

Introduzione alla Simulazione Discreta

Moreno Marzolla

`moreno.marzolla@pd.infn.it`

`http://www.dsi.unive.it/~marzolla/`

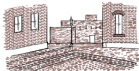
INFN Sezione di Padova

25 giugno 2008

Architettura Universitaria...

ARCHITECTURAL STYLES OF CONTEMPORARY UNIVERSITIES

A guide to the richness of design that you probably ignore on your way to work as you walk with your head down absorbed in thought.



Red Brick Wonderland (c. 1800's)
There are more bricks on campus than books. Not recommended: wearing red at busy intersections.



Modernist "It cost HOW much?" (1990's-present)
Sure, they could have built three buildings for the same price, but those buildings would be functional and non-leaking.

Window details:



WWW.PHDCOMICS.COM



"Gothic Envyist" (c. 1600-1700's)
Nothing says "We're just like Oxford and Cambridge!" like cheerful gothic architecture from the middle ages.



Neo-Penal (1960-1970's)
Influenced by nouveau prison design and the need to quell student riots, concrete structures of this period have the unique ability to drain your very Life Force.

The Campus Tower

No respectable institution of higher learning would be complete without a towering symbol of ecclesiastic domination (also makes a good tourist photo-op).



Finally...

A Designed Environment

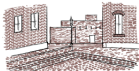
Despite differences in styles, all academic architecture has the same purpose: to create an environment that inspires and fosters learning and intellectual discourse—convince alumni and parents they're getting their money's worth.

<http://www.phdcomics.com/comics/archive.php?comicid=999>

Architettura Universitaria...

ARCHITECTURAL STYLES OF CONTEMPORARY UNIVERSITIES

A guide to the richness of design that you probably ignore on your way to work as you walk with your head down absorbed in thought.



Red Brick Wonderland (c. 1800's)
There are more bricks on campus than books. Not recommended: wearing red at busy intersections.



Modernist "It cost HOW much?"
(1900's-present)
Sure, they could have built three buildings for the same price, but those buildings would be functional and non-leaking.

Window details:



WWW.PHDCOMICS.COM



"Gothic Envyism" (c. 1600-1700's)
Nothing says "We're just like Oxford and Cambridge!" like cheerful gothic architecture from the middle ages.



Neo-Penal (1960-1970's)
Influenced by nouveau prison design and the need to quell student riots, concrete structures of this period have the unique ability to *drain your very Life Force*.

The Campus Tower

No respectable institution of higher learning would be complete without a towering symbol of ecclesiastic domination (also makes a good tourist photo-op).



Finally...

A Designed Environment

Despite differences in styles, all academic architecture has the same purpose: to **create an environment that inspires and fosters learning and intellectual discourse**—convince alumni and parents they're getting their money's worth.

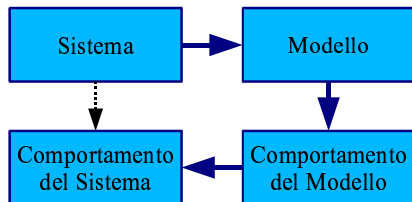


Neo-Penal (1960-1970's)
Influenced by nouveau prison design and the need to quell student riots, concrete structures of this period have the unique ability to *drain your very Life Force*.

<http://www.phdcomics.com/comics/archive.php?comicid=999>

- 1 Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output
- 5 Conclusioni
- 6 Bibliografia

- 1** Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output
- 5 Conclusioni
- 6 Bibliografia



- Si vuole analizzare un certo **sistema fisico** (es., una CPU, una rete di calcolatori, un sistema software...).
 - Il sistema reale potrebbe essere troppo complesso da analizzare nel dettaglio
 - Potrebbe non esistere ancora
 - Potrebbe non essere possibile o troppo oneroso condurre prove
- Si crea un **modello del sistema** come rappresentazione *semplificata* che includa le caratteristiche più importanti
- L'analisi del modello deve fornire informazioni sul al comportamento del sistema reale

Variabili di Stato

Definizione

Le **Variabili di Stato** rappresentano lo stato in cui si trova il sistema in un particolare istante di tempo.

Esempio Supponiamo di voler simulare un server con una coda di richieste associate.



In questo caso lo *stato* del sistema è identificato dal numero di utenti $Q(t)$ nel centro di servizio (area tratteggiata) al tempo t .

Alcuni tipi di Simulazione

Simulazione continua

Nella simulazione continua lo stato del sistema evolve in modo continuo nel tempo.

Simulazione MonteCarlo

Le simulazioni di tipo MonteCarlo sono utilizzate per modellare fenomeni che non dipendono dal tempo.

Simulazione discreta

Nella simulazione discreta lo stato del sistema evolve solo a specifici istanti di tempo.

Alcuni tipi di Simulazione

Simulazione continua

Nella simulazione continua lo stato del sistema evolve in modo continuo nel tempo.

Simulazione MonteCarlo

Le simulazioni di tipo MonteCarlo sono utilizzate per modellare fenomeni che non dipendono dal tempo.

Simulazione discreta

Nella simulazione discreta lo stato del sistema evolve solo a specifici istanti di tempo.

Alcuni tipi di Simulazione

Simulazione continua

Nella simulazione continua lo stato del sistema evolve in modo continuo nel tempo.

Simulazione MonteCarlo

Le simulazioni di tipo MonteCarlo sono utilizzate per modellare fenomeni che non dipendono dal tempo.

Simulazione discreta

Nella simulazione discreta lo stato del sistema evolve solo a specifici istanti di tempo.

Alcuni tipi di Simulazione

Simulazione continua

Nella simulazione continua lo stato del sistema evolve in modo continuo nel tempo.

Simulazione MonteCarlo

Le simulazioni di tipo MonteCarlo sono utilizzate per modellare fenomeni che non dipendono dal tempo.

Simulazione discreta

Nella simulazione discreta lo stato del sistema evolve solo a specifici istanti di tempo.

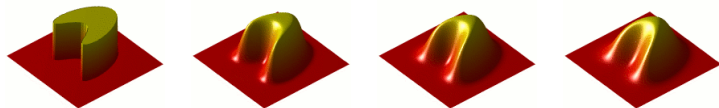
Simulazione Continua

Esempio

Equazione di trasferimento del calore

$$\frac{\partial u}{\partial t} - k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) = 0$$

con $u(x, y, z, t)$ temperatura in (x, y, z) al tempo t .

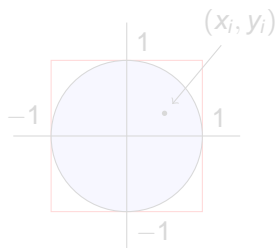


Fonte: http://en.wikipedia.org/wiki/Heat_equation

Simulazione MonteCarlo

La simulazione di tipo **MonteCarlo** viene utilizzata per modellare fenomeni probabilistici che non dipendono dal tempo.

