Simulaciones

Modelado y Análisis de Redes de Telecomunicaciones

Agenda

- Introducción
- Técnicas de simulación
  - Simulación a eventos discretos
- Precisión de la salida
- Generación de números aleatorios
- Monte Carlo

Introducción

- ¿Qué es una simulación?

Conjugar **simular.**

(Del lat. *simulāre*).

1. tr. Representar algo, fingiendo o imitando lo que no es.

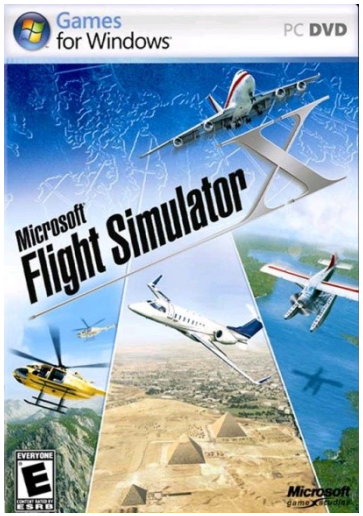
- A mí me gusta más esta:

WIKIPEDIA
La enciclopedia libre

Una definición más formal formulada por R.E. Shannon[1] es: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema".

Introducción

- En nuestro caso en particular:
  - *Un experimento realizado en la computadora donde la realidad es reemplazada por un programa.*



OMNeT++

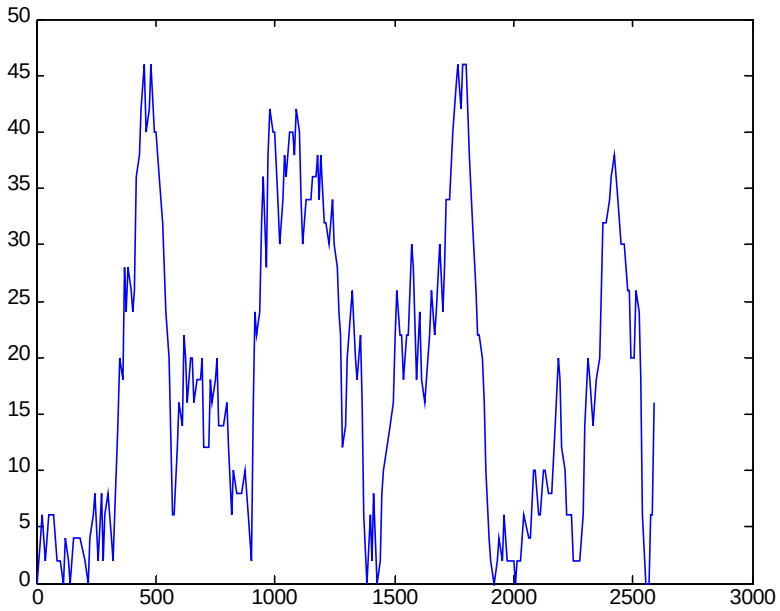
ns-2

Tipos de Simulación

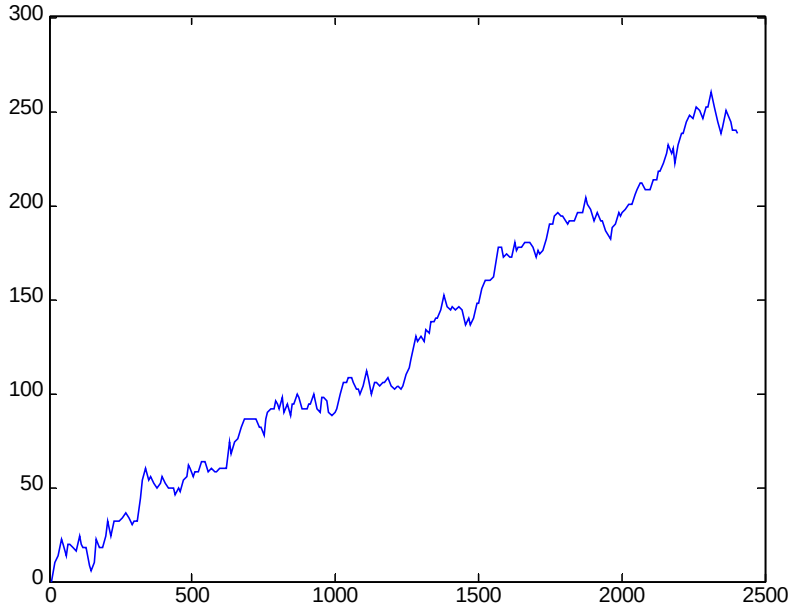
- Determinístico / Estocástico
  - Determinístico:
    - No hay componentes aleatorias
    - Se conoce TODO acerca del sistema a simular
  - Estocástico:
    - Por lo general existe alguna componente "desconocida" que se modela como aleatoria
- Finita / Infinita
  - A veces nos interesa un sistema hasta que sucede cierto evento
  - E.g.
    - El tiempo de respuesta de un sistema
    - La vida útil de un sistema

Estacionario

- Atención con las simulaciones estocásticas infinitas
- Ejemplo:
  - A un servidor llegan trabajos cada tiempo exponencial iid (media=λ)
  - Al servidor le toma un tiempo exp. iid atender cada trabajo (media=μ)
  - Mientras no se atienden, los mensajes se guardan en un buffer
  - Me interesa la ocupación del buffer

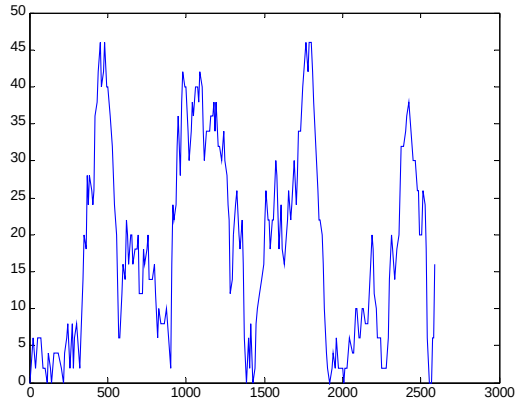


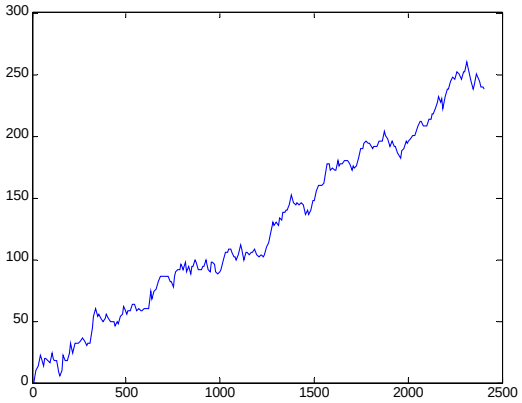
λ/μ= 0.96



λ/μ = 1.05

Estacionario





- La primera simulación tiene un transitorio, y después pasa a un comportamiento “normal” (estacionario)
- La segunda simulación no tiene comportamiento estacionario
  - Se puede inferir el estado de la simulación a partir del estado inicial y del tiempo de simulación
    - ¿Qué sentido tiene una simulación de este tipo cuando no hay un final claro?
    - ¿Qué condición inicial poner?
- Hay que tratar que:
  - La simulación sea estacionaria
  - Medir la variable de interés una vez que se llegó al estado estacionario de la simulación

