\varepsilon_{t'}$, what must be the value of t'?
- We assume from now that $Z_t = \int_0^T Y(t)dt$ (i.e.: it is a mean).
- When $Y(t)$ is second-order stationnary:

$$\begin{aligned}
AR[Z_t] &= E\left[(Z_t - E[Z_t])^2\right] \\
&= E\left[\left(\frac{1}{T}\int_0^T Y(t)dt - E\left[\frac{1}{T}\int_0^T Y(t)dt\right]\right)^2\right] \\
&= E\left[\left(\frac{1}{T}\int_0^T Y(t)dt - \frac{1}{T}\int_0^T E\left[Y(t)\right]dt\right)^2\right] \quad \text{(linearity of the expectation)} \qquad (1)
\end{aligned}$$

Validity of the simulation: Le Gall's method (1967)

$$[Z_t] = E\left[\left(\frac{1}{T}\int_0^T Y(t)dt - \frac{1}{T}\int_0^T E\left[Y(t)\right]dt\right)^2\right] \quad \text{(linearity of the expectation)}$$

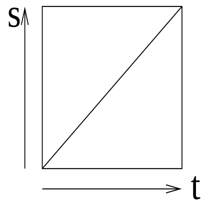
$$= E\left[\left(\frac{1}{T}\int_0^T \left(Y(t) - E\left[Y(t)\right]\right)dt\right)^2\right] \quad \text{(linearity of the integral...)}$$

$$= E\left[\left[\frac{1}{T}\int_0^T \left(Y(t) - E\left[Y(t)\right]\right)dt\right] \times \left[\frac{1}{T}\int_0^T \left(Y(s) - E\left[Y(s)\right]\right)ds\right]\right]$$

$$= E\left[\frac{1}{T}\int_{t=0}^T \frac{1}{T}\int_{s=0}^T \left(Y(t) - E\left[Y(t)\right]\right)\left(Y(s) - E\left[Y(s)\right]\right)dtds\right]$$

$$= \frac{1}{T^2}\int_{t=0}^T \int_{s=0}^T E\left[\left(Y(t) - E\left[Y(t)\right]\right)\left(Y(s) - E\left[Y(s)\right]\right)\right]dtds$$

$$= \frac{1}{T^2}\int_{t=0}^T \int_{s=0}^T COV\left[t,s\right]dtds$$



Validity of the simulation: Le Gall's method (1967)

$$\begin{aligned} VAR[Z_t] &= \frac{1}{T^2}\int_{t=0}^{T}\int_{s=0}^{T} COV\,[t,s]\,dtds \\ &= \frac{1}{T^2}\int_{t=0}^{T}\int_{s=t}^{T} COV\,[t,s]\,dtds \\ &= \frac{2}{T^2}\int_{t=0}^{T}(T-t)\,COV[0,t]dt \end{aligned}$$

(2ond-order stationarity + instead of integrating vertically, we integrate horizontally)

$$= \frac{2}{T}\int_{t=0}^{T}\left(1-\frac{t}{T}\right)COV[0,t]dt \quad (3)$$

- If $COV[0,t]$ tends to 0 faster than $1/u$ in $+\infty$, $\int_{t=0}^{T} t \times COV[0,t]dt$ is bounded.
- Then, let us denote $A^2 = \lim_{T\to+\infty} 2\int_0^T COV[0,u]du$.
- Thus, $\sigma^2(Z_t) \simeq \frac{A^2}{T}$ for T large (since $\lim_{T\to+\infty}\frac{1}{T}\displaystyle\int_0^T u \times COV[0,u]du = 0$).
- Finally, $\boxed{\varepsilon = 4.5\frac{A}{\sqrt{T}}}$.

Validity of the simulation: Le Gall's method (1967)

- **Remark1 :** to compute the confidence interval for a simulation of duration T, it suffices to split it in blocs of durations t, to compute $\varepsilon_t = 4.5\sigma_t$:

$$\begin{cases} \varepsilon_t = 4.5\frac{A}{\sqrt{t}} \\ \varepsilon_T = 4.5\frac{A}{\sqrt{T}} \end{cases} \quad \Rightarrow \quad \varepsilon_t = \sqrt{\frac{t}{T}}\varepsilon_t$$

- **Remark2 :** the spectral analysis gives directly A:
  - The Fourier transform of the autocovariance function is:

$$f(\lambda) = \frac{2}{\pi\sigma^2}\int_0^{+\infty} COV[0,u]cos(\lambda u)du$$

  - and then, $A^2 = \pi\sigma^2 f(0)$.