Esempio: Vogliamo calcolare l'area del cerchio senza conoscere il valore di π .



1 Generiamo n coppie di numeri (x_i, y_i) uniformemente distribuiti in $[-1, 1] \times [-1, 1]$

2 Sia

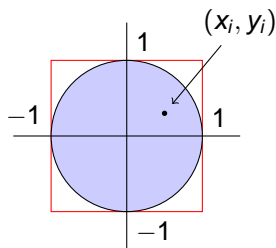
$$C_i = \begin{cases} 1 & \text{se } x_i^2 + y_i^2 \leq 1 \\ 0 & \text{altrimenti} \end{cases} \quad i = 1, 2, \dots, n$$

3 $Area \approx 4 \frac{1}{n} \sum_{i=1}^n C_i$

Simulazione MonteCarlo

La simulazione di tipo **MonteCarlo** viene utilizzata per modellare fenomeni probabilistici che non dipendono dal tempo.

Esempio: Vogliamo calcolare l'area del cerchio senza conoscere il valore di π .



1 Generiamo n coppie di numeri (x_i, y_i) uniformemente distribuiti in $[-1, 1] \times [-1, 1]$

2 Sia

$$C_i = \begin{cases} 1 & \text{se } x_i^2 + y_i^2 \leq 1 \\ 0 & \text{altrimenti} \end{cases} \quad i = 1, 2, \dots, n$$

3 $Area \approx 4 \frac{1}{n} \sum_{i=1}^n C_i$

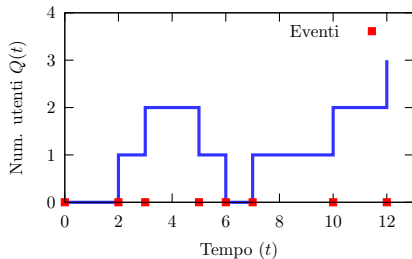
Simulazione Discreta

Esempio

Riprendiamo l'esempio del server con la coda associate:



t	Evento	$Q(t)$
0	Inizio	0
2	Arrivo ₁	1
3	Arrivo ₂	2
5	Complet ₁	1
6	Complet ₂	0
7	Arrivo ₃	1
10	Arrivo ₄	2
12	Arrivo ₅	3
...



(Alcuni) Errori Comuni

- Mancanza di uno scopo chiaro
- Livello di dettaglio inappropriato
- Mancata verifica del modello
- Mancata validazione del modello
- Condizioni iniziali gestite in modo inappropriato
- Generatori pseudocasuali inappropriati

Mancanza di uno scopo chiaro

- Sviluppare una simulazione complessa richiede **risorse** e una notevole **esperienza**. È indispensabile avere le idee chiare prima di investire tempo
- Quale scopo si prefigge la simulazione?
- Cosa intendiamo studiare?
- Che risultati ci aspettiamo?
- A cosa serviranno i risultati che otterremo?

Livello di Dettaglio Inappropriato

Si è portati a ritenere che un modello di simulazione più dettagliato sia **sempre** preferibile ad un modello più semplice, perché si basa su meno assunzioni.

Questo non è sempre vero:

- Modelli dettagliati richiedono tempi di sviluppo più lunghi
- È maggiore la probabilità di introdurre errori
- Possono richiedere un maggior dettaglio nei parametri di input, che potrebbe non essere disponibile
 - **Garbage In, Garbage Out**¹
 - I. Asimov, *La Macchina che Vinse la Guerra*
<http://xoomer.alice.it/mstangal/macchina.html>

Suggerimento

Partire con un modello semplice, e raffinarlo solo se necessario.

¹http://en.wikipedia.org/wiki/Garbage_in,_garbage_out

Livello di Dettaglio Inappropriato

Si è portati a ritenere che un modello di simulazione più dettagliato sia **sempre** preferibile ad un modello più semplice, perché si basa su meno assunzioni.

Questo non è sempre vero:

- Modelli dettagliati richiedono tempi di sviluppo più lunghi
- È maggiore la probabilità di introdurre errori
- Possono richiedere un maggior dettaglio nei parametri di input, che potrebbe non essere disponibile
 - **Garbage In, Garbage Out**¹
 - I. Asimov, *La Macchina che Vinse la Guerra*

<http://xoomer.alice.it/mstangal/macchina.html>

Suggerimento

Partire con un modello semplice, e raffinarlo solo se necessario.

¹http://en.wikipedia.org/wiki/Garbage_in,_garbage_out ◀ ▶ ☰ ☱ ☲ ☳ ☴ ☵ ☶ ☷

Livello di Dettaglio Inappropriato

Si è portati a ritenere che un modello di simulazione più dettagliato sia **sempre** preferibile ad un modello più semplice, perché si basa su meno assunzioni.

Questo non è sempre vero:

- Modelli dettagliati richiedono tempi di sviluppo più lunghi
- È maggiore la probabilità di introdurre errori
- Possono richiedere un maggior dettaglio nei parametri di input, che potrebbe non essere disponibile
 - **Garbage In, Garbage Out**¹
 - I. Asimov, *La Macchina che Vinse la Guerra*

<http://xoomer.alice.it/mstangal/macchina.html>

Suggerimento

Partire con un modello semplice, e raffinarlo solo se necessario.

¹http://en.wikipedia.org/wiki/Garbage_in,_garbage_out

Mancata verifica del modello

Le simulazioni sono generalmente programmi complessi e di grosse dimensioni, e come tali possono contenere **errori nel codice**.

Suggerimenti

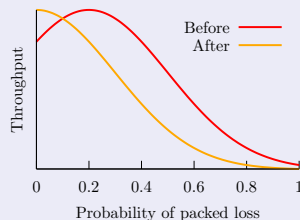
- Scrivere il codice in modo **strutturato** e **modulare**
- Scrivere **unit test** ove possibile
- Eseguire il codice su esempi semplici di cui si conosca (o si possa determinare facilmente) il risultato esatto

Mancata validazione del modello

Anche se il software non contiene errori, i risultati della simulazione potrebbero comunque essere non attendibili perché si basano su **assunzioni sbagliate**.

Suggerimenti

- Intuito
- Confrontare i risultati della simulazione con misure condotte su sistemi reali
- Confrontare i risultati della simulazione con risultati ottenuti per via analitica (es, usando modelli a reti di code)



Condizioni iniziali gestite in modo inappropriato

Nella maggior parte delle simulazioni si è interessati a studiare il comportamento del sistema nello **stato stazionario**, ossia nello stato stabile raggiunto quando il sistema è in esecuzione per un tempo sufficientemente lungo.

La parte iniziale della simulazione è chiamata **transiente**, e come vedremo in seguito **non** è in genere rappresentativa dello stato stazionario.

Suggerimenti

- Effettuare simulazioni lunghe
- Inizializzare la simulazione in modo opportuno
- Rimuovere parte dei dati iniziali dell'output

Generatori Pseudocasuali inappropriati

Le simulazioni necessitano di generatori pseudocasuali per ottenere parametri da utilizzare (es., tempi di risposta, tempi di arrivo delle richieste, ecc.).