Estacionario

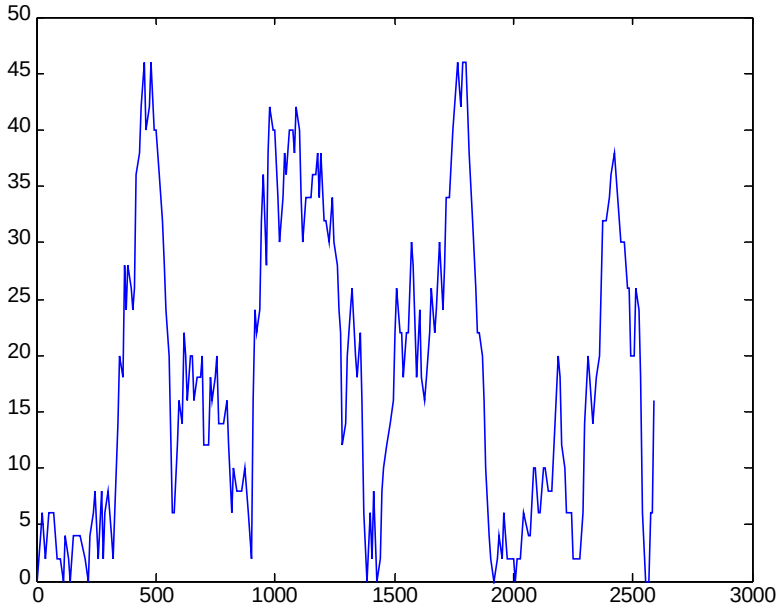
- En realidad estacionario es una propiedad del sistema a ser simulado
- Sea X_t un proceso estocástico que representa el estado de la simulación
- X_t es estacionario sii las estadísticas de todos los órdenes permanecen incambiados por una traslación temporal:

$$F_{X_{t_1+\tau},\ldots,X_{t_n+\tau}}(x_1,\ldots,x_n) = F_{X_{t_1},\ldots,X_{t_n}}(x_1,\ldots,x_n) \quad \forall n,\ \forall t_1,\ldots,t_n$$

- Más en concreto: si hace cuánto está corriendo la simulación no nos dice nada

Estacionario

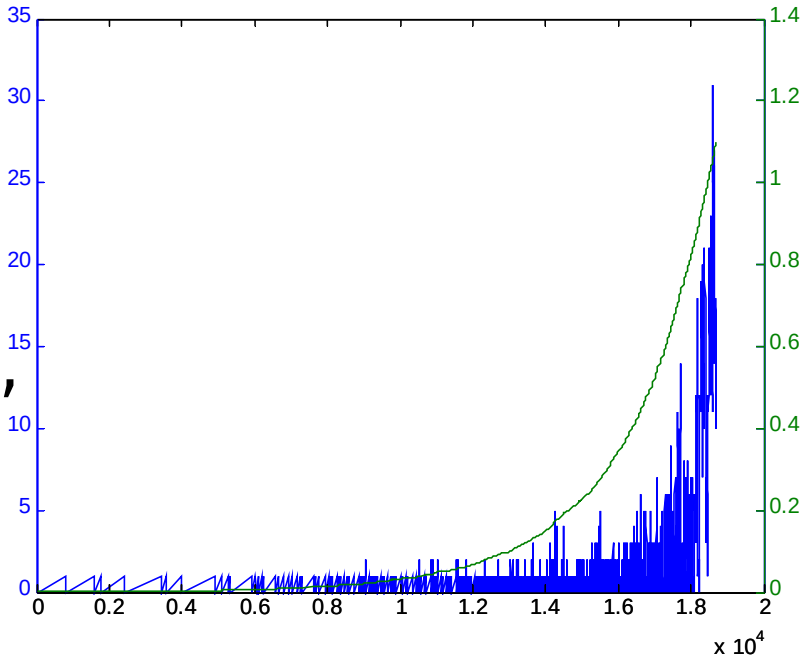
- Algunos sistemas son asintóticamente estacionarios
  - En esos casos hay que lidiar con el transitorio
  - En la mayoría de los casos una inspección visual es suficiente
  - E.g. tomar los valores de ocupación del buffer en t>500



$\lambda/\mu = 0.96$

Estacionario

- Razones para que el sistema no sea estacionario:
  - Dependencias temporales
    - E.g. crecimientos
    - Se puede ignorar en escalas cortas
  - Inestabilidad
    - E.g. buffer con $\lambda/\mu > 1$
- Ejemplo de ambos:
  - Buffer con λ/μ creciente, más allá de 1.0



Agenda

- Introducción
- Técnicas de simulación
  - Simulación a eventos discretos
- Precisión de la salida
- Generación de números aleatorios
- Monte Carlo

Tiempo de simulación

- Tiempo de simulación ≠ Tiempo real
  - Acciones simultáneas en la realidad
  - Eventos serializados en la simulación
    - Acción = compuesta por eventos instantáneos
      - E.g. el envío de un paquete está definido por un comienzo y un final
    - Se mantiene un **tiempo de simulación**
    - Cada evento ocurriría a un cierto tiempo de simulación en el sistema real, y se asume que no puede haber eventos simultáneos
- La simulación avanza de evento en evento por su tiempo de simulación asociado
- Esta técnica se llama **simulación a eventos discretos**

Simulador a E.D.: componentes

- El simulador está formado básicamente por tres elementos:
  - `currentTime`: el tiempo global de la simulación
  - `eventScheduler`: lista de `event`s ordenados de forma creciente según su momento de ejecución
  - `event`: representa una transición y tiene un momento de ejecución asociado
- El funcionamiento es básicamente el siguiente loop:

1. `currentEvent := eventScheduler.nextEvent()`
2. `currentTime := currentEvent.time`
3. `execute(currentEvent)` // Entre otras cosas puede agendar nuevos eventos, o modificar/eliminar eventos futuros
4. `update(counters)` // los contadores de las estadísticas de interés

Ejemplo: simu. de un servidor

- Un servidor recibe pedidos y los va sirviendo de a uno por el orden de llegada.
- El tiempo entre llegadas sucesivas y el tiempo que lleva atender cada pedido se asumen aleatorios e independientes entre sí con distribuciones *F*() y *G*()

- Nos interesan
  - el tiempo medio de atención
  - la cantidad media de pedidos pendientes en el servidor

Ejemplo: simu. de un servidor

- Clases adicionales:
  - `Buffer`
    - Representa la cola de pedidos
    - El atributo `length` contiene la cantidad de pedidos pendientes
    - El atributo `nbRequests` es un contador con la cantidad de pedidos servidos hasta ahora
  - `Request`
    - Representa los pedidos que llegan al servidor
    - Por cada pedido pendiente, el programa tiene un objeto `Request`
    - El atributo `arrivalTime` indica el momento en que el `Request` llegó al sistema

