TEMA 3: Paradigmas de simulación

MODELOS COMPUTACIONALES Y SIMULACIÓN DE SISTEMAS
Curso 2025-2026

UA
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

DFESTS

Índice

Índice

1. Introducción

- El estudio de los **paradigmas de simulación** constituye uno de los pilares fundamentales en la modelización computacional de sistemas.
- Un paradigma de simulación es un enfoque o marco conceptual que define **cómo se representan, procesan y analizan los sistemas reales** dentro de un entorno computacional.
- En esencia, cada paradigma define una **forma diferente de concebir** el tiempo, los eventos, las interacciones entre componentes y la evolución del sistema.

1. Introducción

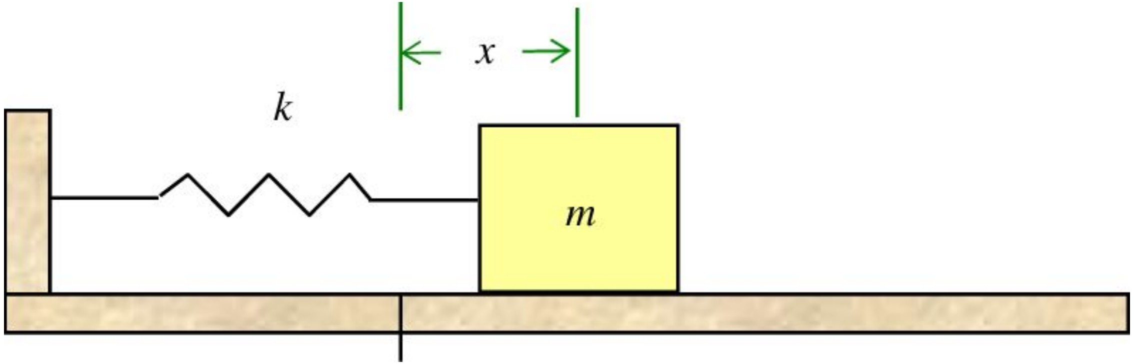
Existen múltiples enfoques según la **naturaleza del sistema** que se desea simular: sistemas discretos, continuos, estocásticos o híbridos:

- Simulación continua basada en ecuaciones diferenciales.
- Simulación basada en eventos discretos (**DES**, Discrete Event Simulation).
- Simulación mediante el método Monte Carlo (**MC**).
- Simulación multiagente y basada en redes.

Índice

2. Simulación continua basada en EDO

- Solución numérica (método de euler): $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot f(\mathbf{x}_k)$



Índice

3. Simulación basada en eventos discretos

- La simulación basada en eventos discretos (DES) es un enfoque en el cual el tiempo avanza de un evento a otro.
  - El estado del sistema solo cambia en los instantes en los que ocurre un evento.
  - Entre un evento y otro, el sistema permanece en un estado constante.

- Ejemplos típicos incluyen:
  - Sistemas de colas en bancos o aeropuertos, procesos de fabricación, redes de comunicación o sistemas de atención al cliente.

3. Simulación basada en eventos discretos

Un modelo típico de simulación discreta se compone de los siguientes elementos:

- **Reloj de simulación**: variable que representa el tiempo actual de la simulación. No avanza de manera continua, sino que salta de un evento a otro.
- **Lista de eventos futuros (LEF)**: estructura que contiene los eventos programados junto con el tiempo en que deben ocurrir. La simulación siempre ejecuta el evento más próximo en el tiempo.
- **Eventos**: sucesos que causan un cambio de estado en el sistema. Ejemplos: llegada de un cliente, inicio o fin de servicio, avería de una máquina.
- **Entidades**: elementos que fluyen a través del sistema (clientes, productos, mensajes, etc.).
- **Recursos**: elementos que atienden o procesan las entidades (empleados, servidores, máquinas…).
- **Colas**: estructuras que almacenan las entidades que esperan ser atendidas.