Random number generation

Random number generation of any probability distribution

- To simulate hazard, we need a sequence of random numbers:
  - numbers stocked in tables;
  - physical generators of white noise;
  - algoritms returning a sequence of non correlated numbers with a given distribution (**pseudo-random numbers**).
- Most of the time, the algorithms generating pseudo-random numbers are used because they allow to rerun several time the same simulation (useful to debug...).
- **Theorem:**
  - Let $U \rightsquigarrow Unif\left([0;1]\right)$ and F be a cumulative distribution function;
  - Then, the cumulative distribution function of $F^{-1}(U)$ is F.
- **Demonstration:**

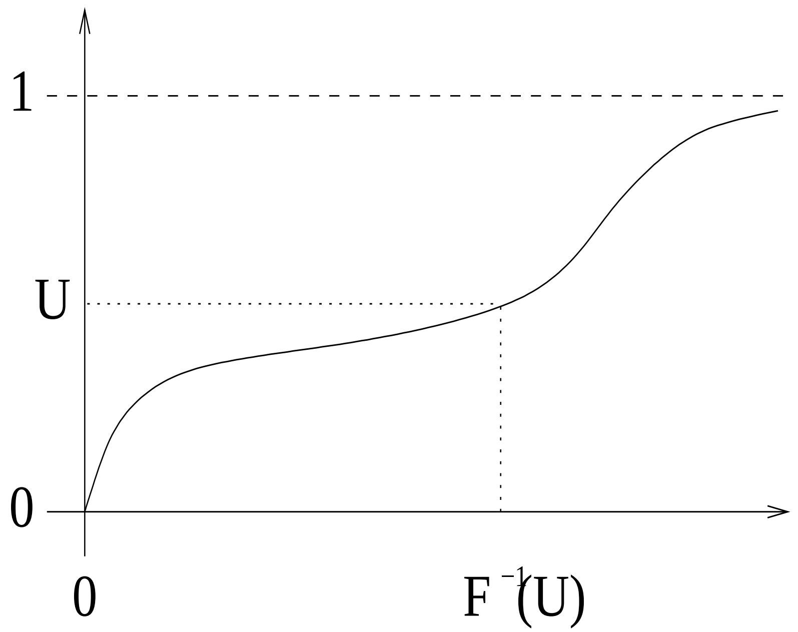
$$P\left(F^{-1}(U) \leq x\right) = P\left(U \leq F(x)\right)$$

because $F^{-1}(U) \leq x \Leftrightarrow U \leq F(x)$ since F is monotonous increasing.

Since U is uniformly distributed on $[0;1]$, $P\left(U \leq u\right) = \displaystyle\int_0^u ds = u$ so,

$$P\left(F^{-1}(U) \leq x\right) = P\left(U \leq F(x)\right) = F(x).$$

Random number generation of any probability distribution



- **Conclusion:** to generate non correlated random numbers following a distribution F, it suffices to draw U uniformly on [0;1], non autocorrelated, and to take $F^{-1}(U)$: it is **the inversion method**.

Generation of random numbers uniformly distributed on $[0; 1]$

- **Congruential method:**
  - $y_{n+1} := (a \times y_n + c) \mod m$
  - Take $z_{n+1} = \frac{y_{n+1}}{m}$ (non integer division).
  - This sequence is uniformly distributed on $[0; 1]$, and not autocorrelated.
  - a, c, and y_0 are integers, y_0 is the **seed**.
- **Theorem:** This sequence has a period.
- **Demonstration:**
  - y_n is integer and can take at most m distinct values.
  - So, there exist number p and q, $p > q$, such as $y_p = y_q$.
  - The algorithm being deterministic, it becomes periodic from q, with a period $< m$.
- **Theorem:** The period is m if and only if:
  - c and m are prime numbers;
  - $a - 1$ is a multiple of all prime divisor of m;
  - If 4 divide m, then 4 divide $a - 1$.

Generation of random numbers uniformly distributed on $[0; 1]$

- **Remark:** if the period is m, all the possible values are taken by y_n. Any y_0 can then be chosen.
- **Theorem:**
  - When $m = 2^\alpha$, $\alpha > 4$ and $c = 0$, the maximum period is $m4$.
  - It is exactly $m4$ if and only if:
    - y_0 is odd;
    - $a \equiv 5[8]$.
- **Remark :** The size of the period is not the only criterium. For some parameter values, there are repetitions of block of numbers.

Conclusions

Conclusions

- Different simulations methods:
  - **Step by step:** simple to implement but complex;
  - **Event driven:** efficient in terms of complexity, but heavy to develop;
  - **Fluid:** very efficient in terms of complexity and development, but sometime not sufficiently accurate
- A simulation must always be validated. Two formulæto know:
  - $\varepsilon_t = 4.5\sigma_t$;
  - $\varepsilon_t = \dfrac{A}{\sqrt{t}}$.
- Random number generations. Two methods to know:
  - **Congruential method**;
  - **Inversion method**.

References

References(1)

References

[1] A.-L. Beylot, M. Becker (Ed.) - *La simulation de réseaux*, Hermès.

[2] A. M. Law, W. D. Kelton - *Simulation modeling and analysis*. Mc Graw Hill.

[3] M. Becker - *Validité des simulations de files d'attente*. Doctorat d'État. 1976. Université de Paris VI.

[4] T. Bonald - *Comparison of TCP Reno and TCP Vegas via Fluid Approximation*. INRIA research report No3563. November 1998. (see also http://ww-sop.inria.fr/mistral/pub/vegas.html)