Ejemplo: simu. de un servidor

- Tipos de eventos:
  - `Arrival`
    - Representa la llegada de un nuevo pedido al servidor
    - `execute(event)`
      - Crea un nuevo objeto de la clase `Request` con `arrivalTime` igual a `currentTime` y lo pone al final de `buffer`
      - Sortea un número Δ con distribución *F*(). Crea un nuevo evento de la clase `Arrival` con `time=currentTime+`Δ y lo inserta en el `eventScheduler`
      - Si la cola estaba vacía antes de la inserción, crear un nuevo evento de la clase `Service` con `time=currentTime`
  - `Service`
    - Representa el comienzo de la atención de un pedido
    - `execute(event)`
      - Sortea un número Δ con distribución *G*(). Crea un nuevo evento de la clase `Departure` con `time=currentTime+`Δ y lo inserta en el `eventScheduler`
  - `Departure`
    - Representa el final de la atención de un pedido
    - `execute(event)`
      - Toma el primer elemento de `buffer` y lo elimina.
      - Si `buffer` no queda vacío, crear un nuevo evento de la clase `Service` con `time=currentTime`

Ejemplo: simu. de un servidor

- Estadística de interés
  - Tamaño medio de la cola
    - Sea `queueLengthCtr`=$\int_0^t q(s)ds$
      - con $q(s)$ `buffer.length` en tiempo s
    - Entonces el tamaño medio de la cola es `queueLengthCtr`/T, con T el tiempo de simulación
    - Para mantener la variable `queueLenghtCtr` ante cada evento, se le adiciona `buffer.length*(currentTime-t_last_event)`
  - Tiempo medio de atención
    - Se mantiene la variable `responseTimeCtr`=$\sum T(n)$
      - con $T(n)$ el tiempo de atención del pedido n
    - Cada evento del tipo `Departure` debe actualizar la variable:
      - `responseTimeCtr += currentTime - request.arrivalTime`
- Notar la diferencia de perspectiva de los contadores:
  - temporal
  - eventual

Agenda

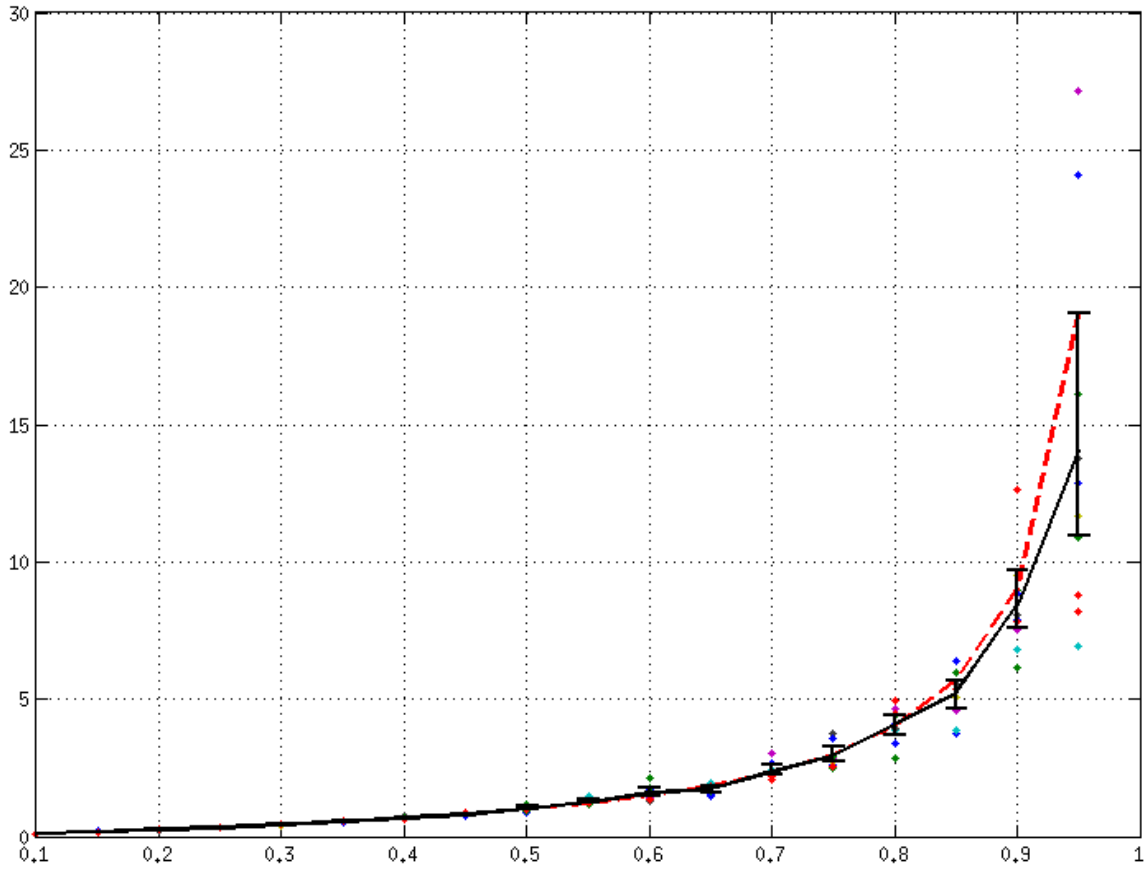
- Introducción
- Técnicas de simulación
  - Simulación a eventos discretos
- Precisión de la salida
- Generación de números aleatorios
- Monte Carlo

Precisión de la salida

- Cuando la simulación es aleatoria, la salida también es aleatoria
- No se puede dar UN único resultado
  - Éste tiene que estar acompañado de una medida de su precisión
- La técnica más clásica: replicación
  - Generara N salidas (N simulaciones)
  - Dar un intervalo de confianza de los N valores
  - CUIDADO: los N valores tienen que haber sido generados independientemente
  - Recordar sacar el transitorio