Formalización del avance temporal

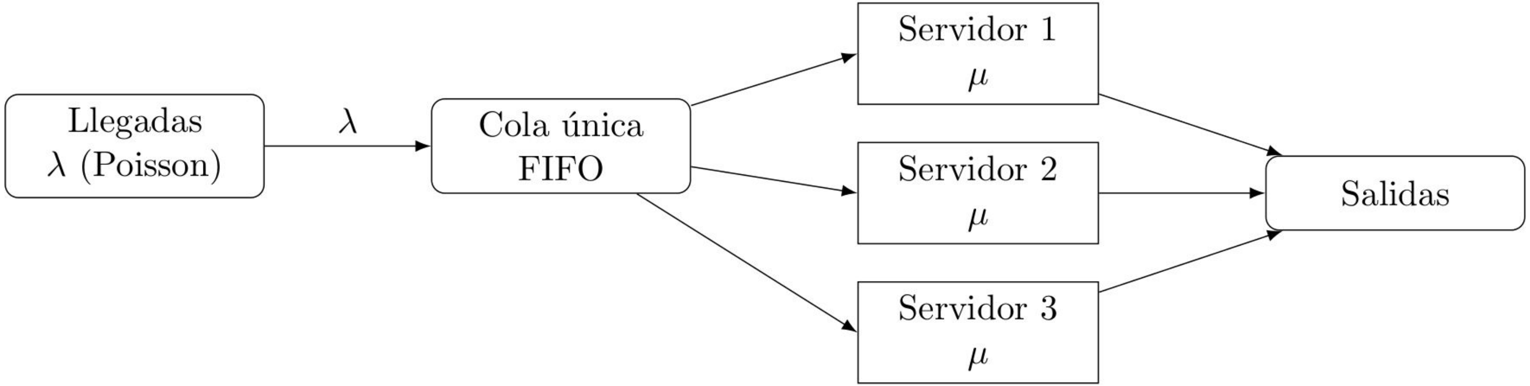
Si designamos t_e como el tiempo del próximo evento, el reloj de simulación avanza de t a t_e y el estado del sistema $S(t)$ se actualiza mediante una función de transición discreta:

$$S(t_e^+) = f(S(t_e^-), e),$$

donde $S(t_e^-)$ es el estado inmediatamente antes del evento, $S(t_e^+)$ el estado después del evento y e representa el tipo de evento ocurrido.

Índice

3.a. Modelo M/M/c y la fórmula de Erlang C



Erlang C (modelo analítico)

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!(1-\rho)} \right]^{-1}$$

$$P_w = \frac{a^c}{c!(1-\rho)} P_0, \quad W_q = \frac{P_w}{c\mu-\lambda}$$

$$W = W_q + \frac{1}{\mu}, \quad L_q = \lambda W_q, \quad L = \lambda W$$

3.a. Modelo M/M/c y la fórmula de Erlang C

El modelo **M/M/c** es uno de los más clásicos en la teoría de colas y sirve como referencia para comparar resultados de simulación con soluciones analíticas. Se trata de un sistema con:

- Llegadas al sistema que siguen un proceso de Poisson con tasa λ (tiempos entre llegadas distribuidos exponencialmente).
- Tiempos de servicio también exponenciales con tasa μ.
- c servidores en paralelo que atienden una cola común con disciplina *FIFO* (First In, First Out).
- Capacidad ilimitada y población infinita de clientes.

3.a. Modelo M/M/c y la fórmula de Erlang C

Relación entre la tasa de llegadas λ y el tiempo medio entre llegadas. En un modelo de colas tipo $M/M/c$, las llegadas siguen un proceso de Poisson con tasa λ (llegadas por unidad de tiempo).

