Studente: Michele Vaccari

corso: Reti di telecomunicazioni e internet | Docente: Gianluca mazzini



UNIVERSITà DEGLI STUDI DI FERRARA

**SIMULAZIONE SISTEMA M/M/1 E M/M/1/Y**

Sommario

[1 Introduzione 3](#_Toc446685466)

[2 Diagramma di flusso del simulatore 4](#_Toc446685467)

[3 Errori tra teoria e simulazioni 5](#_Toc446685468)

[4 Sistema M/M/1 6](#_Toc446685469)

[4.1 Richiami teorici 6](#_Toc446685470)

[4.2 Simulazione del sistema M/M/1 6](#_Toc446685471)

[4.2.1 Tempo medio trascorso nel sistema 6](#_Toc446685472)

[4.2.2 Tempo medio trascorso in coda 10](#_Toc446685473)

[4.2.3 Numero medio di utenti nel sistema 14](#_Toc446685474)

[4.2.4 Numero medio di utenti in coda 17](#_Toc446685475)

[4.3 Errori tra valori teorici e simulati nel sistema M/M/1 20](#_Toc446685476)

[4.3.1 Tempo medio trascorso nel sistema 20](#_Toc446685477)

[4.3.2 Tempo medio trascorso in coda 23](#_Toc446685478)

[4.3.3 Numero medio di utenti nel sistema 26](#_Toc446685479)

[4.3.4 Numero medio di utenti in coda 29](#_Toc446685480)

[5 Sistema M/M/1/Y 32](#_Toc446685481)

[5.1 Richiami teorici 32](#_Toc446685482)

[5.2 Simulazione del sistema M/M/1/Y 32](#_Toc446685483)

[5.2.1 Tempo medio trascorso nel sistema 33](#_Toc446685484)

[5.2.2 Tempo medio trascorso in coda 36](#_Toc446685485)

[5.2.3 Numero medio di utenti nel sistema 39](#_Toc446685486)

[5.2.4 Numero medio di utenti in coda 42](#_Toc446685487)

[5.3 Errori tra valori teorici e simulati nel sistema M/M/1/Y 45](#_Toc446685488)

[5.3.1 Tempo medio trascorso nel sistema 45](#_Toc446685489)

[5.3.2 Tempo medio trascorso in coda 48](#_Toc446685490)

[5.3.3 Numero medio di utenti nel sistema 51](#_Toc446685491)

[5.3.4 Numero medio di utenti in coda 54](#_Toc446685492)

[6 Conclusioni 57](#_Toc446685493)

[Appendice A – Variabile aleatoria distribuita secondo Poisson 58](#_Toc446685494)

[Appendice B – Breve guida all’uso del simulatore 59](#_Toc446685495)

[Appendice C - Codice sorgente del simulatore 61](#_Toc446685496)

# Introduzione

La commutazione di pacchetto è una metodologia di consegna dell'informazione che presuppone una visione del cammino fra trasmettitore e ricevitore formata da una rete di nodi. Ogni nodo, a seconda delle situazioni, può comportarsi da trasmettitore, da ricevitore o da ripetitore. Per svolgere quest'ultima funzione il nodo deve essere dotato di un sistema di code, atto a memorizzare i pacchetti ad esso inviati, prima di instradarli verso le opportune destinazioni (metodologia *store and forward*). L'analisi di sistemi a coda è basata sulla creazione di un modello teorico che, sulla base di opportune ipotesi esemplificative, permette di determinare analiticamente le grandezze che lo descrivono. Un simulatore del sistema, realizzato al calcolatore, diviene uno strumento fondamentale per verificare quanto le ipotesi esemplificative abbiano portato a soluzioni teoriche accettabili, intese come confrontabili con la realtà riprodotta dal simulatore. La generazione di una variabile casuale distribuita secondo Poisson è stata calcolata adottando la tecnica di trasformazione inversa, riportata in appendice A.

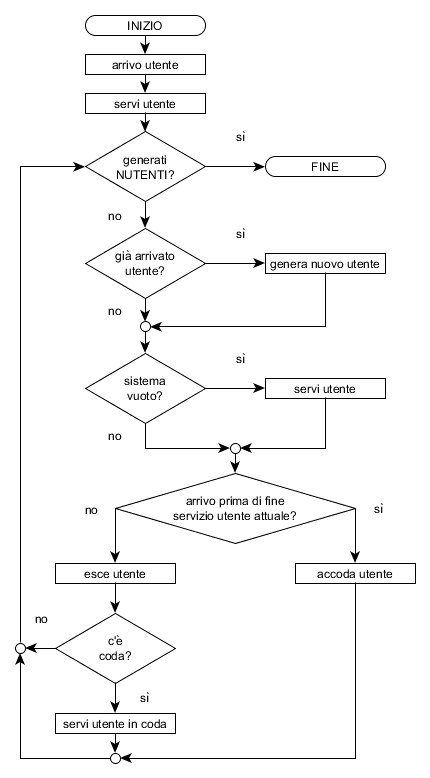
Il simulatore è stato scritto in Visual C#, utilizzando come IDE Visual Studio 2017. La scelta di creare un’applicazione di tipo visuale e non di tipo console è dovuta fondamentalmente alla possibilità di impostare, anche per chi non ha particolari conoscenze di programmazione, i vari parametri della simulazione (numero utenti, massima dimensione coda e passo di incremento di ), oltre che dare la possibilità di scegliere dove verranno salvati i file generati dalla simulazione. Il codice sorgente del simulatore viene riportato nell’Appendice C. Le simulazioni sono state effettuate su un pc con processore Intel i5 a circa 3.0 GHz.

Nei grafici presenti in questa relazione, verranno riportati i valori ottenuti dalla simulazione e i valori teorici, per valori di 103, 104, 105, 106, 5\*106 e 107 utenti. I risultati risultano coincidere con i valori teorici, almeno per valori di di 0.97 ÷ 1.00 o prossimi allo 0, dove l’errore che si commette cresce fino ad arrivare al 15% ÷ 20%.

Il sistema che abbiamo deciso di simulare, è un sistema M/M/1, cioè un sistema dove gli arrivi e i tempi di servizio sono di tipo esponenziale, è presente un solo servitore e la coda non ha una dimensione massima. È possibile anche simulare un sistema di tipo M/M/1/Y (identico al precedente, ad eccezione del fatto che la coda ha dimensione massima Y), impostando in maniera corretta i parametri della simulazione, prima di avviarla. Nell’Appendice B è riportata una breve guida all’uso del simulatore, che spiega come impostare i parametri per simulare i due tipi di sistemi.

# Diagramma di flusso del simulatore

Viene riportato il diagramma di flusso del metodo che effettua la simulazione del sistema a coda.



# Errori tra teoria e simulazioni

Durante il corso abbiamo sviluppato un modello matematico di carattere generale che, attraverso alcune ipotesi esemplificative, è stato possibile risolvere analiticamente. Il simulatore conferma i risultati teorici con una certa approssimazione, strettamente legata al numero di utenti impiegati per determinare i valori medi delle grandezze di interesse. Le discrepanze fra gli andamenti simulati e quelli teorici diventano sempre meno evidenti al crescere del numero di utenti come mostrano i grafici presenti in questa relazione. Anche una valutazione degli errori relativi percentuali, calcolati come:

conferma l'aumento della precisione raggiunta al crescere del numero di utenti.

Per concludere questa breve ma dovuta discussione sulla validità della simulazione, facciamo alcune osservazioni. Gli errori relativi percentuali iniziali risultano molto elevati perché le grandezze teoriche assumono inizialmente valori prossimi allo zero, per cui assistiamo ad un fenomeno di amplificazione delle fluttuazioni statistiche sui dati iniziali. Successivamente gli errori decrescono. Al crescere del numero di utenti aumenta la precisione del simulatore ma cresce altresì il tempo reale di simulazione al calcolatore, come mostra la seguente tabella:

|  |  |
| --- | --- |
| Numero utenti | Durata simulazione |
| 103 | 0.06 s |
| 104 | 0.6 s |
| 105 | 6.3 s |
| 106 | 1 m |
| 5 \* 106 | 5 m 19 s |
| 107 | 10 m |

# Sistema M/M/1

## Richiami teorici

Un sistema a coda M/M/1 ha una distribuzione dei tempi interarrivo e di servizio di tipo esponenziale, un solo servitore e una dimensione della coda illimitata. In pratica questo equivale a dire che:

Specializzando le soluzioni generali dei processi di nascita e morte in equilibrio, otteniamo:

Il numero medio di utenti è:

Il tempo medio speso nel sistema è:

Infine, il tempo medio speso in coda è:

Il numero medio di utenti in coda è:

## Simulazione del sistema M/M/1

Per simulare il comportamento di un sistema M/M/1, basta impostare la dimensione della coda del simulatore pari o superiore al numero di utenti con il quale la simulazione verrà effettuata.