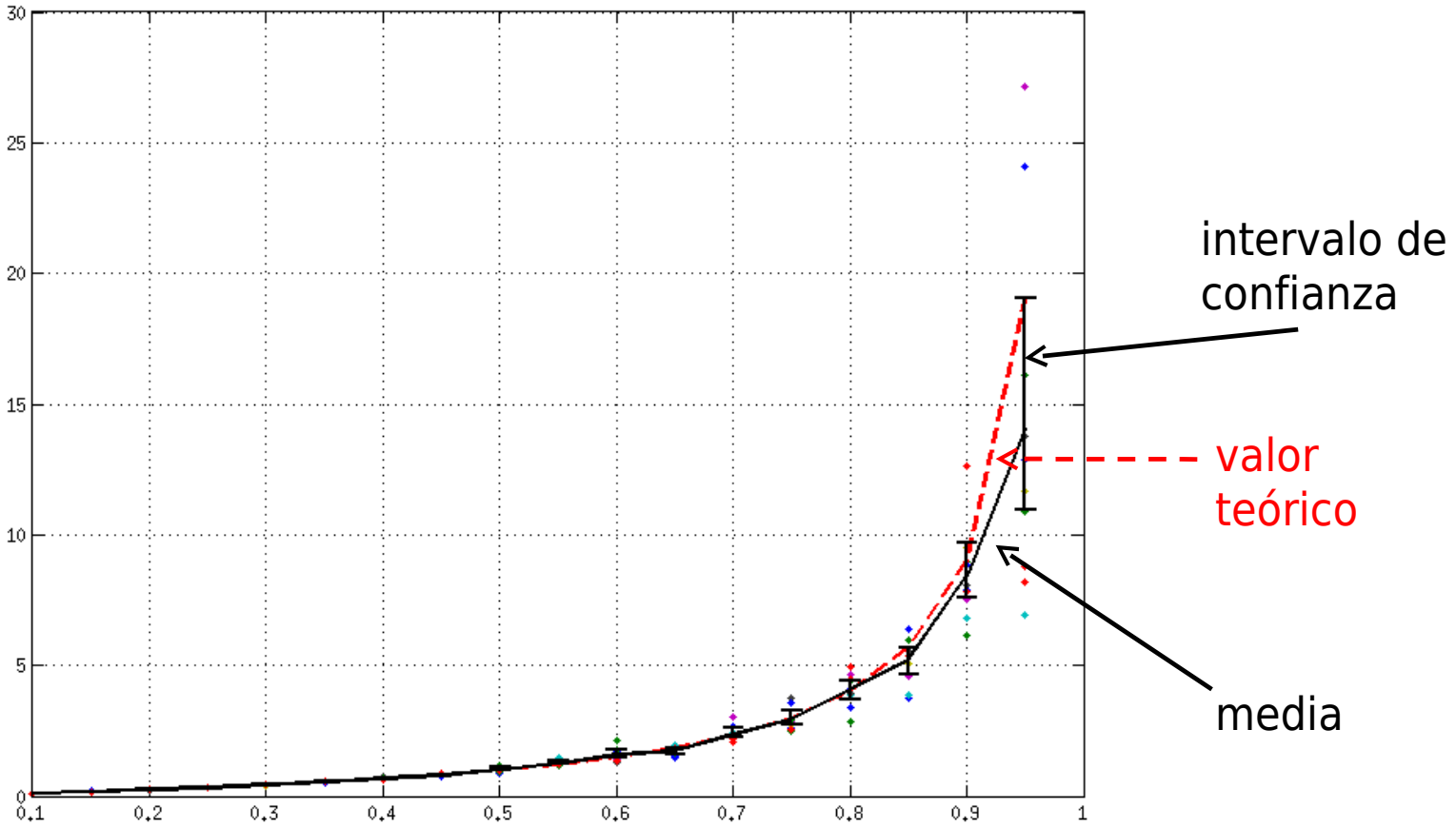
Ejemplo

- Tamaño medio de la cola del servidor (10 simulaciones por valor de λ/μ)



Ejemplo

- Tamaño medio de la cola del servidor (10 simulaciones por valor de λ/μ)



Agenda

- Introducción
- Técnicas de simulación
  - Simulación a eventos discretos
- Precisión de la salida
- Generación de números aleatorios
- Monte Carlo

Generación de Nos. Aleatorios

- Toda simulación estocástica requiere una función que genere números aleatorios
  - Sortea un número Δ con distribución $F()$. Crea un nuevo evento de la clase `Arrival` con `time=currentTime+`Δ y lo inserta en el `eventScheduler`
  - Sortea un número Δ con distribución $G()$. Crea un nuevo evento de la clase `Departure` con `time=currentTime+`Δ y lo inserta en el `eventScheduler`
- Veremos que para generar números con una distribución $F()$ basta con generar $U[0,1]$
- ¿Cómo generar números aleatorios?
  - Procesos mecánicos
    - Tirar dados o cartas
    - …
    - Fenómenos cuánticos (e.g. fotones atravesando ciertas superficies)
  - Métodos computacionales

Generación de Nos. Aleatorios

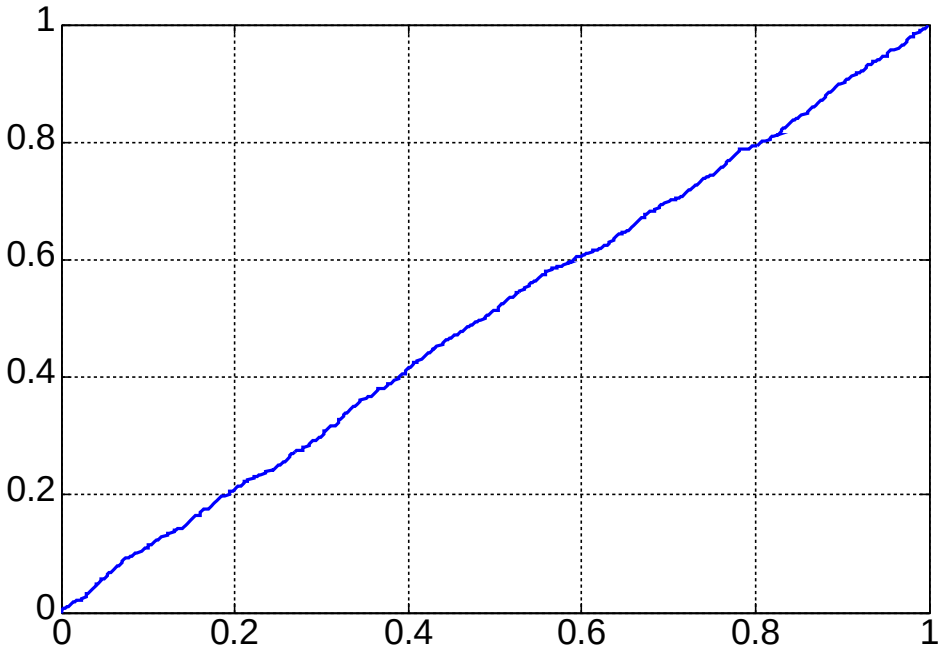
- Métodos computacionales:
  - Números pseudo-aleatorios
- Los generadores aleatorios implementados en software no son exactamente aleatorios
  - Generan una secuencia perfectamente determinística a partir de un valor inicial (llamado semilla o **seed**)
  - La secuencia resultante "parece" aleatoria (desde un punto de vista estadístico)
  - La idea es que pequeños cambios del seed generen grandes cambios en la secuencia de números generados
- Ejemplo: linear congruential generator (usado en algunas versiones de ns-2)