Los *tiempos entre llegadas* siguen entonces una distribución exponencial con media:

$$\mathbb{E}[T_{\text{entre}}] = \frac{1}{\lambda}.$$

Por lo tanto, si en la simulación se define el parámetro `mean_interarrival` como el tiempo medio entre llegadas, la tasa de llegadas utilizada en las fórmulas analíticas debe calcularse como:

$$\lambda = \frac{1}{\text{mean\_interarrival}}.$$

Por ejemplo, si en promedio llega un cliente cada 1.8 minutos, entonces:

$$\lambda = \frac{1}{1.8} \approx 0.5556 \text{ llegadas/minuto}.$$

3.a. Modelo M/M/c y la fórmula de Erlang C

El comportamiento del sistema depende de la **intensidad de tráfico**:

$$\rho = \frac{\lambda}{c\,\mu}$$

que representa el grado de utilización promedio de los servidores. Para que el sistema sea estable, debe cumplirse $\rho < 1$.

3.a. Modelo M/M/c y la fórmula de Erlang C

Fórmula de Erlang C. La *fórmula de Erlang C* calcula la probabilidad de que un cliente tenga que esperar antes de ser atendido, es decir, que todos los servidores estén ocupados cuando llega un nuevo cliente. Primero se define la probabilidad de que el sistema esté vacío:

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!\,(1-\rho)} \right]^{-1}, \qquad \text{donde } a = \frac{\lambda}{\mu}.$$

A partir de P_0, la probabilidad de que un cliente deba esperar en la cola (todos los servidores ocupados) es:

$$P_w = \frac{a^c}{c!\,(1-\rho)}\, P_0.$$

3.a. Modelo M/M/c y la fórmula de Erlang C

La espera media en cola (W_q) y en el sistema completo (W) se calculan mediante:

$$W_q = \frac{P_w}{c\mu - \lambda}, \qquad W = W_q + \frac{1}{\mu}.$$

Finalmente, los números medios de clientes en cola y en el sistema son:

$$L_q = \lambda W_q, \qquad L = \lambda W.$$

3.a. Modelo M/M/c y la fórmula de Erlang C

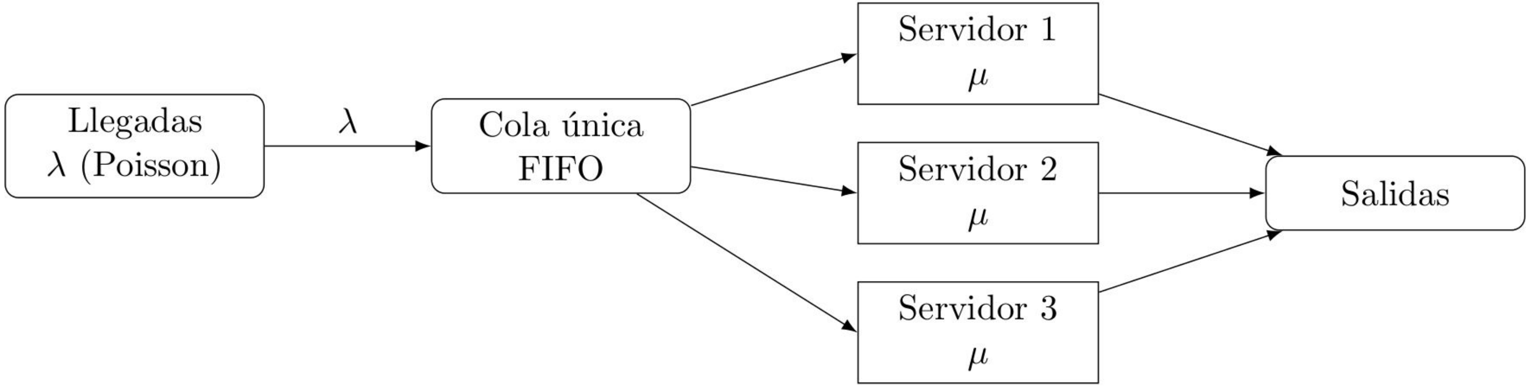
Interpretación:

- W_q mide el tiempo medio que un cliente pasa esperando antes de ser atendido.
- W incluye tanto el tiempo de espera como el tiempo de servicio.
- L_q y L son las longitudes medias de la cola y del sistema, respectivamente.
- La fórmula de Erlang C permite **predecir el rendimiento del sistema** sin necesidad de simulación, sirviendo como *modelo analítico de referencia*.