### Tempo medio trascorso nel sistema

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[T] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

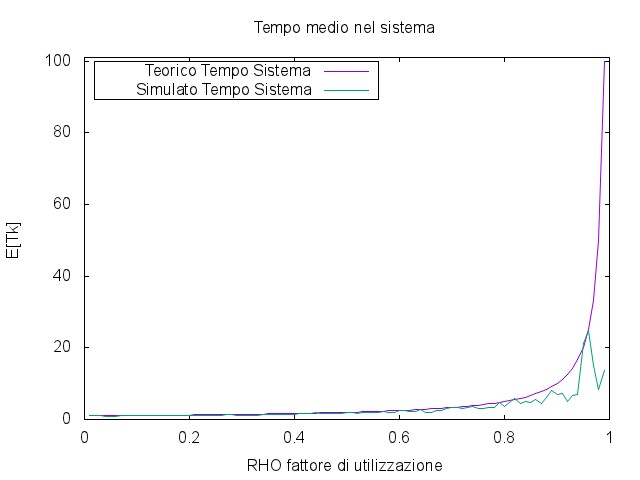


Figura 4.1 Tempo medio trascorso nel sistema con 1000 utenti

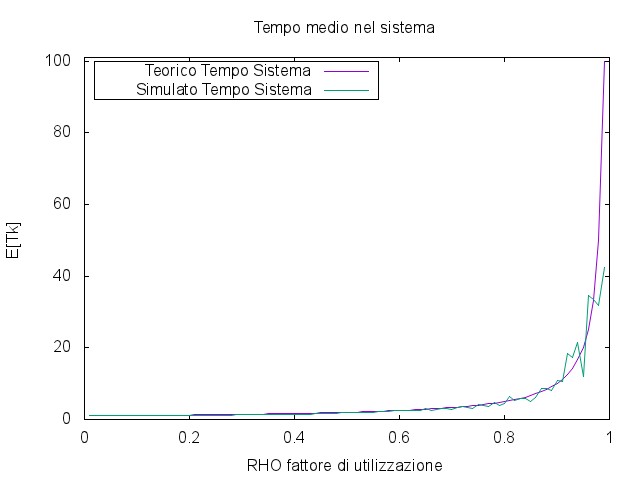


Figura 4.2 Tempo medio trascorso nel sistema con 10000 utenti

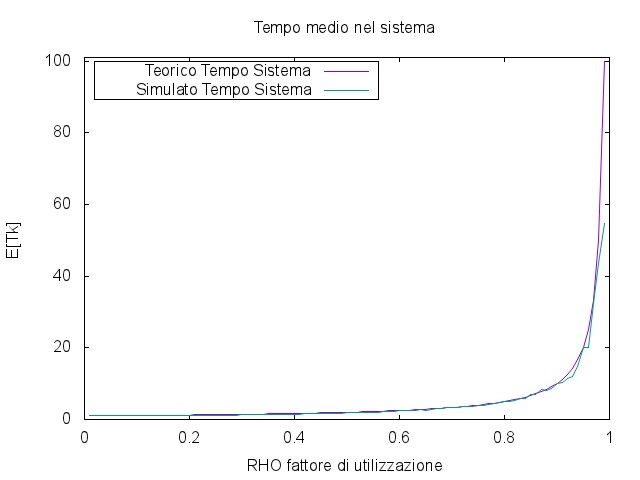


Figura 4.3 Tempo medio trascorso nel sistema con 100000 utenti

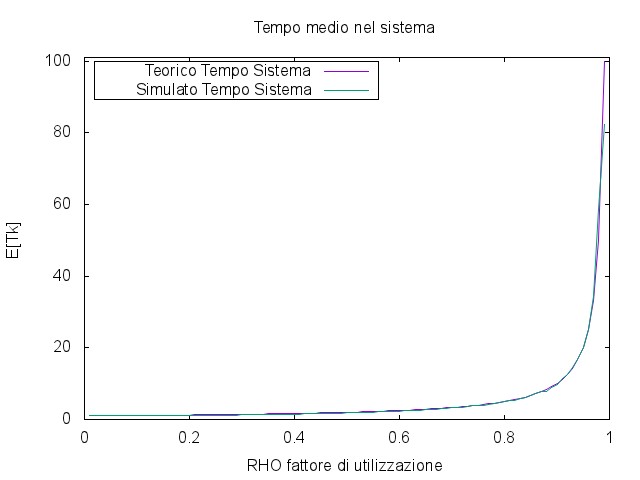


Figura 4.4 Tempo medio trascorso nel sistema con 1000000 utenti

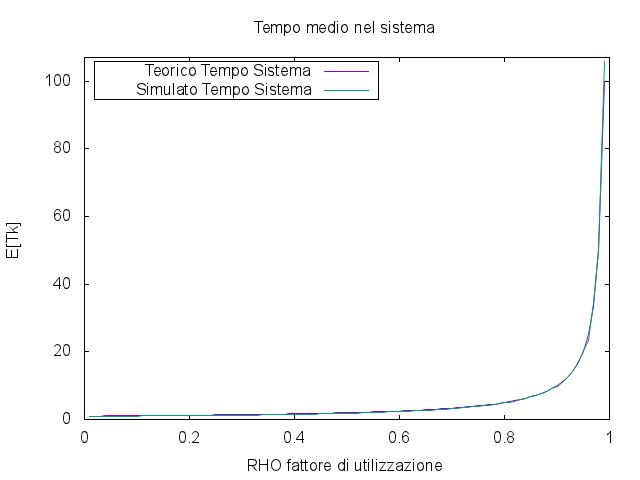


Figura 4.5 Tempo medio trascorso nel sistema con 5000000 utenti

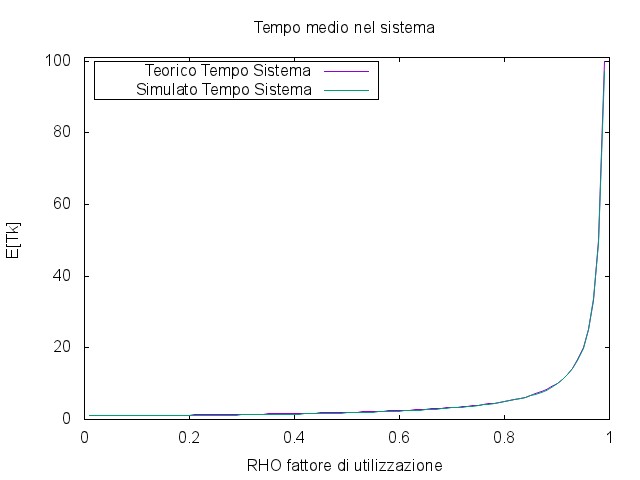


Figura 4.6 Tempo medio trascorso nel sistema con 10000000 utenti

### Tempo medio trascorso in coda

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[Tq] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