$$x_n = (a x_{n-1} + c) \bmod m$$

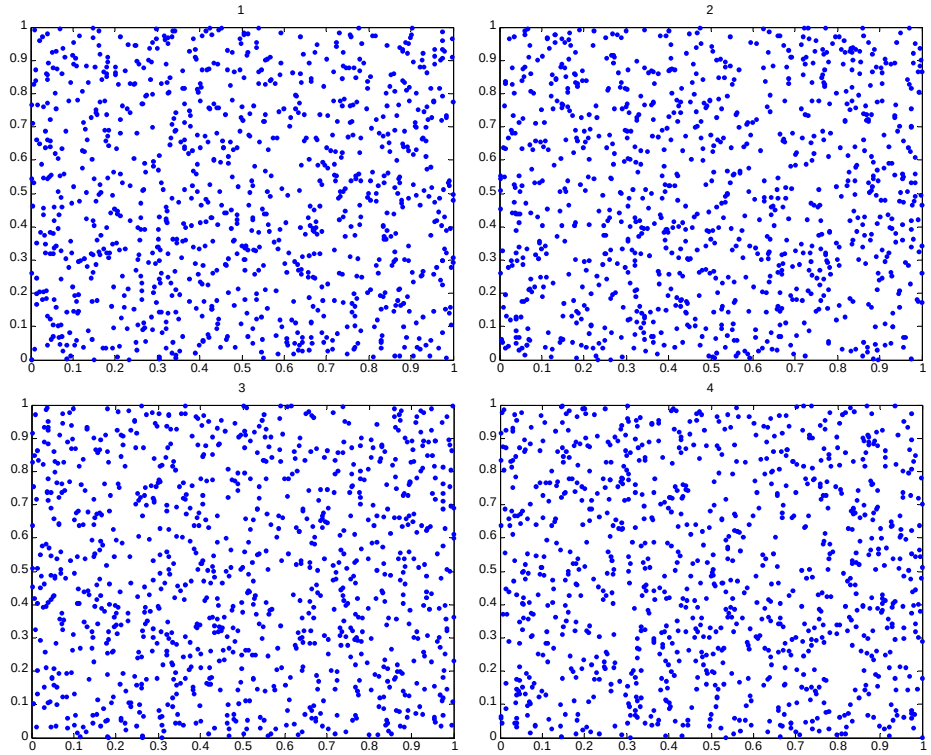
  - x_0 es la semilla

Generación de Nos. Aleatorios

- Ejemplo: 1000 muestras del linear congruential generator con a=16807, $m=2^{31}-1$, c=0, $x_0=1$



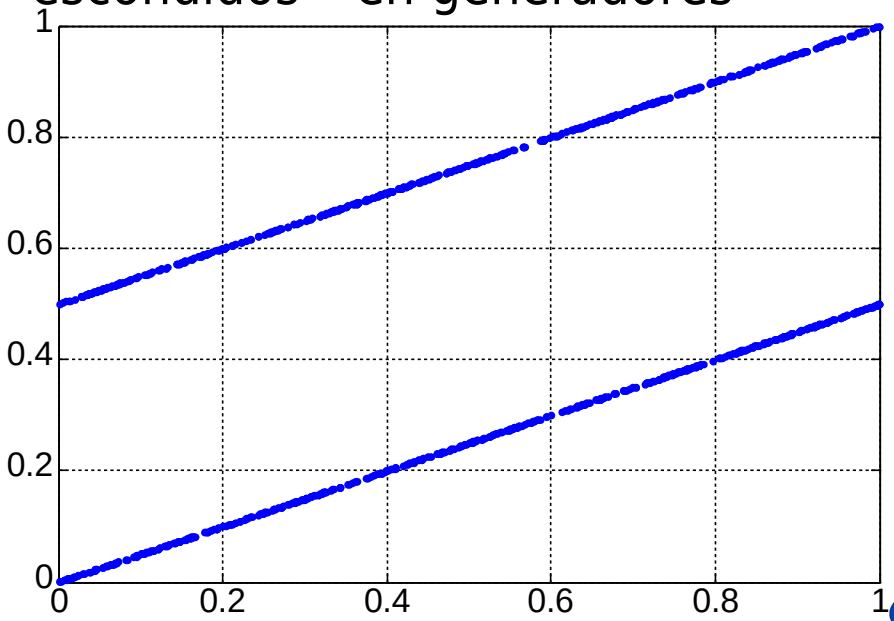
Distribución empírica



(X_n, X_{n-a}) (a=1,2,3,4)

Generación de Nos. Aleatorios

- Algunas consideraciones importantes:
  - Todas las secuencias de números aleatorios tienen un cierto período
    - Es NECESARIO que la cantidad de números que uno “saca” de la secuencia sea menor que el período
    - En el caso particular del linear congruential generator el período es m
  - Hay algunos problemas “escondidos” en generadores demasiado simples
    - E.g. (x_n,y_n) con x_n e y_n generados con seeds 1 y 2
- Es IMPORTANTE usar un buen generador



Generación de Nos. Aleatorios

On Shortcomings of the ns-2
Random Number Generator

Univ.-Doz. Dr. Karl Entacher† and Dipl.-Ing.(FH) Bernhard Hechenleitner†‡

Table 1: Results of simulations using the original RNG component of ns-2.

Simulation	RNG seed set 1 ('good' seeds)	RNG seed set 2 ('bad' seeds)	RNG seed set 3 ('good' seeds)	RNG seed set 4 ('bad' seeds)
$\overline{q}$	19.451	24.288	19.374	17.573

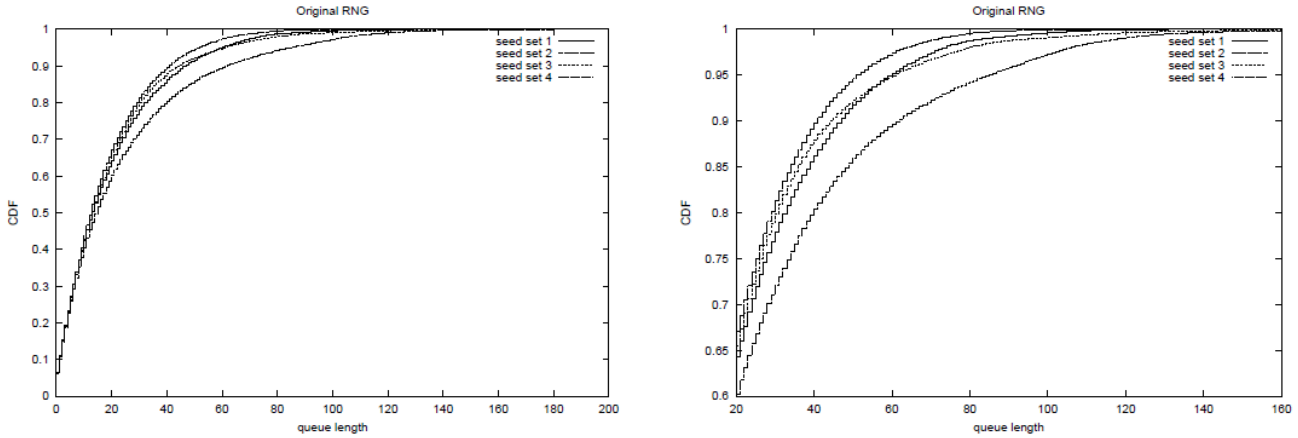


Figure 5: CDFs of the queue lengths when using the original RNG component of ns-2 in combination with different seed sets (right plot = enlarged view).