En la práctica, se utiliza para comparar los resultados de simulaciones de colas (por ejemplo, mediante SimPy o AnyLogic) con las métricas teóricas derivadas del modelo M/M/c.

3.a. Modelo M/M/c y la fórmula de Erlang C

UA
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior



Erlang C (modelo analítico)

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!(1-\rho)}\right]^{-1}$$

$$P_w = \frac{a^c}{c!(1-\rho)} P_0, \quad W_q = \frac{P_w}{c\mu - \lambda}$$

$$W = W_q + \frac{1}{\mu}, \quad L_q = \lambda W_q, \quad L = \lambda W$$

3.a. Modelo M/M/c y la fórmula de Erlang C

UA
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

- Para eventos discretos:
https://simpy.readthedocs.io/en/latest/

SimPy es una biblioteca de Python para *simulación basada en eventos discretos* (DES). Su idea central es modelar **procesos** como *corutinas* (funciones generadoras) que van *cediendo* el control al motor de simulación mediante `yield` de **eventos** como `timeout` (paso del tiempo), `request`/`release` (uso de recursos), o señales de sincronización (`Event`, `AllOf`, `AnyOf`). El **reloj de simulación** avanza al tiempo del *próximo evento*.

3.a. Modelo M/M/c y la fórmula de Erlang C

Métricas principales. Entre las medidas de desempeño más habituales se encuentran:

- **Utilización de los servidores:** proporción de tiempo en que los servidores están ocupados, $\rho = \frac{\lambda}{c\mu}$.
- **Tiempo medio en cola:** W_q, tiempo promedio que los clientes esperan antes de ser atendidos.
- **Tiempo total en el sistema:** $W = W_q + \frac{1}{\mu}$.
- **Número medio en cola:** $L_q = \lambda W_q$.
- **Número medio en el sistema:** $L = \lambda W$.

Índice

UA
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

4. Simulación con el método Monte Carlo

- La simulación Monte Carlo (**MC**) es un paradigma de simulación basado en la generación de **números aleatorios** para modelar fenómenos inciertos o complejos.
  - El nombre proviene del famoso casino de Monte Carlo, debido al papel central del azar en el método.

- En esencia, consiste en **repetir muchas veces un experimento aleatorio** para aproximar un valor esperado, una probabilidad, o una integral difícil de resolver analíticamente.

4. Simulación con el método Monte Carlo

Idea básica. Si queremos estimar un valor esperado de una función $f(X)$ de una variable aleatoria X, se puede aproximar por el promedio muestral de N simulaciones independientes:

$$\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i),$$

donde X_i son valores generados al azar según la distribución de X.

Cuando N es grande, por el *teorema de los grandes números*, la media muestral converge al valor esperado real.

4. Simulación con el método Monte Carlo

Ventajas del método Monte Carlo.

- Permite resolver problemas sin solución analítica cerrada.
- Es flexible y aplicable a cualquier tipo de distribución o sistema estocástico.
- La precisión mejora con el número de simulaciones (error $\sim 1/\sqrt{N}$).
- Es muy útil en física, finanzas, fiabilidad, optimización y estadística.

4. Simulación con el método Monte Carlo

Limitaciones.

- Requiere muchas simulaciones para obtener alta precisión.
- No siempre ofrece información determinista: sólo estimaciones probabilísticas.
- Los resultados dependen de la calidad del generador de números aleatorios.

Índice

4.a. Estimación de Pi

Uno de los ejemplos clásicos de Monte Carlo es la estimación de π mediante muestreo aleatorio.

Se considera un cuadrado de lado 2 que encierra un círculo de radio 1. Si se generan puntos aleatorios (x, y) uniformemente distribuidos en el cuadrado, la probabilidad de que un punto caiga dentro del círculo es igual al cociente de sus áreas:

$$P = \frac{\text{Área del círculo}}{\text{Área del cuadrado}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}.$$

Entonces:

$$\pi \approx 4 \times \frac{n_{\text{dentro}}}{N},$$

donde n_{dentro} es el número de puntos que satisfacen $x^2 + y^2 \leq 1$.