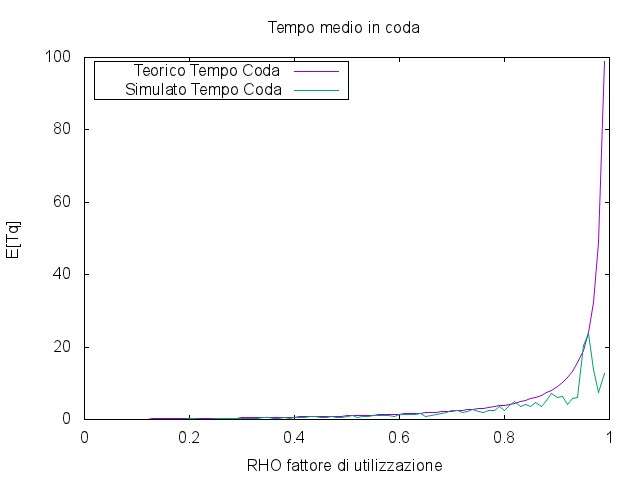


Figura 4.7 Tempo medio trascorso in coda con 1000 utenti

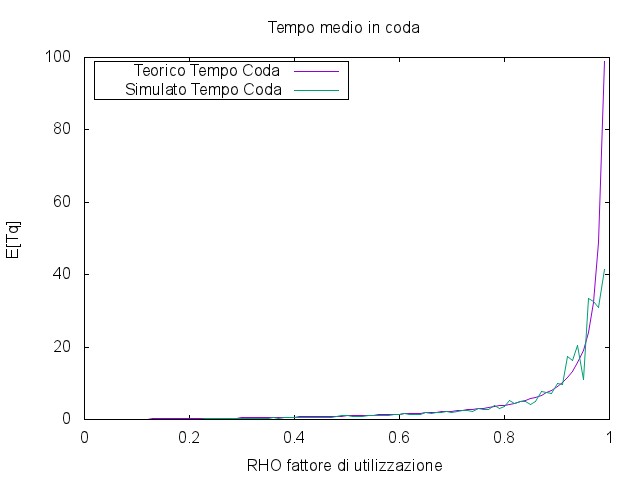


Figura 4.8 Tempo medio trascorso nel sistema con 10000 utenti

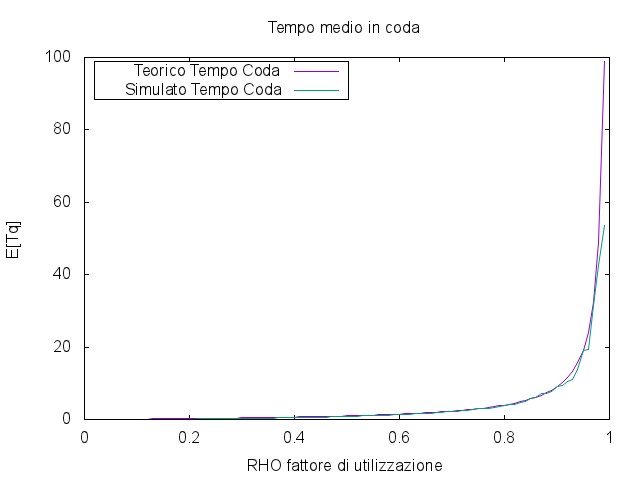


Figura 4.9 Tempo medio trascorso nel sistema con 100000 utenti

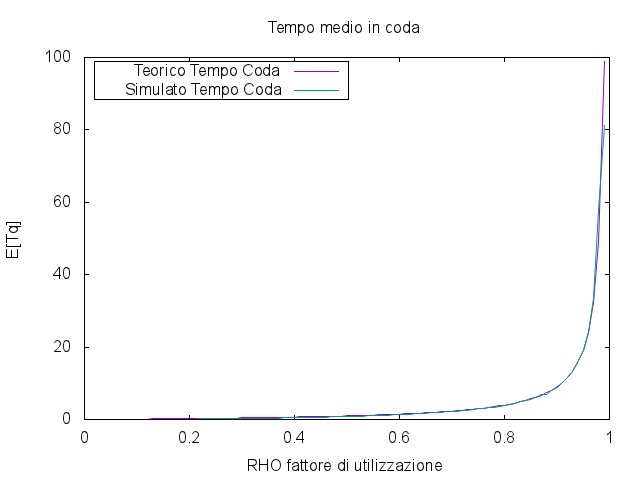


Figura 4.10 Tempo medio trascorso nel sistema con 1000000 utenti

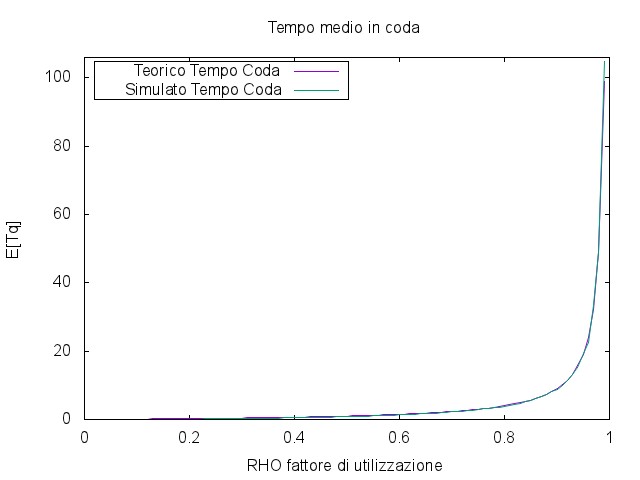


Figura 4.11 Tempo medio trascorso nel sistema con 5000000 utenti

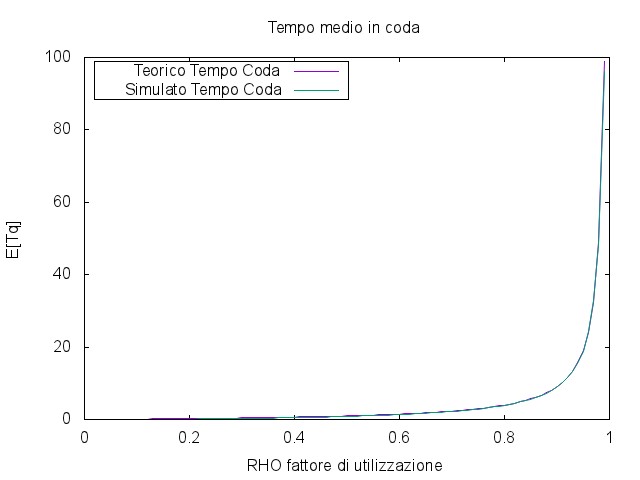


Figura 4.12 Tempo medio trascorso nel sistema con 10000000 utenti

### Numero medio di utenti nel sistema

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[k] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