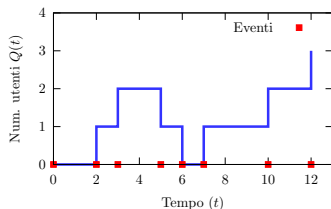
La qualità dei risultati ottenuti dipende in maniera **fondamentale** dalla qualità dei generatori pseudocasuali adottati.

Suggerimenti

- Chiedersi **sempre** che tipo di generatore pseudocasuale si sta utilizzando, e che proprietà abbia
- Non inizializzare il generatore con semi casuali (ad es, ottenuti dall'ora corrente). Le simulazioni **devono essere ripetibili**

- 1 Introduzione
- 2 Simulazione Discreta**
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output
- 5 Conclusioni
- 6 Bibliografia

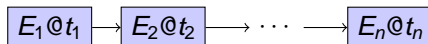
Simulazione ad Eventi



In una simulazione ad eventi lo **stato** del sistema **cambia solo in determinati istanti di tempo** (eventi).

Può essere implementata mediante una lista di eventi (**Future Event List** o **FEL**) mantenuta ordinata in base all'istante di occorrenza.

Future Event List (FEL)



L'evento E_i viene eseguito al tempo t_i .

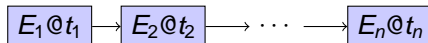
La lista viene mantenuta ordinata in senso non decrescente in base al timestamp di occorrenza degli eventi ($t_1 \leq t_2 \leq \dots \leq t_n$).

Un evento può

- Modificare lo stato della simulazione
- Schedulare uno o più nuovi eventi per il futuro
- Rimuovere uno o più eventi già schedulati

La simulazione consiste semplicemente nello scorrere la FEL ed eseguire gli eventi a partire dal primo.

Future Event List (FEL)



L'evento E_i viene eseguito al tempo t_i .

La lista viene mantenuta ordinata in senso non decrescente in base al timestamp di occorrenza degli eventi ($t_1 \leq t_2 \leq \dots \leq t_n$).

Un evento può

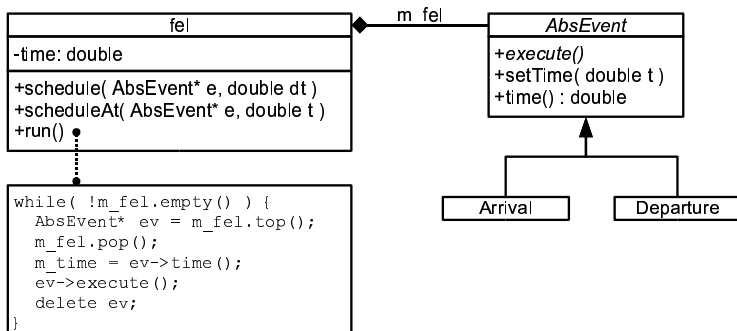
- Modificare lo stato della simulazione
- Schedulare uno o più nuovi eventi per il futuro
- Rimuovere uno o più eventi già schedulati

La simulazione consiste semplicemente nello scorrere la FEL ed eseguire gli eventi a partire dal primo.

Esempio

t	Evento	FEL Dopo
0	Inizio	$A_1@2 \rightarrow A_2@3 \rightarrow A_3@7 \rightarrow A_4@10 \rightarrow A_5@12$
2	Arrivo ₁	$A_2@3 \rightarrow C_1@5 \rightarrow A_3@7 \rightarrow A_4@10 \rightarrow A_5@12$
3	Arrivo ₂	$C_1@5 \rightarrow A_3@7 \rightarrow A_4@10 \rightarrow A_5@12$
5	Complet ₁	$C_2@6 \rightarrow A_3@7 \rightarrow A_4@10 \rightarrow A_5@12$
6	Complet ₂	$A_3@7 \rightarrow A_4@10 \rightarrow A_5@12$
7	Arrivo ₃	...
...

Schema di un simulatore ad eventi



```
class AbsEvent {
public:
    // ...
    virtual void execute( void ) = 0;
    double time( void ) const { return m_time; };
    void setTime( double t ) { m_time = t; };
protected:
    AbsEvent( ) : m_time( 0 ) { };
    double m_time; // Istante in cui eseguire l'evento
};
```

```
class Fel {
public:
    // ...
    // Schedula l'evento ev al tempo now()+dt
    void schedule( AbsEvent* ev, double dt );
    // Schedula l'evento ev al tempo (assoluto) t
    void scheduleAt( AbsEvent* ev, double t );
    // Ritorna il tempo corrente della simulazione
    double time( void ) const { return m_time; };
    void run();
protected:
    struct ev_compare {
        bool operator()( const AbsEvent* e1, const AbsEvent* e2 ) const {
            return (e1->time() > e2->time());
        };
    };
    std::priority_queue<
        AbsEvent*,
        std::vector< AbsEvent* >,
        Fel::ev_compare
    > m_fel;
    double m_time;
};
```

```
void Fel::run( void )
{
    while( !m_fel.empty() ) {
        AbsEvent* e( m_fel.top() ); m_fel.pop();
        m_time = e->time( );
        e->execute( );
        delete e;
    }
}

void Fel::scheduleAt( AbsEvent* e, double t )
{
    e->setTime( t );
    m_fel.push( e );
}

void Fel::schedule( AbsEvent* e, double dt )
{
    e->setTime( time()+dt );
    m_fel.push( e );
}
```

```
typedef struct job {  
public:  
    job( int i ) : id( i ), arrival_time(0), completion_time(0) { };  
  
    int id;  
    double arrival_time;  
    double completion_time;  
} job;
```

Simulatore di Centro di Servizio

Implementazione metodo `evArrival::execute()`

```
void evArrival::execute( void )
{
    m_job->arrival_time = time();
    if ( m_server->empty() ) {
        AbsEvent* ev( new evComplete( m_fel, m_server ) );
        double t=m_server->servTime();
        m_fel->schedule( ev, t );
    }
    m_server->addJob( m_job );
};
```

Simulatore di Centro di Servizio

Implementazione metodo `evComplete::execute()`

```
void evComplete::execute( void )
{
    job* j = m_server->removeJob();
    j->completion_time = time();
    cout << j->completion_time - j->arrival_time << endl;
    delete j;
    if ( !m_server->empty() ) {
        double t=m_server->servTime();
        AbsEvent* ev( new evComplete( m_fel, m_server ) );
        m_fel->schedule( ev, t );
    }
}
```

Simulatore di Centro di Servizio

Avvio della Simulazione

```
// ...
rng* arrival_time = new rngExp( "Arrival", 1.0/arr_rate );
double t = arrival_time->value();
// Genera 1000 arrivi
for ( int i=0; i<n_arrivals; ++i ) {
    AbsEvent* ev = new evArrival( &the_fel, the_server, new job( i ) );
    the_fel.scheduleAt( ev, t );
    t += arrival_time->value();
}
the_fel.run();
```

- 1 Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali**
- 4 Analisi dell'Output
- 5 Conclusioni
- 6 Bibliografia

Generatori Pseudocasuali

Uno degli ingredienti fondamentali di una simulazione è la possibilità di **generare sequenze di numeri pseudocasuali con una distribuzione specificata**.