Generación de Nos. Aleatorios

LA DIRECCION NACIONAL
RESUELVE:

1) El mecanismo a aplicar por la Dirección Nacional de Loterías y Quinielas , para estos concursos será el siguiente:

a) Se ordenarán los postulantes del 1 al M (siendo M el total de participantes)

b) Se determinará un número elegido al azar (X_1) que será el primer número sorteado.

c) Se utilizarán los bolilleros actuales, que contendrán la cantidad de dígitos que requiera el número a ser sorteado.

d) Se sorteará un segundo (Y_1) que oficiará como número intervalo(no se trata de un número sorteado).

e) La cantidad de dígitos que contenga este segundo número(intrvalo) tendrá un dígito menos que el primer número sorteado.

f) Los subsiguientes números sorteados (X_2 al X_s) se determinarán de acuerdo con la siguiente ecuación:

$X_1 + Y = X_2$

$X_2 + Y = X_3$ y así sucesivamente.....

$X_{n-1} + Y = X_n$ (n variando entre 1 y s)

g) En el caso de que el número sorteado más el intervalo supere al número M (número maximo del universo) la formula será : $X_{n-1}+Y_1-M = X_n$

h) Si el número M fuese multiplo del número Y y se diese el caso que se repitieran los números sorteados, se procederá a partir de que ello ocurra a sortear un segundo número Y_2 intervalo.

i) Este procedimiento se aplicará hasta elegir la cantidad de números sorteados (s) que se requieran.

2) Pase al Departamento de Escribanía a los efectos correspondientes.

Cr. ALBERTO G. QUEIJO
DIRECTOR NACIONAL DE
LOTERIAS Y QUINIELAS

As. 800/2011
AGQ/gz

Generación de Nos. Aleatorios

Ingreso Datos		
Datos Sorteados	Número sorteado de Inicio	3190
	Número sorteado de Intervalo	692
Datos Fijos	Total de Inscriptos	4365
	Cantidad de Cargos	200

- Algunos de los sorteados:

2177	4.311.574,3	Normey Gomez, Zoraida
2178	4.311.573,7	Normey Gomez, Zaira Yesmin
314	4.477.551,2	Losada De Leòn, Camilo
315	4.216.069,0	Losada De Leòn, Maximiliano
952	1.782.346,3	Gerner Tchikachuili, Andrea
953	1.782.335,0	Gerner Tchikachuili, Mónica
1378	4.459.909,3	Quinteros Da Silva, Sharon Vanessa
1379	4.459.907,1	Quinteros Da Silva, Carlos Omar
2017	4.019.433,6	Lenkowicz Bialostosky, Fabiàn
2018	4.032.964,6	Lenkowicz Bialostosky, Ariel

- ¿Escándalo?

Generación de Nos. Aleatorios

- Las versiones nuevas (>2.1b9) implementan un generador nuevo (MRG32k3a de L'Ecuyer)
  - No es necesario asignar el seed
  - Para generar series no correlacionadas
    - llamar el método `next-substream`
    - crear un generador nuevo
  - => Para generar una corrida independiente, se procede de la siguiente forma (con `$run` un número cualquiera, diferente para cada simulación):

```
# create the RNGs and set them to the correct substream
set arrivalRNG [new RNG]
set sizeRNG [new RNG]
for {set j 1} {$j < $run} {incr j} {
  $arrivalRNG next-substream
  $sizeRNG next-substream
}
```

Generación de Nos. Aleatorios

- Ahora que tenemos números aleatorios entre 0 y 1, ¿cómo generar realizaciones de una V.A. con una distribución arbitraria *F*()?
- Para muchas de las distribuciones “típicas” el propio programa ya se encarga de eso
  - E.g. NS-2

```
# arrival_ is a exponential random variable describing the time
# between consecutive packet arrivals
set arrival_ [new RandomVariable/Exponential]
$arrival_ set avg_ 5
$arrival_ use-rng $arrivalRNG
```

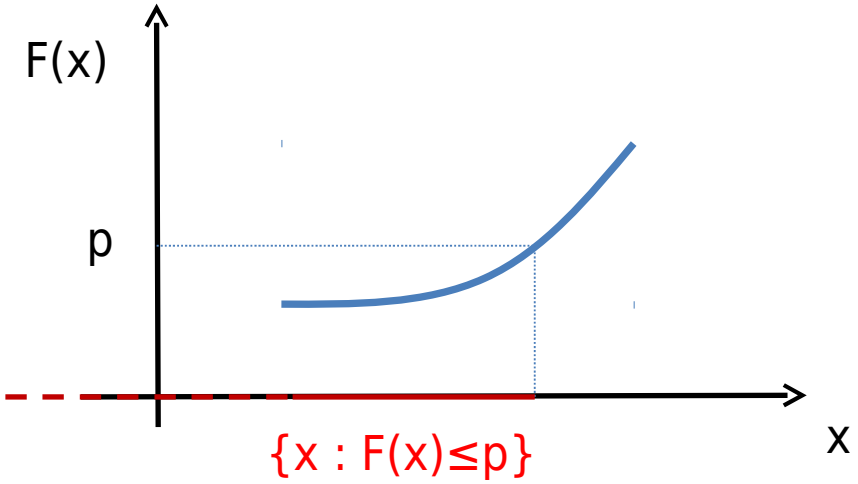
  - De todas formas vamos a repasar algunos métodos

Inversión de CDF

- Sean X una V.A. con distribución $F(x)$
  - La pseudo-inversa F^{-1} se define:

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

    - Si $F(x)$ es continua y creciente en x, entonces $F^{-1}(x)$ se encuentra "fácilmente" resolviendo $F(x)=p$

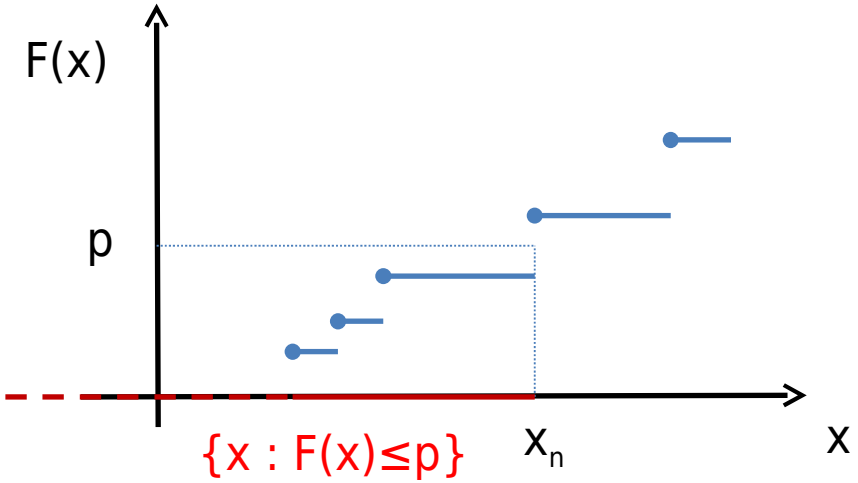


Inversión de CDF

- Sean X una V.A. con distribución $F(x)$
  - La pseudo-inversa F^{-1} se define:

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

  - Si $F(x)$ es discreta: $F^{-1}(p)=x_n$ <=> $F(x_n-1) \leq p < F(x_n)$



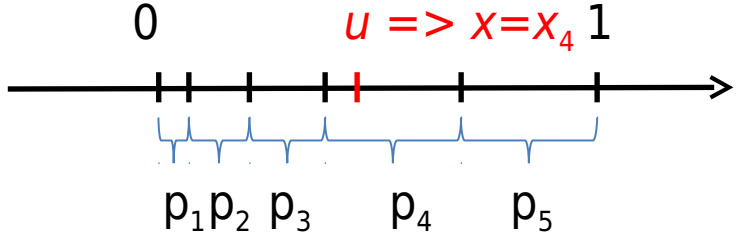
Inversión de CDF

- Teorema:
  - Sea *U* una V.A. con distribución unif. en [0,1] => la V.A. $F^{-1}(U)$ tiene distribución *F*()
  - Dem: (para el caso absolutamente continuo y creciente)

$$P(F^{-1}(U) \leq y) = P(U \leq F(y)) = F(y)$$

- Lema:
  - Sea X una V.A. discreta con $P(X=x_k)=p_k$ => una muestra *x* de *X* se obtiene de una muestra *u* de *U*~*U*[0,1] asignando *x* como x_k, donde *k* cumple que

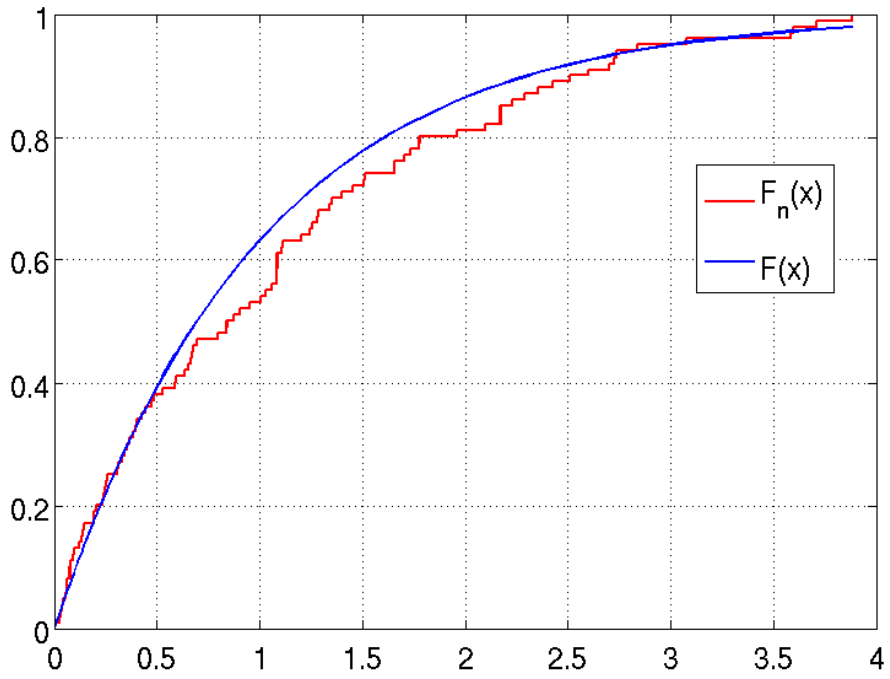
$$\sum_{i=0}^{k-1} p_i \leq u < \sum_{i=0}^{k} p_i$$



Inversión de CDF

- Ejemplo: generación de una exponencial (100 muestras)

$$F(x) = \begin{cases} 1 - e^{-\lambda x}; & x \geq 0 \\ 0 & x < 0 \end{cases} \Rightarrow F^{-1}(p) = -\frac{\log(1-p)}{\lambda}$$



Rejection Sampling

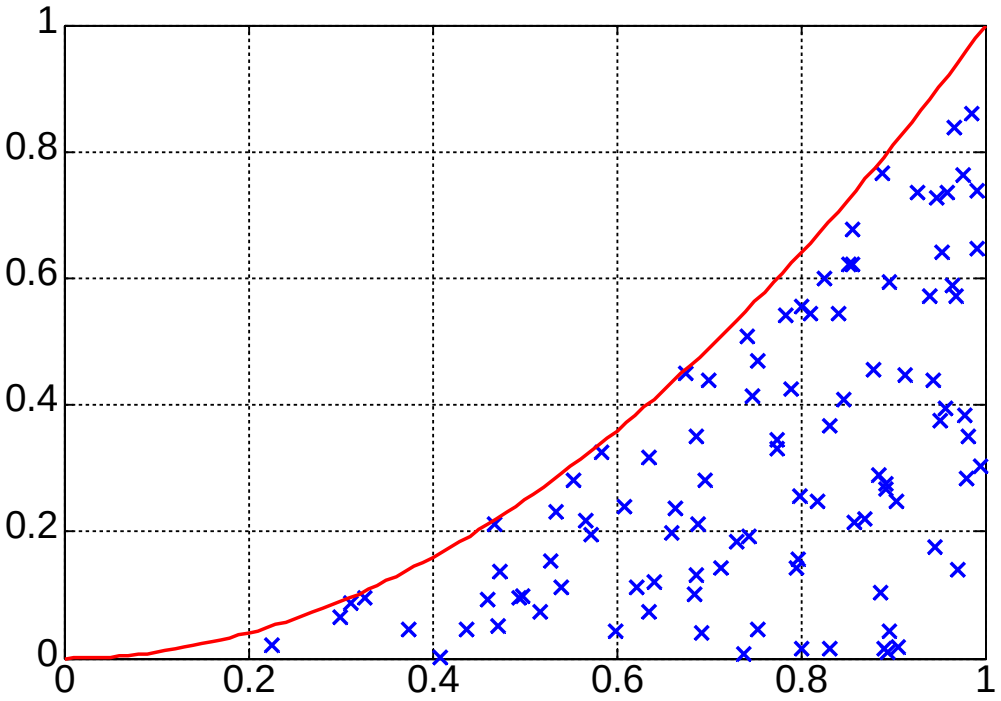
- Cuando
  - invertir es difícil
  - quiero generar vectores aleatorios
  - conozco la densidad a excepción de una constante

  se puede usar el método de Rejection Sampling
- Teorema:
  - Sea X una V.A. (o Vector Aleatorio) tal que su distribución es la distribución de la V.A. Y dado que la V.A. $Z \in A$ (i.e. para el caso unidimensional $P(X \leq x) = P(Y \leq x \mid Z \in A)$)
  - Si se conoce una forma de generar muestras de Y y Z entonces una forma de obtener una muestra de X es

Do
    Sacar una muestra y y z de Y y Z
Until $z \in A$
Return y

Rejection Sampling

- Ejemplo: Sea X un vector aleatorio en R^2 uniformemente distribuido en un conjunto acotado cualquiera *A*
  - Para generar una muestra de *X* basta con generar muestras de *U* uniforme en un conjunto que contenga *A* hasta que la muestra esté en *A*
  - Es decir: *Y=U* y *Z=U* (preguntas: ¿cómo generar muestras de *U*? ¿cómo probar P(X∈B)=P(U∈B | U∈A)?)