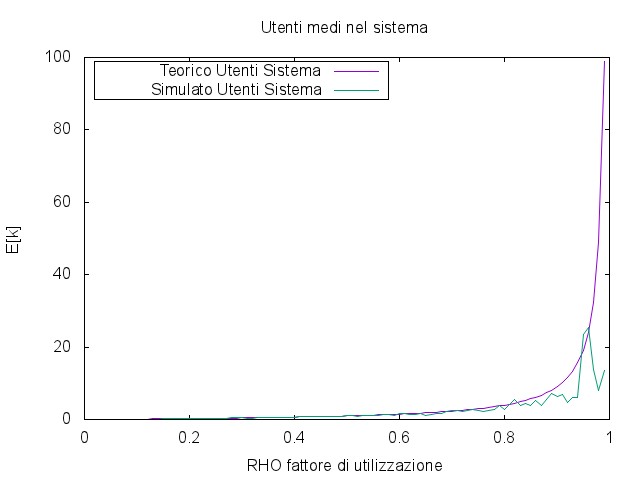


Figura 4.13 Numero medio di utenti nel sistema con 1000 utenti

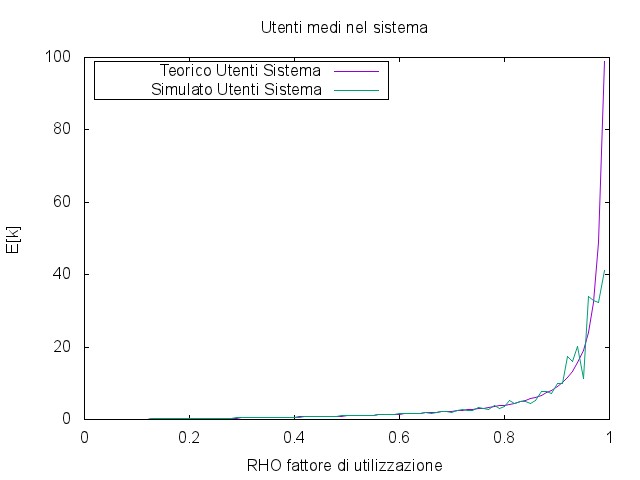


Figura 4.14 Numero medio di utenti nel sistema con 10000 utenti

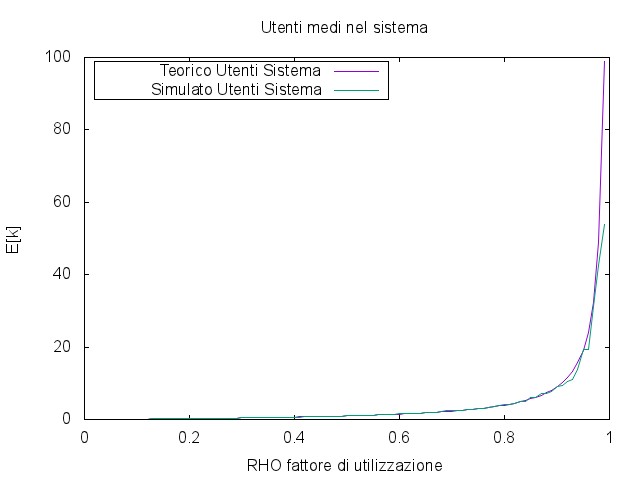


Figura 4.15 Numero medio di utenti nel sistema con 100000 utenti

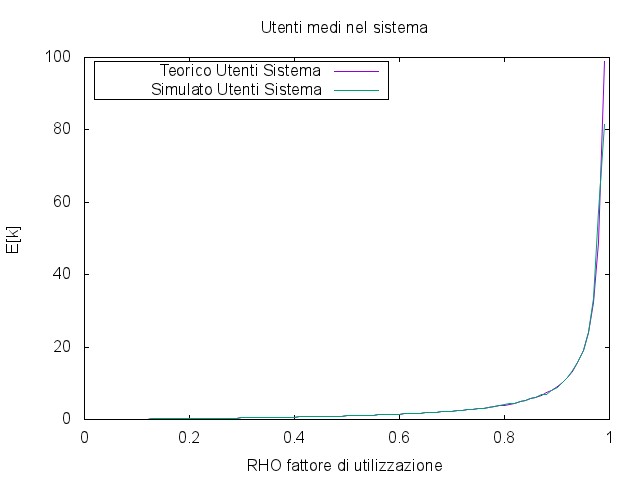


Figura 4.16 Numero medio di utenti nel sistema con 1000000 utenti

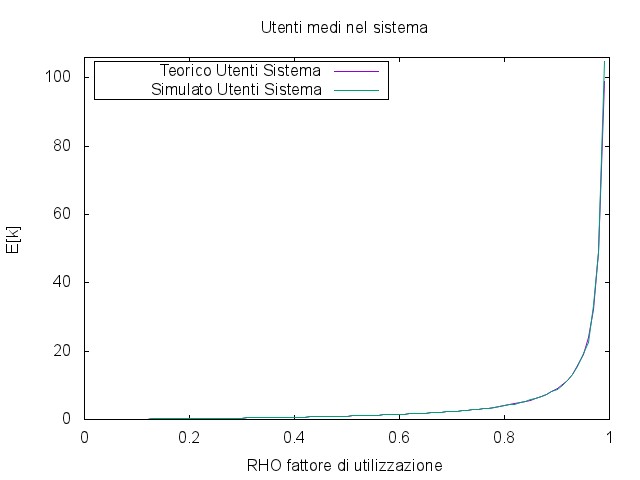


Figura 4.17 Numero medio di utenti nel sistema con 5000000 utenti

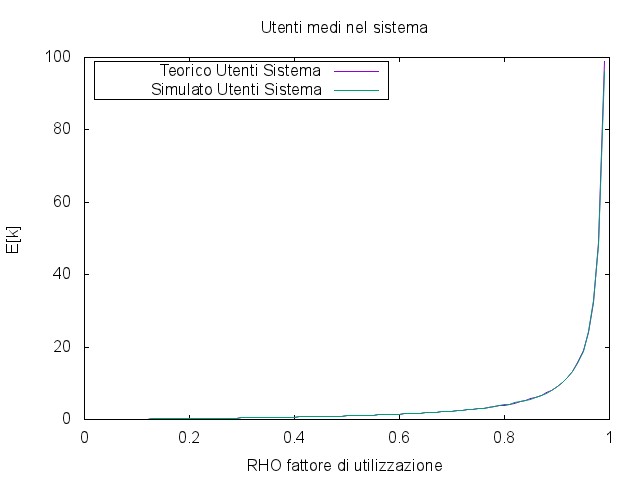


Figura 4.18 Numero medio di utenti nel sistema con 10000000 utenti

### Numero medio di utenti in coda

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[q] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

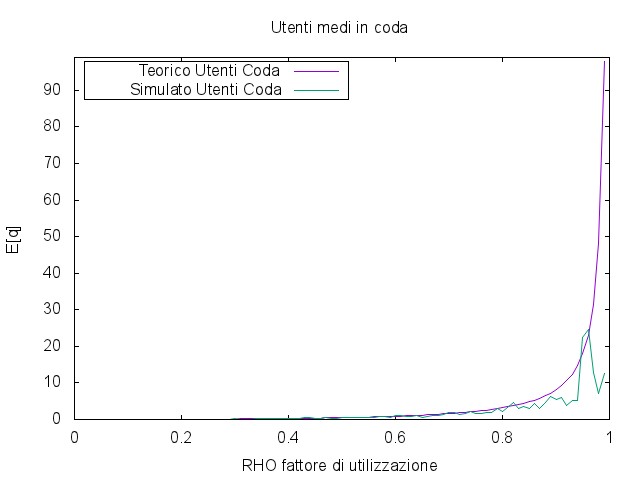


Figura 4.19 Numero medio di utenti in coda con 1000 utenti

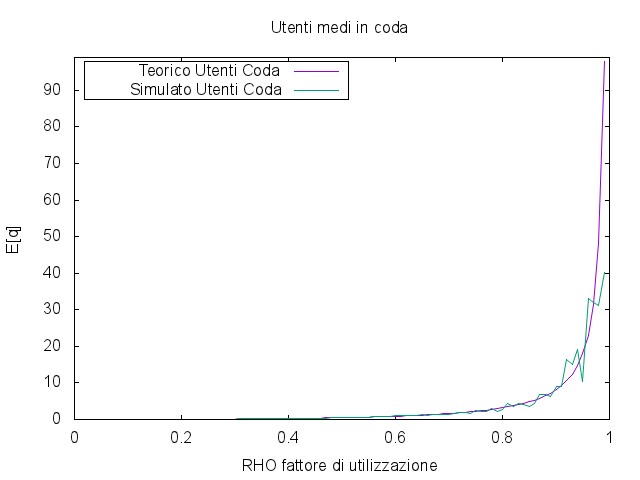


Figura 4.20 Numero medio di utenti in coda con 10000 utenti

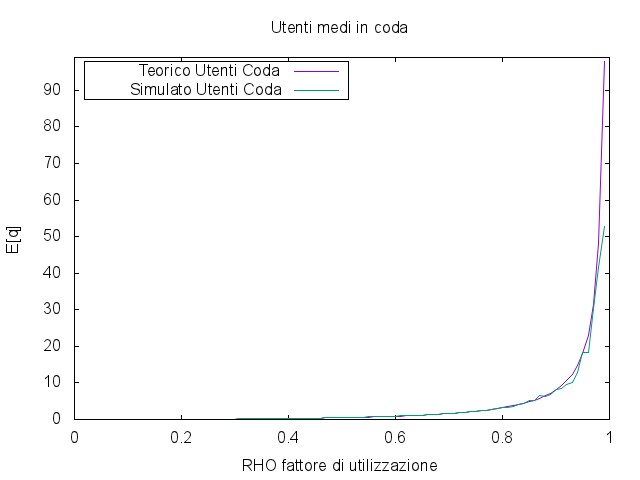


Figura 4.21 Numero medio di utenti in coda con 100000 utenti

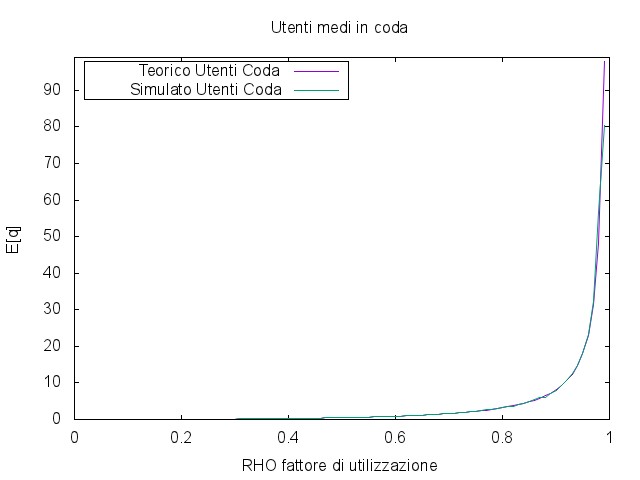


Figura 4.22 Numero medio di utenti in coda con 1000000 utenti

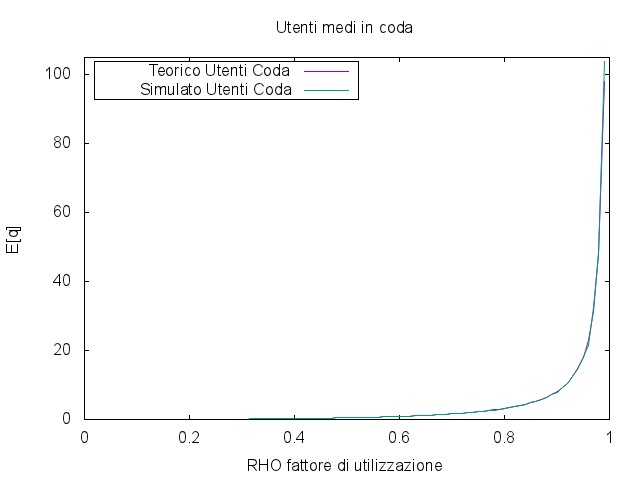


Figura 4.23 Numero medio di utenti in coda con 5000000 utenti

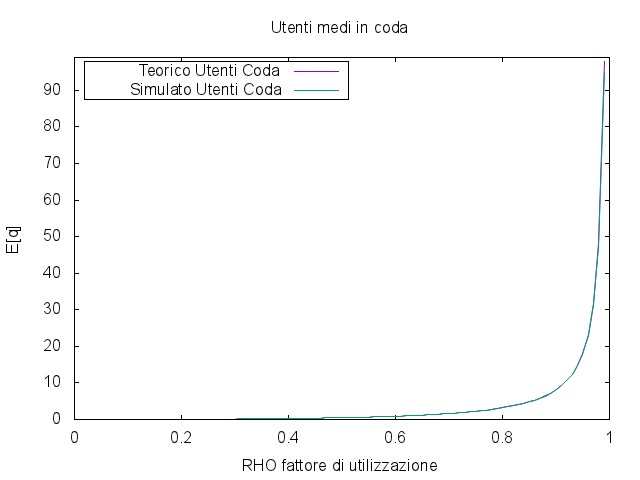


Figura 4.24 Numero medio di utenti in coda con 10000000 utenti

## Errori tra valori teorici e simulati nel sistema M/M/1

### Tempo medio trascorso nel sistema

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[T], al variare del numero di utenti con il quale la simulazione viene effettuata.