Per realizzare ciò, si procede in due passi:

- 1 Si genera una sequenza di numeri pseudocasuali uniformemente distribuiti nell'intervallo $[0, 1]$;
- 2 La sequenza ottenuta al punto 1 è trasformata in modo da ottenere una nuova sequenza con la distribuzione richiesta.

Generatori Pseudocasuali

Uno degli ingredienti fondamentali di una simulazione è la possibilità di **generare sequenze di numeri pseudocasuali con una distribuzione specificata**.

Per realizzare ciò, si procede in due passi:

- 1 Si genera una sequenza di numeri pseudocasuali uniformemente distribuiti nell'intervallo $[0, 1]$;
- 2 La sequenza ottenuta al punto 1 è trasformata in modo da ottenere una nuova sequenza con la distribuzione richiesta.

Generatori Pseudocasuali

Uno degli ingredienti fondamentali di una simulazione è la possibilità di **generare sequenze di numeri pseudocasuali con una distribuzione specificata**.

Per realizzare ciò, si procede in due passi:

- 1 Si genera una sequenza di numeri pseudocasuali uniformemente distribuiti nell'intervallo $[0, 1]$;
- 2 La sequenza ottenuta al punto 1 è trasformata in modo da ottenere una nuova sequenza con la distribuzione richiesta.

Il modo più comune di definire numeri (pseudo)casuali è mediante relazioni ricorrenti che calcolano l' n -esimo numero noti alcuni dei precedenti:

$$x_n = f(x_{n-1}, x_{n-2}, \dots)$$

Esempio

$$x_n = 5x_{n-1} + 1 \pmod{16} \quad (1)$$

In questo caso $x_i \in \{0, 1, \dots, 15\}$, da cui $x_i/15 \in [0, 1]$.

Esempio

Primi 48 numeri della sequenza, partendo da $x_0 = 5$

esempio_rng.m

```
function x_n = esempio_rng( x_nml )
    x_n = mod(5 * x_nml + 1, 16);
endfunction

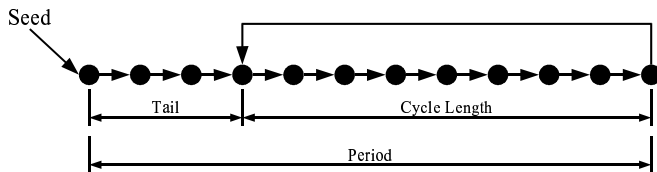
x_nml = 5;
for i=1:48
    x_n = esempio_rng( x_nml );
    printf("%d_",x_n);
    x_nml = x_n;
endfor
printf("\n");
```

Output

```
10 3 0 1 6 15 12 13 2 11 8 9 14 7 4 5
10 3 0 1 6 15 12 13 2 11 8 9 14 7 4 5
10 3 0 1 6 15 12 13 2 11 8 9 14 7 4 5
```

Definizioni

- Il valore iniziale x_0 da cui partire con la sequenza si chiama **seme** (*seed*)
- Dato il seme, il comportamento del generatore è totalmente deterministico. Si parla quindi di **generatori pseudo-casuali**.
- Il **periodo** un un RNG si compone di una parte iniziale detta **coda** seguita da un **ciclo** che si ripete.



Nell'esempio precedente la **coda ha lunghezza 0**, e il **ciclo è lungo 16**

Caratteristiche di un buon RNG

1 Deve essere facile da valutare.

La funzione $f()$ di cui sopra deve essere efficientemente calcolabile, visto che in genere in una simulazione possono essere richiesti molti numeri pseudocasuali.

2 Il periodo deve essere lungo.

Un periodo breve fa sì che la sequenza si ripeta spesso, generando correlazione tra gli eventi della simulazione e quindi riducendone la durata massima.

3 I valori della sequenza devono essere indipendenti e uniformemente distribuiti.

La correlazione tra numeri successivi nella sequenza deve essere piccola.

Test di indipendenza

Test Spettrale

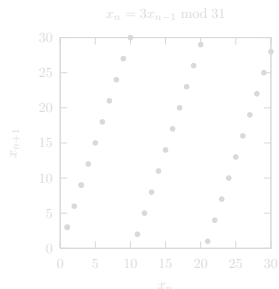
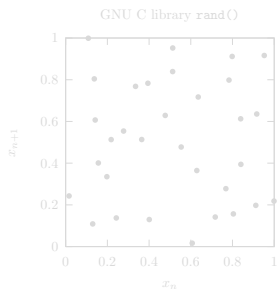
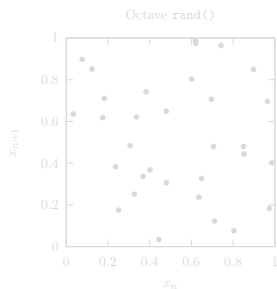
- Esistono numerosi test statistici per decidere se i valori di una sequenza sono indipendenti e uniformemente distribuiti.
- Uno di questi è il **Test Spettrale**: in dimensione k richiede di calcolare in che modo le k -tuple di valori successivi $(x_n, x_{n+1}, \dots, x_{n+k-1})$, $n = 0, 1, \dots$ (*overlapping tuples*) si dispongono nello spazio euclideo a k dimensioni.
- Un “buon” generatore dovrebbe produrre punti disposti uniformemente nello spazio a k dimensioni.

Esempio

Nel caso $k = 2$ si tratta di verificare in che modo i punti di coordinate (x_n, x_{n+1}) , $n = 0, 1, \dots$ si dispongono sul piano.