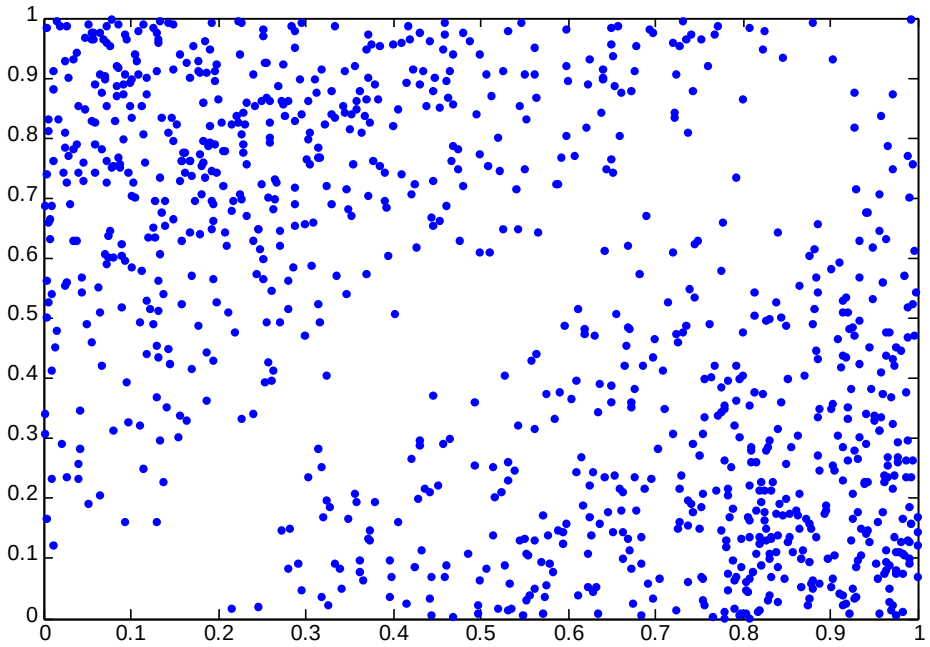
Rejection Sampling

- Consecuencia del teorema anterior es el método de Rejection Sampling
- Teorema
  - Sean X e Y dos V.A. (o vectores) con densidades $f_X(x)$ y $f_Y(x)$ respectivamente
  - Asumir que:
    - Se conoce un método para obtener muestras de Y
    - La densidad de X se conoce hasta una cierta constante (i.e. $f_X(x)=Kg_X(x)$ con $g_X(x)$ conocida)
    - Existe c>0 tal que $g_X(x)/f_Y(x)\leq c$
  - Una muestra de X se puede obtener del siguiente método:

Do
  Sacar una muestra y y u de Y y $U\sim U[0,c]$
Until $u\leq g_X(y)/f_Y(y)$
Return y

Rejection Sampling

- Ejemplo:
  - Queremos generar un vector aleatorio (X_1,X_2) en el cuadrado unitario cuya densidad sea proporcional a $|X_1-X_2|$
    - $f_X(x)=K|X_1-X_2|=Kg_X(x)$
    - $Y \sim U([0,1]^2)$
    - $g_X(x)/f_Y(x)=|X_1-X_2|/1 \leq 1 \Rightarrow c=1$



Muestras de una V.A. Normal

- ¿Cómo generar muestras normales?
  - Invertir la CDF de la normal no se puede
  - Podemos aplicar rejection sampling
    - Generamos muestras para una V.A. con soporte infinito (e.g. la exponencial usando inversión de CDF)
    - Cuidado porque la exponencial tiene soporte sólo para los positivos
      - se soluciona fácil generando solo normales positivas y después sorteándoles un signo
    - ¿desventajas?

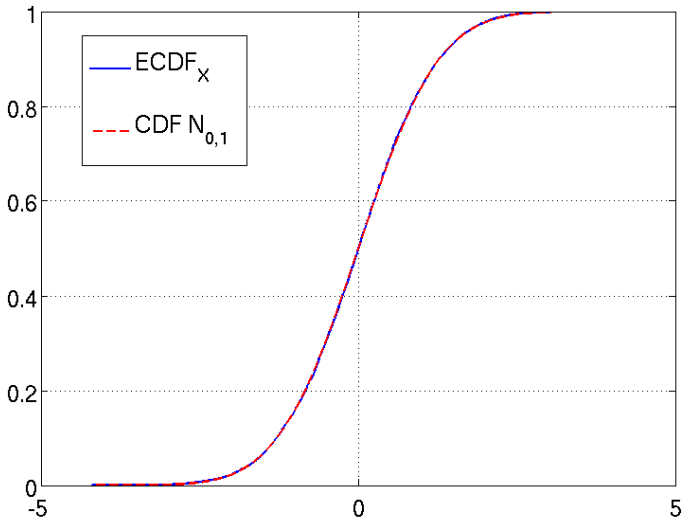
$g_X(x)=e^{-xx/2}1_{x\geq 0}$; $f_Y(x)= e^{-x}1_{x\geq 0}$ => $c=e^{-1/2}$

Do

Sacar una muestra y y u de Y y $U\sim U[0,c]$

Until $u \leq e^{-y(y/2-1)}$

Return signo$(v-0.5)y$ con $v\sim U[0,1]$



Muestras de una V.A. Normal

- Método de Box-Müller
  - Sean X e Y dos V.A. con distribución normal estándar
  - Sean:

$$R = \sqrt{X^2 + Y^2}$$

$$\Theta = arg(X + jY)$$

    - Se puede probar que $R \sim \text{Rayleigh}_1$ ($F(x)=1-e^{-xx/(2\sigma\sigma)}$) y que $\Theta \sim U[0,2\pi]$

> Sacar una muestra u y θ con $U \sim U[0,1]$ y
> $\Theta \sim U[0,2\pi]$
> $r=(-2\log(1-u))^{1/2}$
> $x = r \cos \theta$
> $y = r \sin \theta$
> Devolver x e y

- El método de Box-Müller es un ejemplo de la técnica de cambio de variable

Agenda

- Introducción
- Técnicas de simulación
  - Simulación a eventos discretos
- Precisión de la salida
- Generación de números aleatorios
- Monte Carlo

Monte Carlo

- El método de Monte Carlo hace referencia a muchas cosas
- Por lo general, se refiere a integración por Monte Carlo
- La idea se basa en la ley fuerte de los grandes números y es la siguiente:

$$X \sim U[0,1]^n \Rightarrow E\{g(X)\} = \int_{[0,1]^n} g(x)dx$$

$$\int_{[0,1]^n} g(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

donde x_i son muestras independientes de X

- Por teorema central del límite

$$\frac{\overline{g(X)} - E\{g(X)\}}{\sigma/\sqrt{N}} \sim N_{0,1}$$

  - I.e. la varianza del error (y por lo tanto el error cuadrático medio) decrece como 1/N => para aumentar la precisión por 10 hacen falta 100 veces más muestras

Monte Carlo

- Tomando el caso específico de $g(X)=1_{\{X\in A\}}$ resulta en una estimación de la probabilidad $P(X\in A)$
- Es útil cuando $P(X\in A)$ es difícil de calcular, pero sabemos como generar muestras de X
- En las próximas clases veremos cómo hacer para estimar el "error" y así poder decidir cuántas muestras tenemos que tomar para asegurarnos una cierto grado de precisión