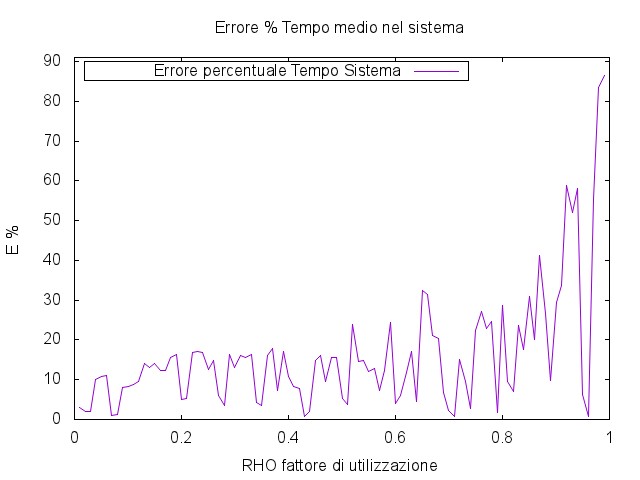


Figura 4.25 Errore percentuale E[T] con 1000 utenti

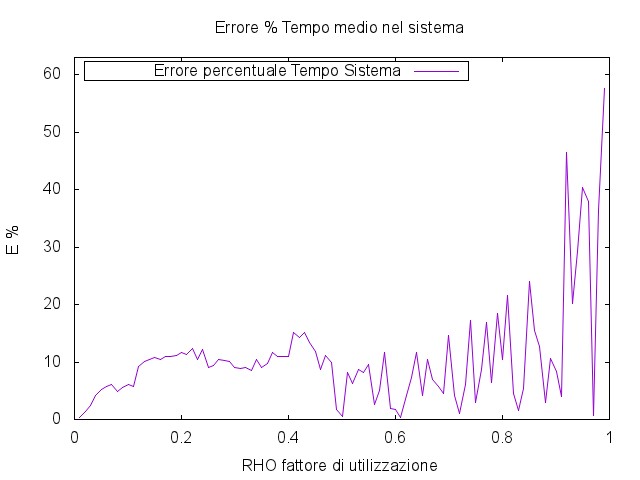


Figura 4.26 Errore percentuale E[T] con 10000 utenti

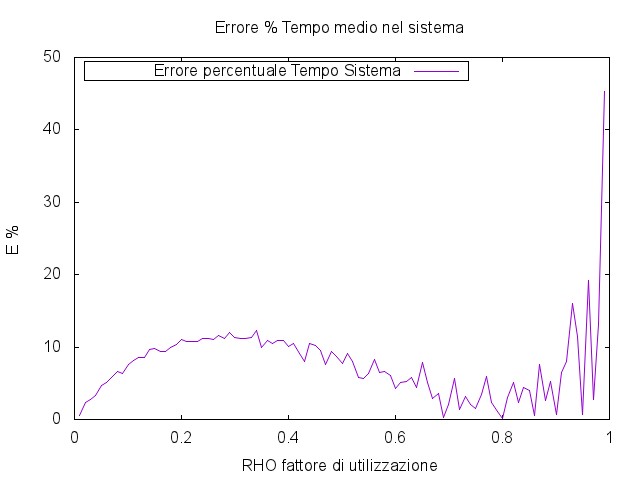


Figura 4.27 Errore percentuale E[T] con 100000 utenti

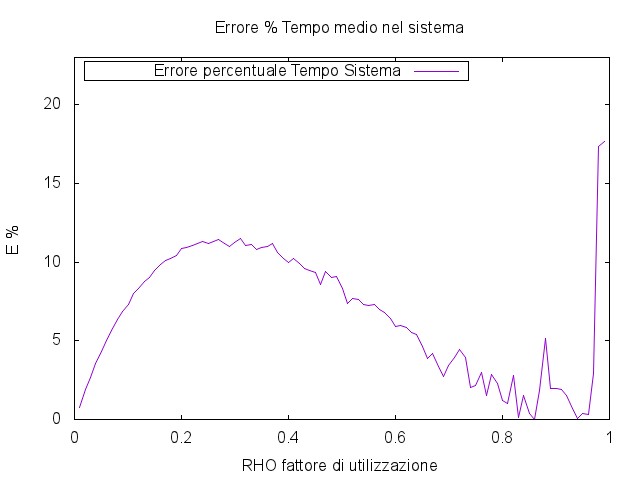


Figura 4.28 Errore percentuale E[T] con 1000000 utenti

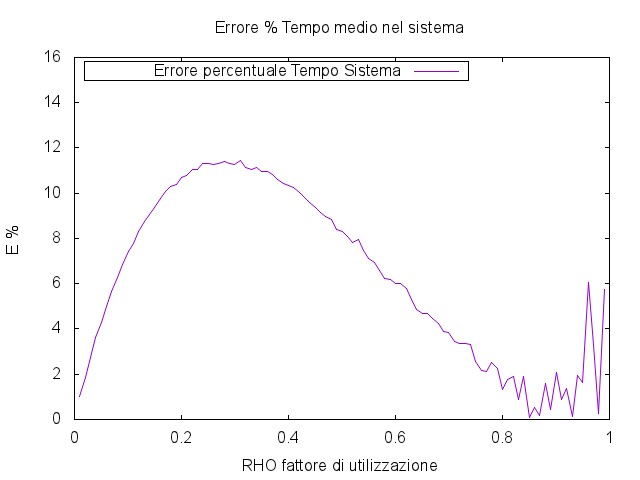


Figura 4.29 Errore percentuale E[T] con 5000000 utenti

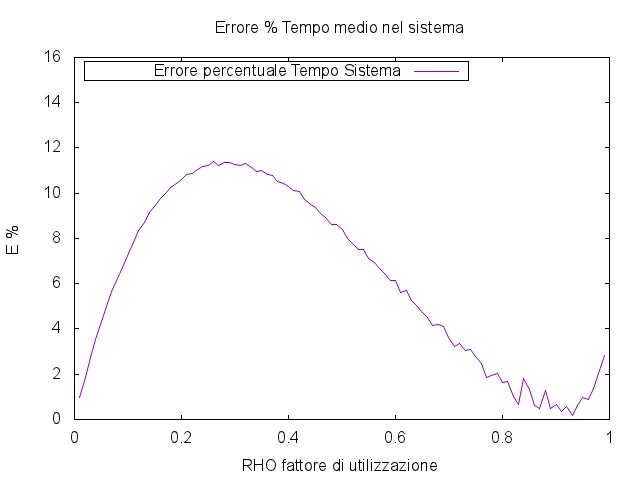


Figura 4.30 Errore percentuale E[T] con 10000000 utenti

### Tempo medio trascorso in coda

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[Tq], al variare del numero di utenti con il quale la simulazione viene effettuata.