Test Spettrale

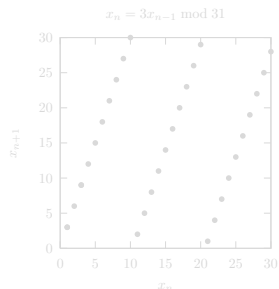
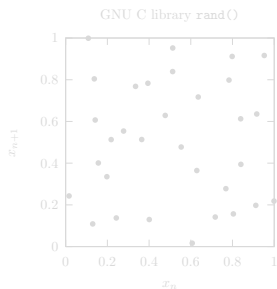
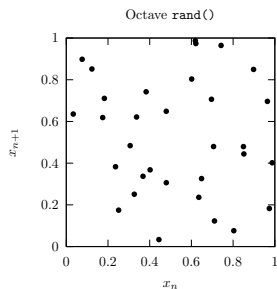
Plottiamo i primi 32 valori della sequenza, $k = 2$



- L'implementazione di GNU Octave usa un generatore con periodo $2^{19937} - 1$ [4].
- L'implementazione usata nella GNU C library fa parte della famiglia dei *Linear Congruential Generators* (LCG); si veda <http://www.mscs.dal.ca/~selinger/random/> per una descrizione comprensibile

Test Spettrale

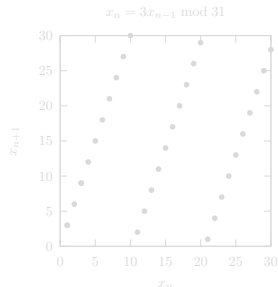
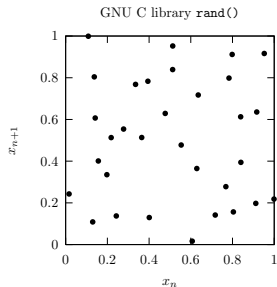
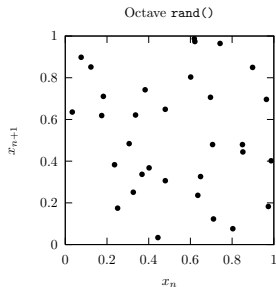
Plottiamo i primi 32 valori della sequenza, $k = 2$



- L'implementazione di GNU Octave usa un generatore con periodo $2^{19937} - 1$ [4].
- L'implementazione usata nella GNU C library fa parte della famiglia dei *Linear Congruential Generators* (LCG); si veda <http://www.mscs.dal.ca/~selinger/random/> per una descrizione comprensibile

Test Spettrale

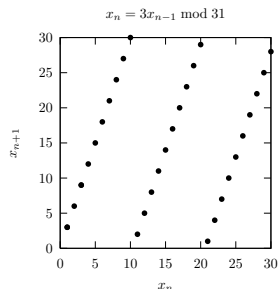
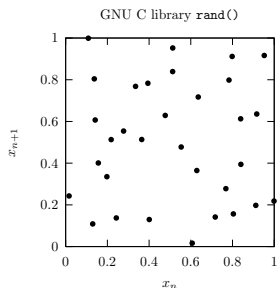
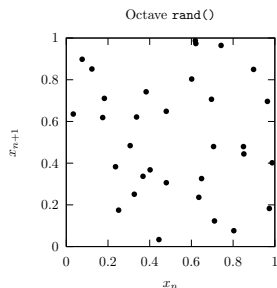
Plottiamo i primi 32 valori della sequenza, $k = 2$



- L'implementazione di GNU Octave usa un generatore con periodo $2^{19937} - 1$ [4].
- L'implementazione usata nella GNU C library fa parte della famiglia dei *Linear Congruential Generators* (LCG); si veda <http://www.mscs.dal.ca/~selinger/random/> per una descrizione comprensibile

Test Spettrale

Plottiamo i primi 32 valori della sequenza, $k = 2$



- L'implementazione di GNU Octave usa un generatore con periodo $2^{19937} - 1$ [4].
- L'implementazione usata nella GNU C library fa parte della famiglia dei *Linear Congruential Generators* (LCG); si veda <http://www.mscs.dal.ca/~selinger/random/> per una descrizione comprensibile

Errore

Usare il timestamp come seme del generatore pseudocasuale

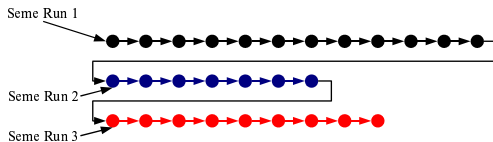
- Usare il timestamp rende le simulazioni non riproducibili!!
- Occorre fare in modo che **ogni volta che si esegue la simulazione** (anche su piattaforme diverse) si ottengano **esattamente** le stesse sequenze di numeri pseudocasuali.

Errori da evitare

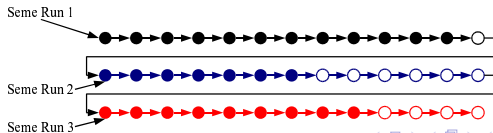
Errore

Usare l'ultimo seme di un run come inizializzazione del run successivo in run che dovrebbero essere indipendenti

- **Mai** fare questo:



- Occorre preallocare sequenze di lunghezza uguale, e sufficiente ad includere la sequenza più lunga richiesta da tutti i run:



Random Variate Generators

Fin qui abbiamo visto come generare valori uniformemente distribuiti nell'intervallo $[0, 1]$. **Come generare valori distribuiti in altro modo?**

Definizione

Una V.A. X ha una *distribuzione* F_X se $F_X(x) = \Pr(X \leq x)$ per ogni $x \in \mathbb{R}$

Teorema

Sia X una V.A. con distribuzione *continua monotona crescente* F_X . Consideriamo la V.A. $U = F_X(X)$ avente distribuzione F_U : allora U è uniformemente distribuita in $[0, 1]$

Infatti si ha:

$$\begin{aligned} F_U(u) &= \Pr(U \leq u) = \Pr(F_X(X) \leq u) \\ &= \Pr\left(X \leq F_X^{-1}(u)\right) = F_X\left(F_X^{-1}(u)\right) \\ &= u \end{aligned}$$

Random Variate Generators

Fin qui abbiamo visto come generare valori uniformemente distribuiti nell'intervallo $[0, 1]$. **Come generare valori distribuiti in altro modo?**

Definizione

Una V.A. X ha una **distribuzione** F_X se $F_X(x) = \Pr(X \leq x)$ per ogni $x \in \mathbb{R}$

Teorema

Sia X una V.A. con distribuzione **continua monotona crescente** F_X . Consideriamo la V.A. $U = F_X(X)$ avente distribuzione F_U : allora U è uniformemente distribuita in $[0, 1]$

Infatti si ha:

$$\begin{aligned} F_U(u) &= \Pr(U \leq u) = \Pr(F_X(X) \leq u) \\ &= \Pr(X \leq F_X^{-1}(u)) = F_X(F_X^{-1}(u)) \\ &= u \end{aligned}$$

Random Variate Generators

Fin qui abbiamo visto come generare valori uniformemente distribuiti nell'intervallo $[0, 1]$. **Come generare valori distribuiti in altro modo?**

Definizione

Una V.A. X ha una **distribuzione** F_X se $F_X(x) = \Pr(X \leq x)$ per ogni $x \in \mathbb{R}$

Teorema

Sia X una V.A. con distribuzione **continua monotona crescente** F_X . Consideriamo la V.A. $U = F_X(X)$ avente distribuzione F_U : allora U è uniformemente distribuita in $[0, 1]$

Infatti si ha:

$$\begin{aligned} F_U(u) &= \Pr(U \leq u) = \Pr(F_X(X) \leq u) \\ &= \Pr(X \leq F_X^{-1}(u)) = F_X(F_X^{-1}(u)) \\ &= u \end{aligned}$$

Random Variate Generators

Data una V.A. U con distribuzione uniforme in $[0, 1]$, la V.A. $X = F_X^{-1}(U)$ ha distribuzione F_X .