4.a. Estimación de Pi

Ejemplo numérico. Supongamos que se generan $N = 100{,}000$ puntos aleatorios y 78,520 caen dentro del círculo:

$$\pi \approx 4 \times \frac{78{,}520}{100{,}000} = 3.1408.$$

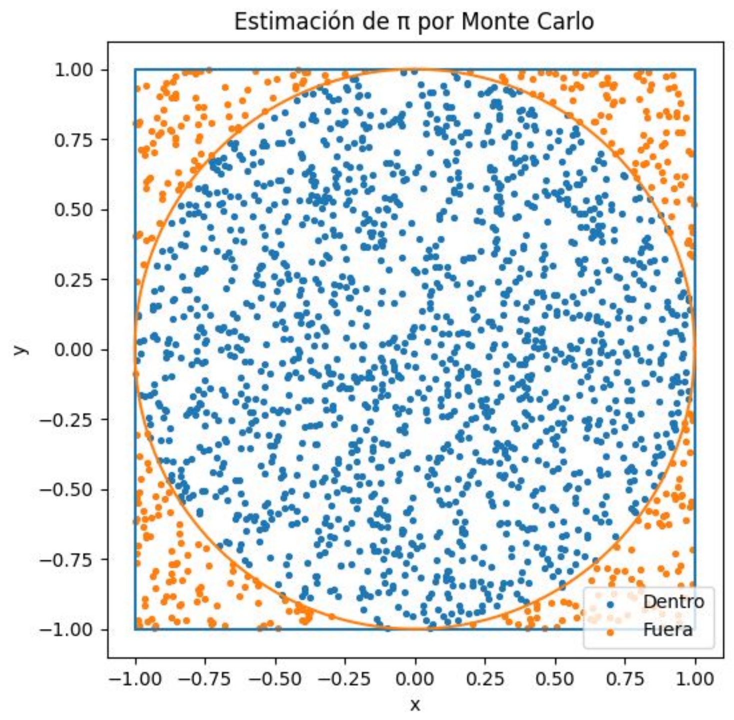
Este valor ya es una buena aproximación de $\pi \approx 3.1416$.

A medida que se incrementa N, el error medio cuadrático disminuye aproximadamente como $1/\sqrt{N}$.

$$\text{Error} \propto \frac{1}{\sqrt{N}}.$$

4.a. Estimación de Pi

UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior



Índice

UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

4.b. Propagación de incertidumbre

- Modelo cinemático robot móvil:

$$x_{k+1} = x_k + v_k \cos(\theta_k)\, \Delta t,$$

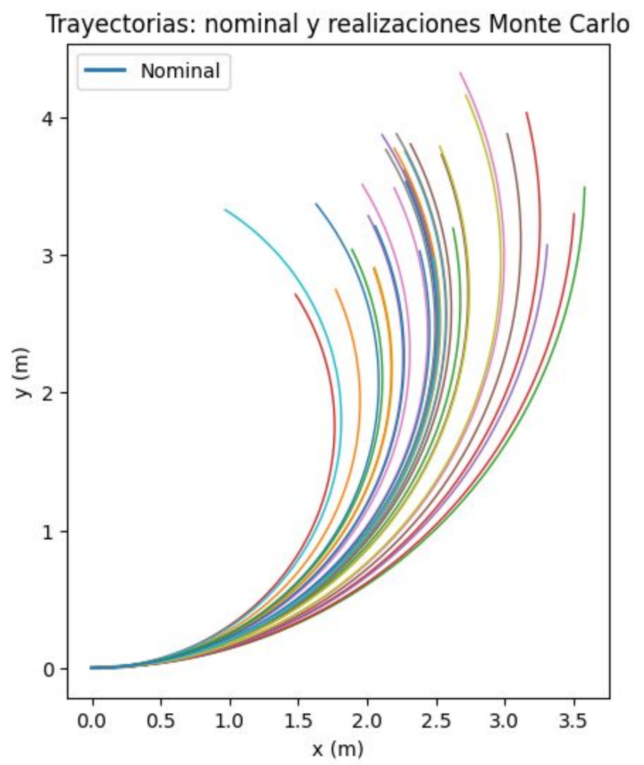
$$y_{k+1} = y_k + v_k \sin(\theta_k)\, \Delta t,$$