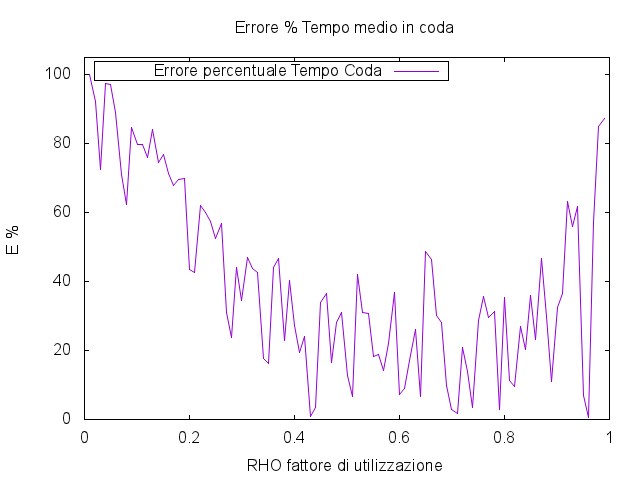


Figura 4.31 Errore percentuale E[Tq] con 1000 utenti

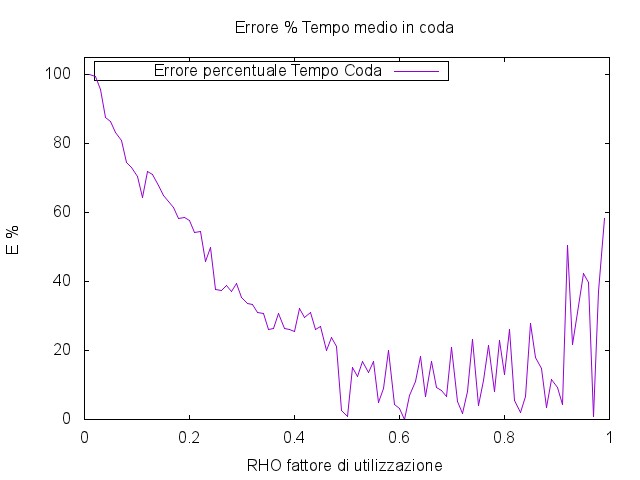


Figura 4.32 Errore percentuale E[Tq] con 10000 utenti

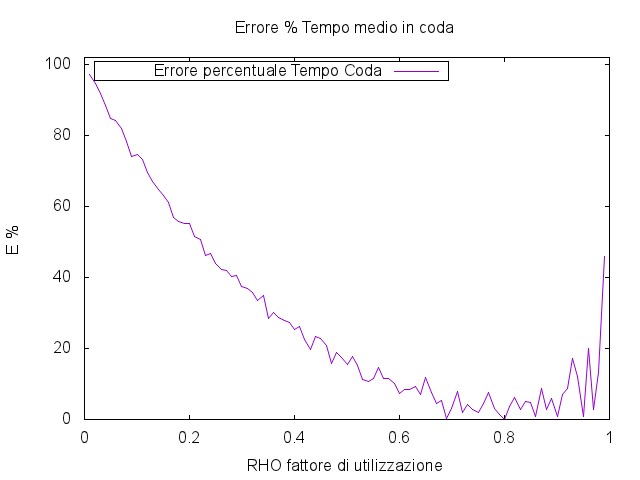


Figura 4.33 Errore percentuale E[Tq] con 100000 utenti

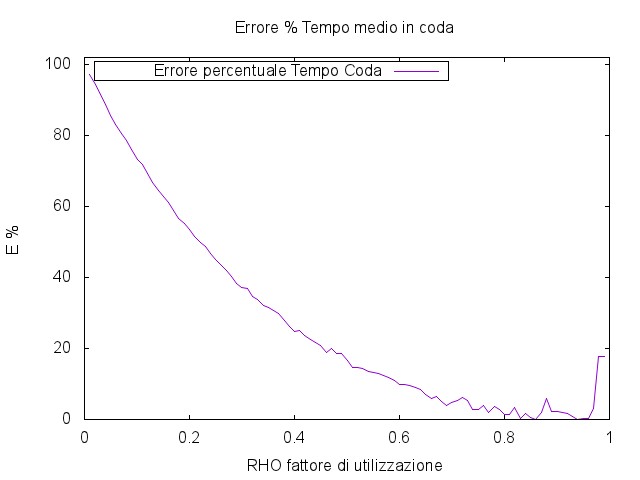


Figura 4.34 Errore percentuale E[Tq] con 1000000 utenti

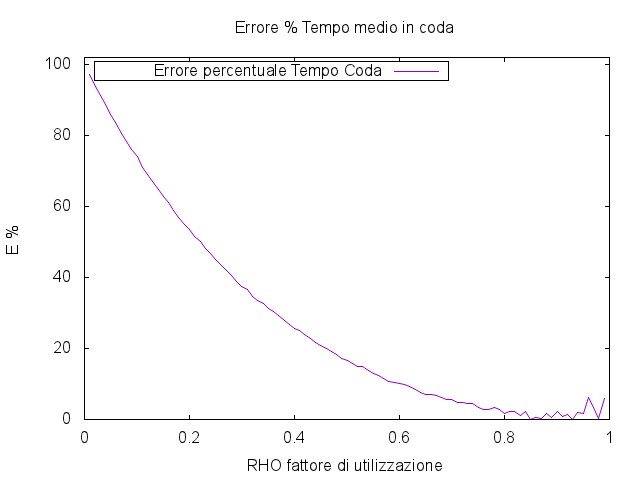


Figura 4.35 Errore percentuale E[Tq] con 5000000 utenti

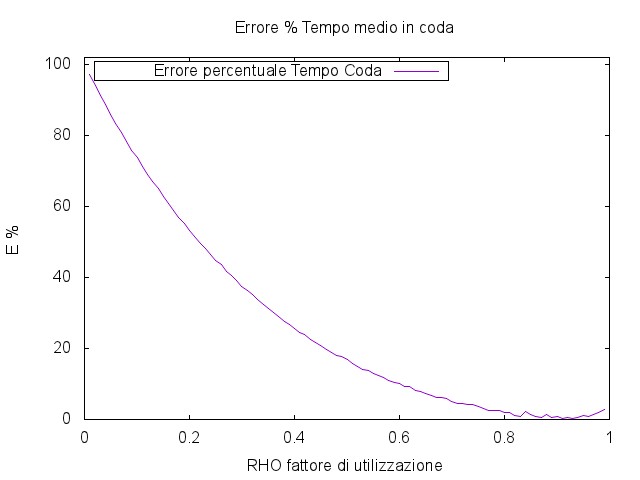


Figura 4.36 Errore percentuale E[Tq] con 10000000 utenti

### Numero medio di utenti nel sistema

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[k], al variare del numero di utenti con il quale la simulazione viene effettuata.