Esempio: Una V.A. X è **esponenzialmente distribuita** con parametro λ se la sua distribuzione F_X è

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

Posto $1 - e^{-\lambda x} = u$ si ha $x = -\frac{1}{\lambda} \ln(1 - u)$. Quindi una sequenza di valori esponenzialmente distribuiti x_1, x_2, \dots può essere generata a partire da una sequenza di valori uniformemente distribuiti u_1, u_2, \dots applicando a ciascun u_i la trasformazione $x_i = -\frac{1}{\lambda} \ln(1 - u_i)$.

Random Variate Generators

Data una V.A. U con distribuzione uniforme in $[0, 1]$, la V.A. $X = F_X^{-1}(U)$ ha distribuzione F_X .

Esempio: Una V.A. X è **esponenzialmente distribuita** con parametro λ se la sua distribuzione F_X è

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

Posto $1 - e^{-\lambda x} = u$ si ha $x = -\frac{1}{\lambda} \ln(1 - u)$. Quindi una sequenza di valori esponenzialmente distribuiti x_1, x_2, \dots può essere generata a partire da una sequenza di valori uniformemente distribuiti u_1, u_2, \dots applicando a ciascun u_i la trasformazione $x_i = -\frac{1}{\lambda} \ln(1 - u_i)$.

Random Variate Generators

Data una V.A. U con distribuzione uniforme in $[0, 1]$, la V.A. $X = F_X^{-1}(U)$ ha distribuzione F_X .

Esempio: Una V.A. X è **esponenzialmente distribuita** con parametro λ se la sua distribuzione F_X è

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

Posto $1 - e^{-\lambda x} = u$ si ha $x = -\frac{1}{\lambda} \ln(1 - u)$. Quindi una sequenza di valori esponenzialmente distribuiti x_1, x_2, \dots può essere generata a partire da una sequenza di valori uniformemente distribuiti u_1, u_2, \dots applicando a ciascun u_i la trasformazione $x_i = -\frac{1}{\lambda} \ln(1 - u_i)$.

Research Diagram / Research Reality

What your research supposedly looks like:

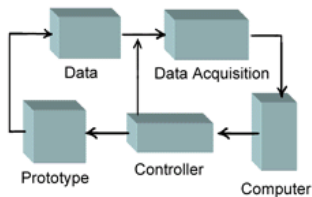


Figure 1. Experimental Diagram

What your research *actually* looks like:

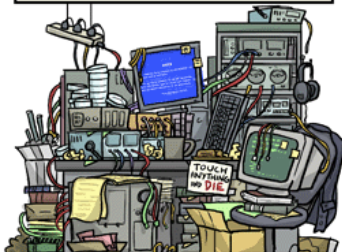


Figure 2. Experimental Mess

<http://www.phdcomics.com/comics/archive.php?comicid=961>

JORGE CHAM © 2008

WWW.PHDCOMICS.COM

- 1 Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output**
- 5 Conclusioni
- 6 Bibliografia

Coda M/M/1



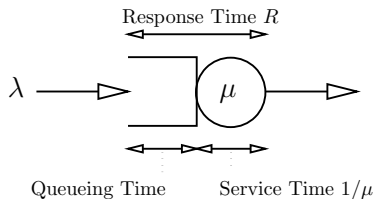
Supponiamo di voler simulare questo sistema, con i parametri seguenti:

- **Tasso di arrivo** λ richieste/s
- **Tasso di servizio** μ richieste completate/s
- Tempo di interarrivo e tempo di servizio seguono una distribuzione **esponenziale negativa** di parametri λ e μ rispettivamente:

$$F_{\lambda}(x) = \begin{cases} 1 - e^{-\lambda x} & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases}$$

(il valore medio è $1/\lambda$).

Coda $M/M/1$



In seguito, considereremo una coda $M/M/1$ con tasso di arrivo $\lambda = 0.09$ richieste/s e tasso di servizio $\mu = 0.1$ richieste/s.

Obiettivo: valutare tramite simulazione il tempo medio di risposta R

Intervallo di confidenza sulla media

Vale solo per osservazioni *non correlate*

Consideriamo n osservazioni **non correlate** X_1, X_2, \dots, X_n di una V.A. X . Posto:

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i \quad (2)$$

$$\sigma^2(X) = \frac{1}{n-1} \sum_{i=1}^n (\bar{X} - X_i)^2 \quad (3)$$

Si ha che un intervallo di confidenza a livello $100(1 - \alpha)\%$ per la media $E[X]$ è:

$$\left[\bar{X} \mp z_{1-\alpha/2} \sqrt{\frac{\sigma^2(X)}{n}} \right] \quad (4)$$

dove $z_{1-\alpha/2}$ è il $(1 - \alpha/2)$ -quantile della distribuzione normale standard.

Intervallo di confidenza sulla media

Definizione

Dire che $I = [a, b]$ è un *intervallo di confidenza* di livello $100(1 - \alpha)\%$ per la media $E[X]$ di una V.A. X , significa che

$$\Pr(a \leq E[X] \leq b) = 1 - \alpha$$

Output di una simulazione

Idea:

- Generiamo n arrivi di job;
- Otteniamo quindi una sequenza di tempi di risposta X_1, X_2, \dots, X_n (**osservazioni**)

Quindi:

$$\text{Tempo medio di Risposta} \approx \frac{1}{n} \sum_{i=1}^n X_i \quad \text{????}$$

Output di una simulazione

Idea:

- Generiamo n arrivi di job;
- Otteniamo quindi una sequenza di tempi di risposta X_1, X_2, \dots, X_n (**osservazioni**)

Quindi:

$$\text{Tempo medio di Risposta} \approx \frac{1}{n} \sum_{i=1}^n X_i \quad \text{????}$$

Output di una simulazione

Idea:

- Generiamo n arrivi di job;
- Otteniamo quindi una sequenza di tempi di risposta X_1, X_2, \dots, X_n (**osservazioni**)

Quindi:

$$\text{Tempo medio di Risposta} \approx \frac{1}{n} \sum_{i=1}^n X_i \quad \text{????}$$



Dove sta il problema?

Perché non si può semplicemente calcolare il valore medio come media delle osservazioni?