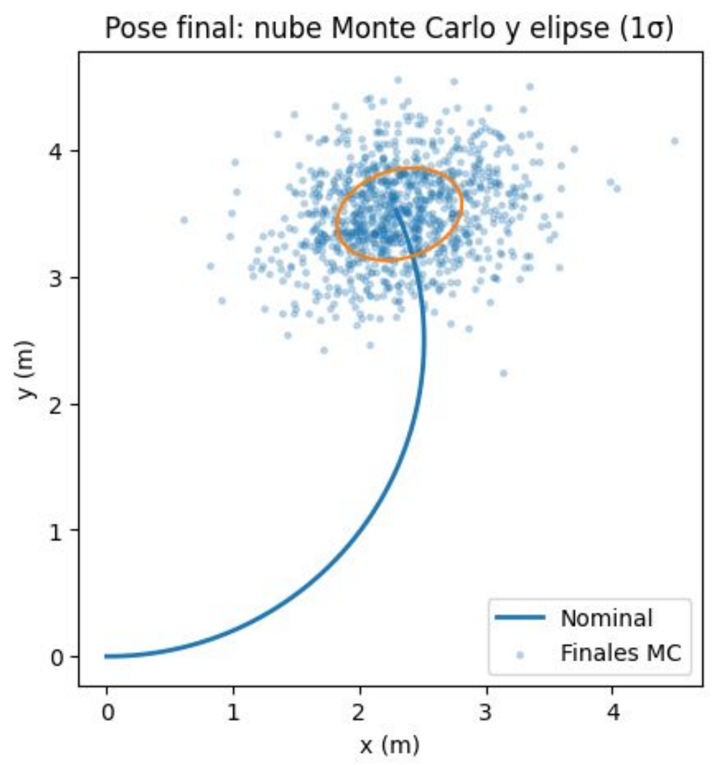
$$\theta_{k+1} = \theta_k + \omega_k\, \Delta t.$$

- Con incertidumbre modelada como:

$$v_k \sim \mathcal{N}(\bar{v}, \sigma_v^2), \qquad \omega_k \sim \mathcal{N}(\bar{\omega}, \sigma_\omega^2).$$

4.b. Propagación de incertidumbre





UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

Índice

UA
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

Estructura básica de un modelo multiagente

- Conjunto de agentes $A = \{a_1, a_2, \ldots, a_N\}$.
- Entorno E en el que se mueven o interactúan.
- Funciones de percepción, decisión y acción: $a_i : E \to A_i$.
- Reglas de interacción entre agentes y entorno.

Índice

6. Software de simulación

Entorno	**Paradigma principal**
Simulink (MATLAB)	Continuo / híbrido
AnyLogic	Multi-paradigma (eventos, agentes, continuo)
NetLogo	Basado en agentes
Arena / SimEvents	Eventos discretos

6. Software de simulación

Paradigma	Bibliotecas recomendadas
Eventos discretos	`simpy`
Continuo / ecuaciones diferenciales	`scipy.integrate`, `numpy`, `matplotlib`
Monte Carlo	`numpy`, `random`, `pandas`
Basado en agentes	`mesa`, `networkx`
Basado en redes	`networkx`, `igraph`

TEMA 3: Paradigmas de simulación

MODELOS COMPUTACIONALES Y SIMULACIÓN DE SISTEMAS
Curso 2025-2026

UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Escola Politècnica Superior
Escuela Politécnica Superior

DFESTS