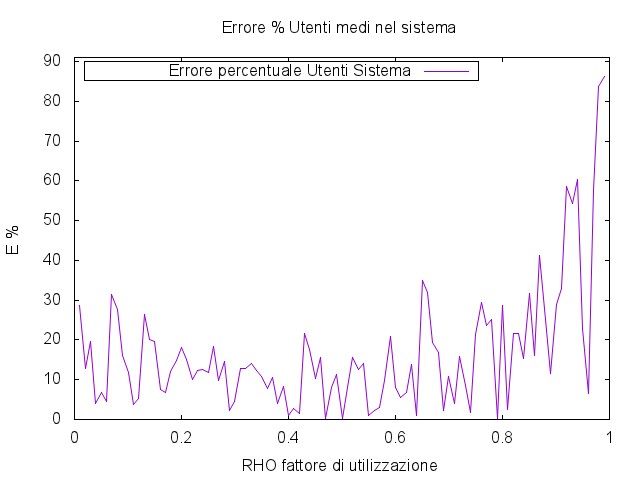


Figura 4.37 Errore percentuale E[k] con 1000 utenti

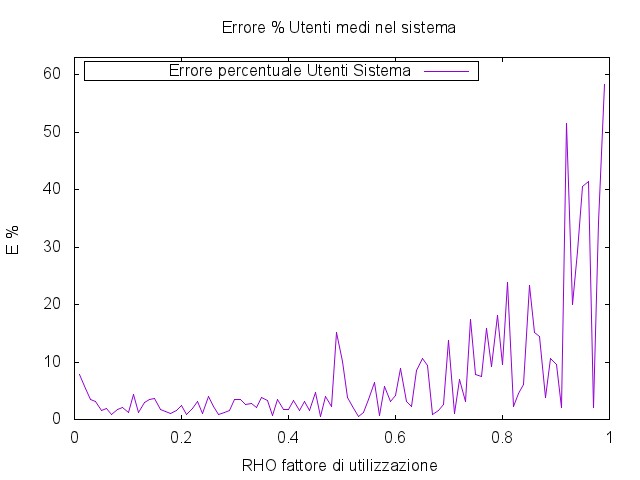


Figura 4.38 Errore percentuale E[k] con 10000 utenti

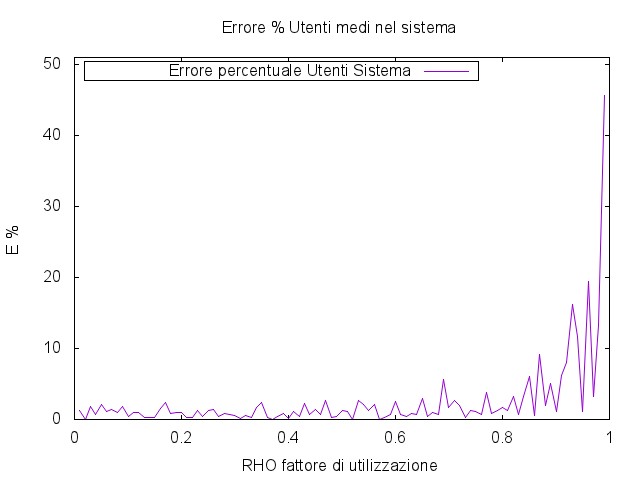


Figura 4.39 Errore percentuale E[k] con 100000 utenti

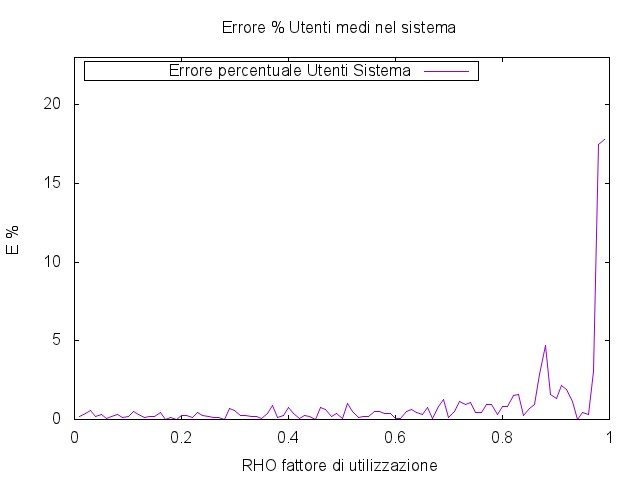


Figura 4.40 Errore percentuale E[k] con 1000000 utenti

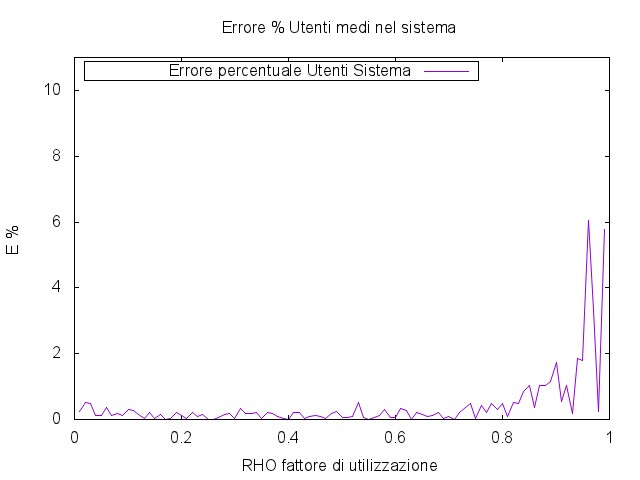


Figura 4.41 Errore percentuale E[k] con 5000000 utenti

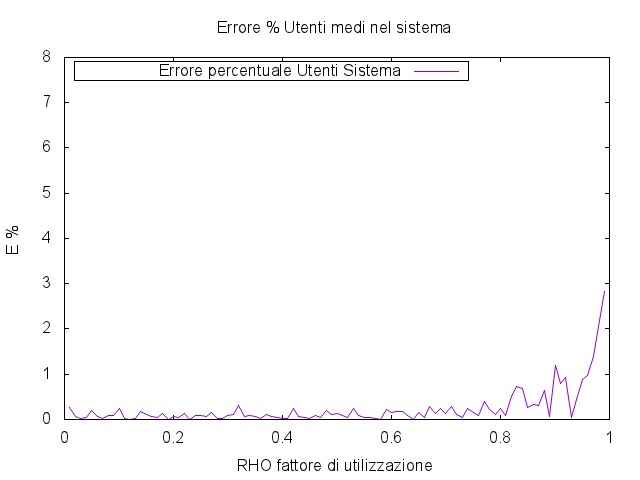


Figura 4.42 Errore percentuale E[k] con 10000000 utenti

### Numero medio di utenti in coda

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[q], al variare del numero di utenti con il quale la simulazione viene effettuata.

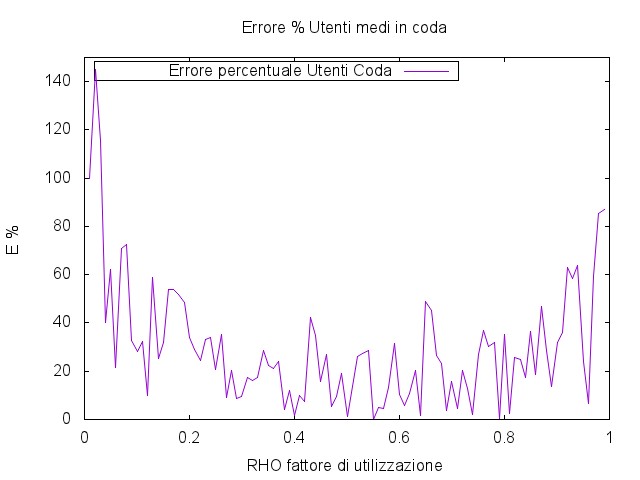


Figura 4.43 Errore percentuale E[q] con 1000 utenti

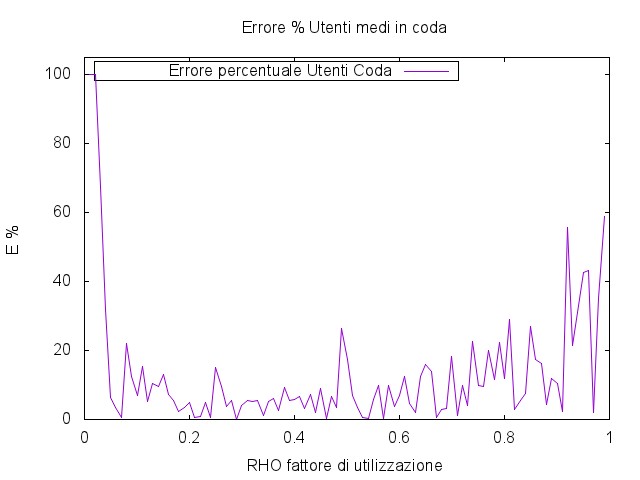


Figura 4.44 Errore percentuale E[q] con 10000 utenti

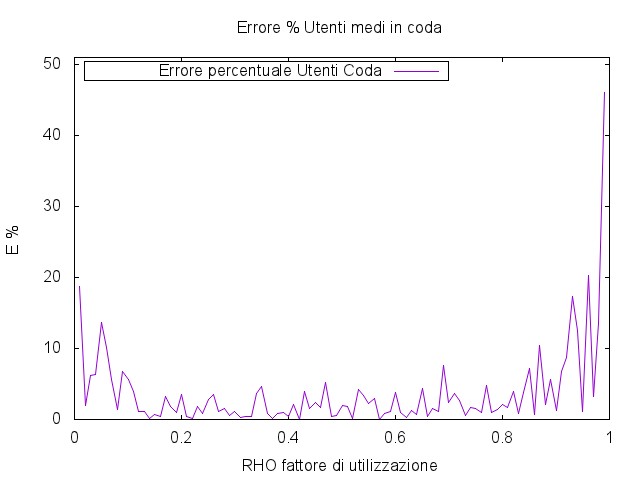


Figura 4.45 Errore percentuale E[q] con 100000 utenti

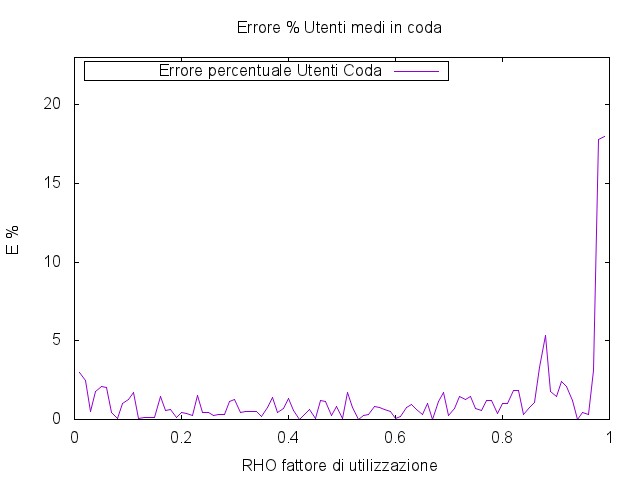


Figura 4.46 Errore percentuale E[q] con 1000000 utenti

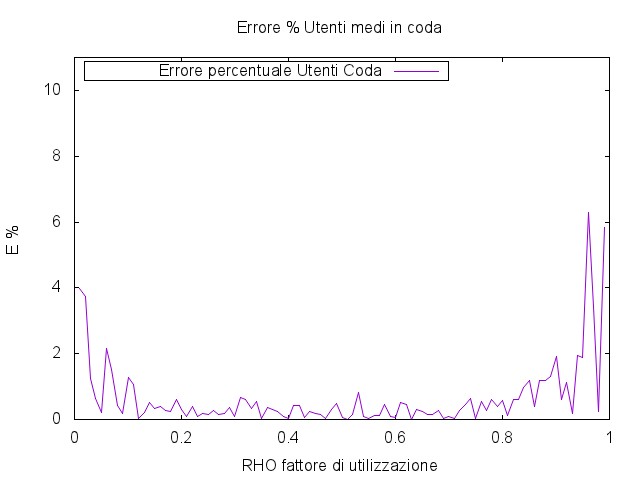


Figura 4.47 Errore percentuale E[q] con 5000000 utenti

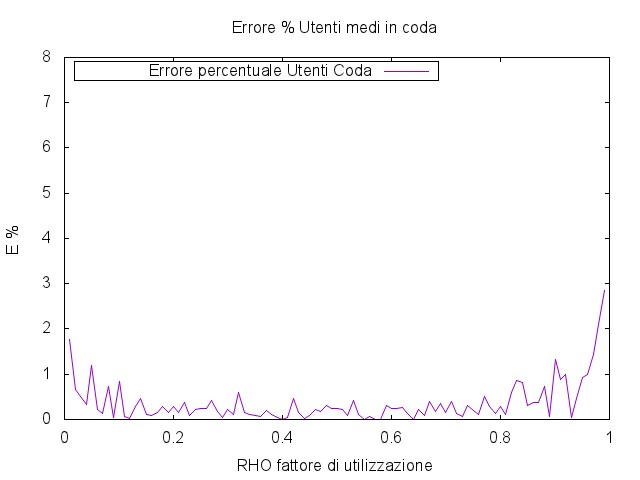


Figura 4.48 Errore percentuale E[q] con 10000000 utenti

# Sistema M/M/1/Y

## Richiami teorici

Un sistema a coda M/M/1/Y ha una distribuzione dei tempi interarrivo e di servizio di tipo esponenziale, un solo servitore e una dimensione della coda di Y. Quando nel sistema sono presenti Y utenti, quelli in arrivo non vengono accettati. In pratica questo equivale a dire che:

Specializzando le soluzioni generali dei processi di nascita e morte in equilibrio, otteniamo:

avendo posto .

Il numero medio di utenti è:

Il numero medio di utenti in coda è:

Il tempo medio speso nel sistema e in coda sono esprimibili mediante il Risultato di Little, con la relazione:

dove .

## Simulazione del sistema M/M/1/Y

Per simulare il comportamento di un sistema M/M/1/Y, basta impostare la dimensione della coda del simulatore inferiore al numero di utenti con il quale la simulazione verrà effettuata. I grafici che seguono sono il risultato di una simulazione effettuata con Y = 10, ma nulla vieta di effettuare simulazioni impostando valori diversi.