- 1 Le osservazioni X_1, X_2, \dots, X_n sono in generale autocorrelate, e non indipendenti come sarebbe necessario per calcolare il valore medio come media aritmetica
- 2 Le osservazioni X_1, X_2, \dots, X_n sono affette dal problema del **transiente iniziale** (*initialization bias*)

Per queste due ragioni purtroppo risulta in generale che

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

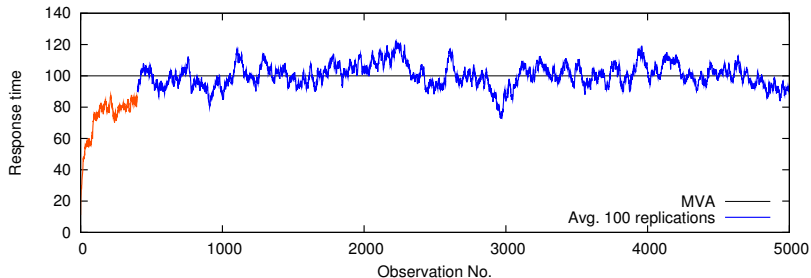
- Normalmente le condizioni iniziali della simulazione **non** corrispondono con le condizioni del sistema “a regime”
 - Es: all’inizio della simulazione di una rete di code aperta tutti i centri di servizio sono vuoti. Questa condizione raramente si verifica a regime.
- I primi valori della sequenza di osservazioni X_1, X_2, \dots, X_{n_0} per un certo $n_0 \ll n$ vanno scartati:

$$\underbrace{X_1, \dots, X_{n_0}}_{\text{Transiente}}, X_{n_0+1}, \dots, X_n$$

Transiente

Esempio

Tempo di risposta, media di $m = 100$ repliche indipendenti di $n = 2000$ osservazioni ciascuna.



Purtroppo **non esiste una procedura universalmente valida** per determinare la lunghezza n_0 del transiente [2, 1].

Come limitare l'effetto del transiente?

1 Effettuare simulazioni molto lunghe

L'effetto del transiente diventa trascurabile al crescere della lunghezza complessiva del run di simulazione

2 Troncamento

Rimuovere dalla sequenza di osservazioni la parte di transiente iniziale, e calcolare le misure di interesse su ciò che rimane

3 Inizializzazione appropriata

Ad esempio, fare in modo che all'avvio della simulazione ci sia già qualche utente in coda nel centro di servizio

Repliche Indipendenti

Effettuiamo m run **indipendenti** di n osservazioni ciascuno. Cioè si ripete m volte la simulazione usando sequenze **non sovrapposte** di numeri casuali.

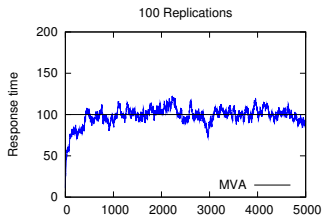
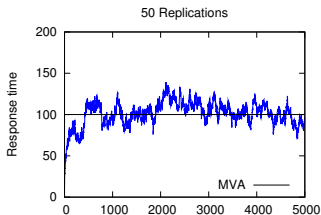
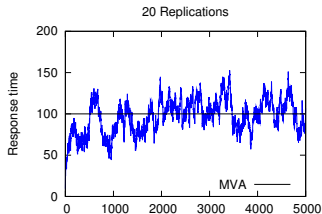
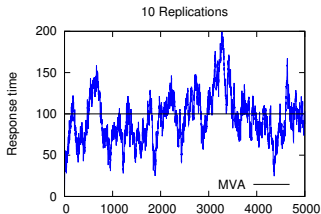
	Osservazioni				Medie
Run 1	X_{11}	X_{12}	\dots	X_{1n}	$\bar{X}_{1,*}$
Run 2	X_{21}	X_{22}	\dots	X_{2n}	$\bar{X}_{2,*}$
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
Run m	X_{m1}	X_{m2}	\dots	X_{mn}	$\bar{X}_{m,*}$
Medie	$\bar{X}_{*,1}$	$\bar{X}_{*,2}$	\dots	$\bar{X}_{*,n}$	

$$\bar{X}_{i,*} = \frac{1}{n} \sum_{j=1}^n X_{ij} \quad i = 1, 2, \dots, m \quad (5)$$

$$\bar{X}_{*,j} = \frac{1}{m} \sum_{i=1}^m X_{ij} \quad j = 1, 2, \dots, n \quad (6)$$

Repliche Indipendenti

Tempo di risposta di un server $M/M/1$ con $\lambda = 0.09$ e $\mu = 0.1$.
Notare come all'aumentare di m diminuisce la varianza della media delle osservazioni.



La regola **Marginal Confidence Rule** (MCR) si basa sull'idea che eliminando il transiente iniziale, la varianza della sequenza di osservazioni residue diminuisce (e quindi si restringe anche l'intervallo di confidenza sulla media).

Diversamente da [5, 6] noi applicheremo MCR sulla sequenza $\{\bar{X}_{*,1}, \bar{X}_{*,2}, \dots, \bar{X}_{*,n}\}$, ossia sulle osservazioni mediate su m run indipendenti.

- 1 Sia \mathcal{S}_i la sequenza $\{\bar{X}_{*,i}, \bar{X}_{*,i+1}, \dots, \bar{X}_{*,n}\}$, $i = 1, 2, \dots, n-1$
- 2 Sia $\sigma^2(\mathcal{S}_i)$ la varianza della sequenza \mathcal{S}_i :

$$\sigma^2(\mathcal{S}_i) = \frac{1}{n-i} \sum_{j=i}^n (\bar{X}_{*,j} - \bar{S}_i)^2 \quad i = 1, 2, \dots, n-1$$

ove $\bar{S}_i = \frac{1}{n-i+1} \sum_{k=i}^n \bar{X}_{*,k}$

- 3 L'intervallo di confidenza (4) per la media di \mathcal{S}_i è proporzionale a

$$d_i = \sqrt{\frac{\sigma^2(\mathcal{S}_i)}{(n-i+1)}}$$

- 4 Il valore di i che rende minimo d_i darà la stima della lunghezza n_0 del transiente

Implementazione Octave di MCR

MCR.m

```
## Detect the length of the transient period given observations d. d is
## a 1*n matrix of n observations. The length n_0 of the transient is
## returned as result
function n0 = MCR( d )
    n = size(d,2);
    v = zeros(1,n-1);
    for i=1:size(v,2)
        v(i) = sqrt( var( d(i:n) ) / (n-i+1) );
    endfor
    [x,n0] = min( v );
endfunction
```

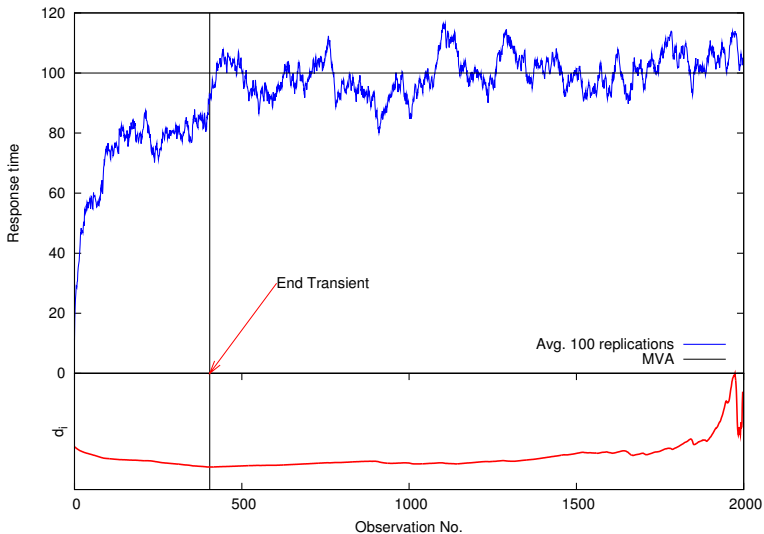
Esempio di utilizzo

```
# lambda=0.09, mu=0.1, n = 5000, m = 10
X = generate_response_time( 0.09, 0.1, 5000, 10 );
# Compute transient length on the averages
tr = MCR( mean(X) )

ans = 378
```

Esempio di applicazione di MCR

Coda $M/M/1$, $\lambda = 0.09$, $\mu = 0.1$, $m = 100$ repliche



Calcolo della media

m repliche indipendenti di n osservazioni ciascuna

Date m sequenze di osservazioni **indipendenti**, possiamo considerare le medie $\bar{X}_{j,*}$ delle sequenze come se fossero indipendenti, anche se le osservazioni delle singole sequenze non lo sono.

Si può quindi calcolare un intervallo di confidenza sulla media come segue:

- 1 Individuare n_0 usando l'algoritmo appena visto
- 2 Definire

$$Y_j = \frac{1}{n - n_0} \sum_{i=n_0+1}^n X_{ij} \quad j = 1, 2, \dots, m$$

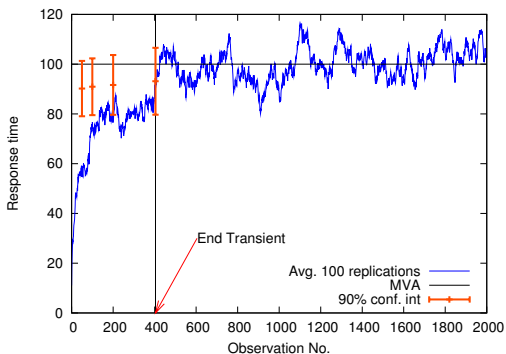
- 3 Calcolare la media e l'intervallo di confidenza della sequenza Y_1, Y_2, \dots, Y_m considerandole osservazioni indipendenti applicando (4)

Esempio

Effetto del transiente sulla media

- Centro di servizio con $\lambda = 0.09$, $\mu = 0.1$
- $m = 100$ repliche di $n = 2000$ osservazioni ciascuna

Calcoliamo l'intervallo di confidenza sulle medie delle repliche usando l'algoritmo di pag. 78 al variare del numero n_0 di osservazioni iniziali scartate.

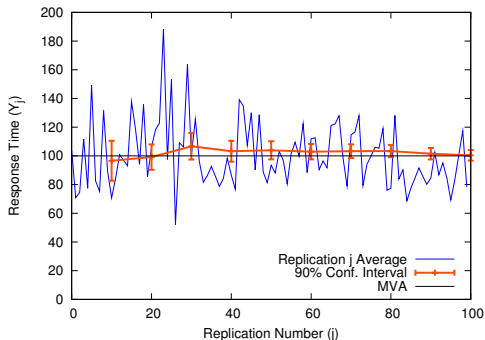


Esempio

Effetto del numero m di repliche sulla media

- Centro di servizio con $\lambda = 0.09$, $\mu = 0.1$
- m repliche con $n = 5000$ osservazioni ciascuna

Calcoliamo l'intervallo di confidenza sulla media usando l'algoritmo di pag. 78 al variare del numero di repliche m .



- 1 Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output
- 5 Conclusioni**
- 6 Bibliografia

Conclusioni

- La simulazione discreta è un settore molto interessante sotto diversi punti di vista (programmazione, algoritmi, analisi numerica, ...)
- Effettuare simulazioni **ben fatte** richiede molta attenzione a tanti piccoli dettagli
- È un settore ancora in evoluzione
 - Parallel and Distributed Simulation
 - Real-Time Simulation
 - Massive Online Gaming

The end!!

Domande?

ADDRESSING REVIEWER COMMENTS

BAD REVIEWS ON YOUR PAPER? FOLLOW THESE GUIDELINES AND YOU MAY YET GET IT PAST THE EDITOR:

Reviewer comment:

"The method/device/paradigm the authors propose is clearly wrong."

How NOT to respond:

✗ "Yes, we know. We thought we could still get a paper out of it. Sorry."

Correct response:

✓ "The reviewer raises an interesting concern. However, as the focus of this work is exploratory and not performance-based, validation was not found to be of critical importance to the contribution of the paper."

Reviewer comment:

"The authors fail to reference the work of Smith et al., who solved the same problem 20 years ago."

How NOT to respond:

✗ "Huh. We didn't think anybody had read that. Actually, their solution is better than ours."

Correct response:

✓ "The reviewer raises an interesting concern. However, our work is based on completely different first principles (we use different variable names), and has a much more attractive graphical user interface."

Reviewer comment:

"This paper is poorly written and scientifically unsound. I do not recommend it for publication."

How NOT to respond:

✗ "You #@%* reviewer! I know who you are! I'm gonna get you when it's my turn to review!"

Correct response:

✓ "The reviewer raises an interesting concern. However, we feel the reviewer did not fully comprehend the scope of the work, and misjudged the results based on incorrect assumptions."

www.phdcomics.com

JORGE CHAM © 2005

<http://www.phdcomics.com/comics/archive.php?comicid=581>

- 1 Introduzione
- 2 Simulazione Discreta
- 3 Generatori Pseudocasuali
- 4 Analisi dell'Output
- 5 Conclusioni
- 6 Bibliografia**

Riferimenti

- [1] Christos Alexopoulos and Andrew F. Seila.
Advanced methods for simulation output analysis.
In Proceedings of the 30th conference on Winter simulation, pages 113–120. IEEE Computer Society Press, 1998.
- [2] Jerry Banks, editor.
Handbook of Simulation.
Wiley–Interscience, 1998.
- [3] Raj Jain.
The Art of Computer Systems Performance Analysis–Techniques for Experimental Design, Measurement, Simulation, and Modeling.
Wiley-Interscience, 1991.
- [4] M. Matsumoto and T. Nishimura.
Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator.
ACM Trans. on Modeling and Computer Simulation, 8(1):3–30, February 2008.
- [5] Jr. White, K.P.
A simple rule for mitigating initialization bias in simulation output: comparative results.
Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on, 1:206–211 vol.1, Oct 1995.
- [6] Jr. White, K.P. and M.A. Minnox.
Minimizing initialization bias in simulation output using a simple heuristic.
Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology', 1994 IEEE International Conference on, 1:215–220 vol.1, Oct 1994.