### Tempo medio trascorso nel sistema

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[T] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

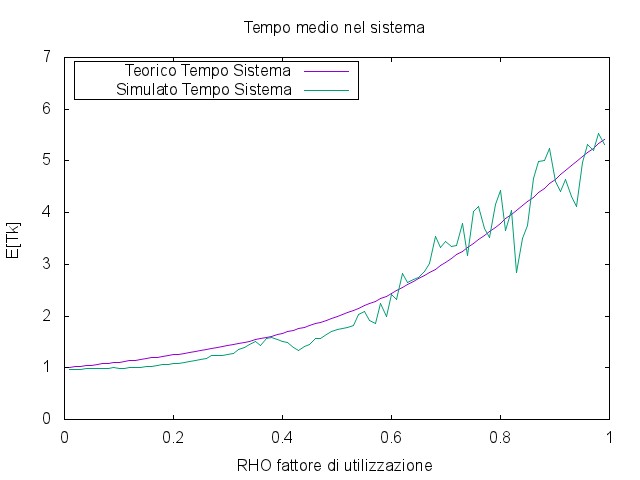


Figura 5.1 Tempo medio speso nel sistema con 1000 utenti

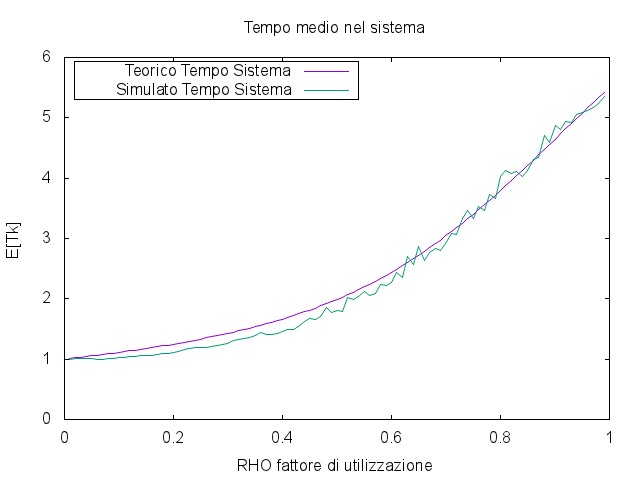


Figura 5.2 Tempo medio speso nel sistema con 10000 utenti

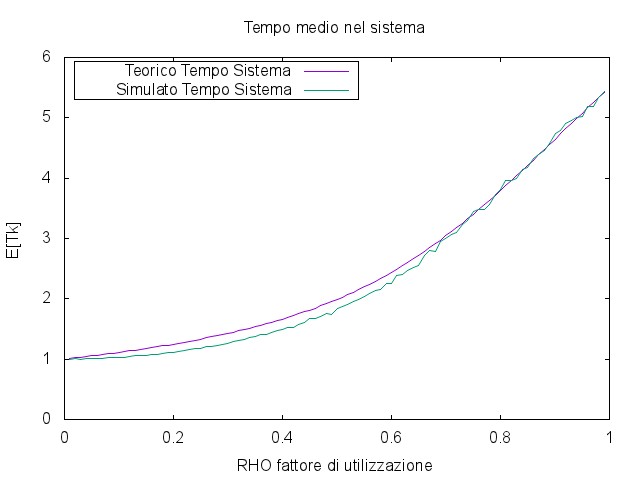


Figura 5.3 Tempo medio speso nel sistema con 100000 utenti

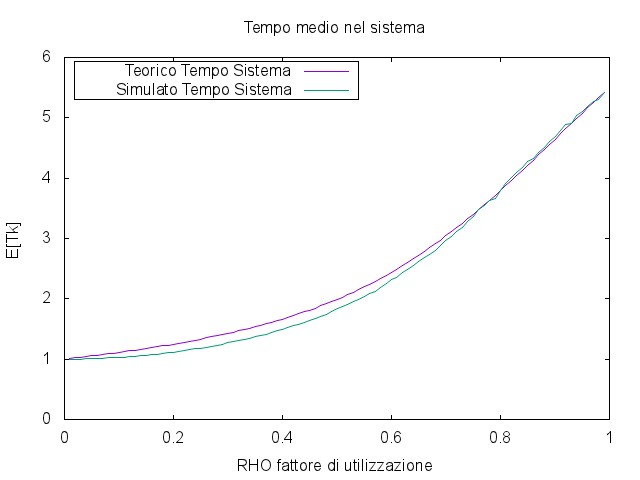


Figura 5.4 Tempo medio speso nel sistema con 1000000 utenti

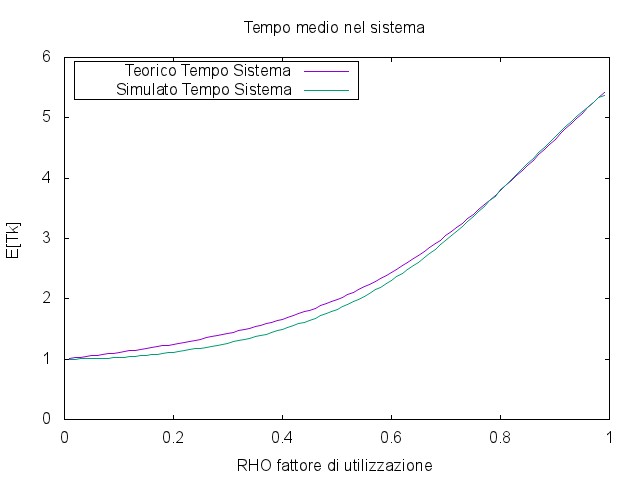


Figura 5.5 Tempo medio speso nel sistema con 5000000 utenti

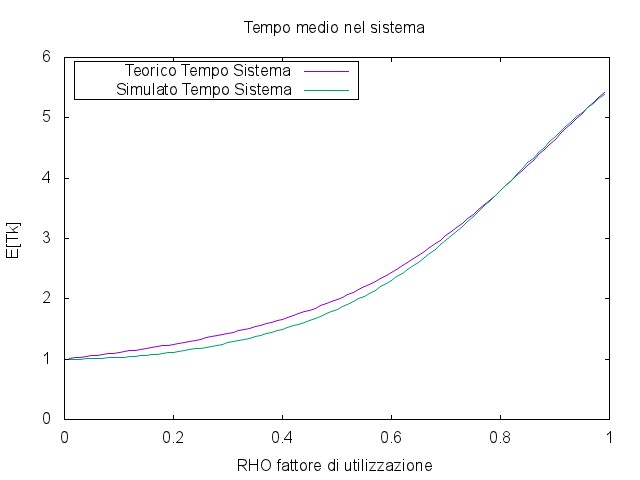


Figura 5.6 Tempo medio speso nel sistema con 10000000 utenti

### Tempo medio trascorso in coda

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[Tq] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

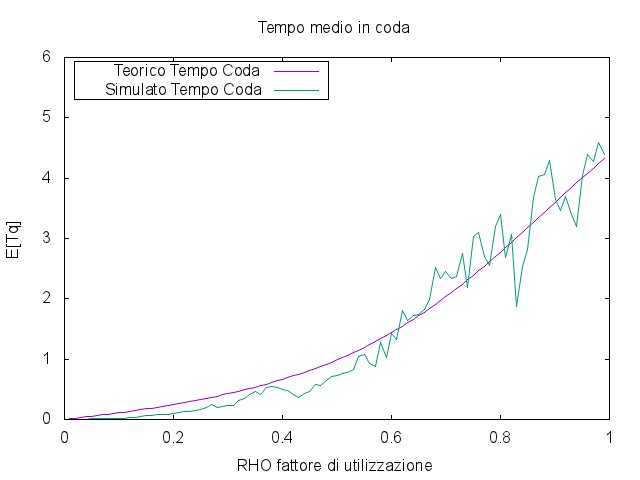


Figura 5.7 Tempo medio speso in coda con 1000 utenti

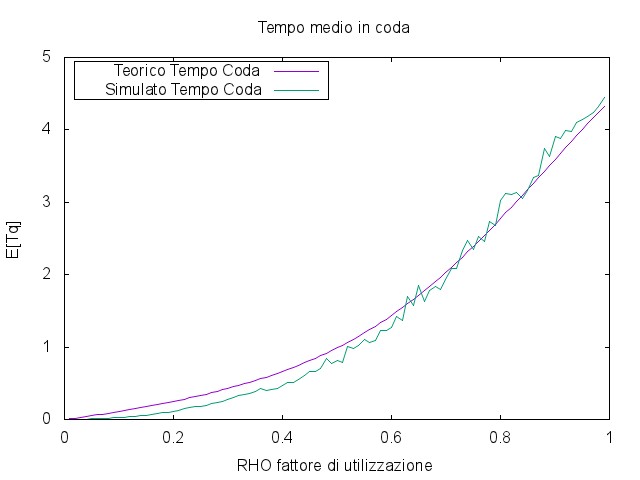


Figura 5.8 Tempo medio speso in coda con 10000 utenti

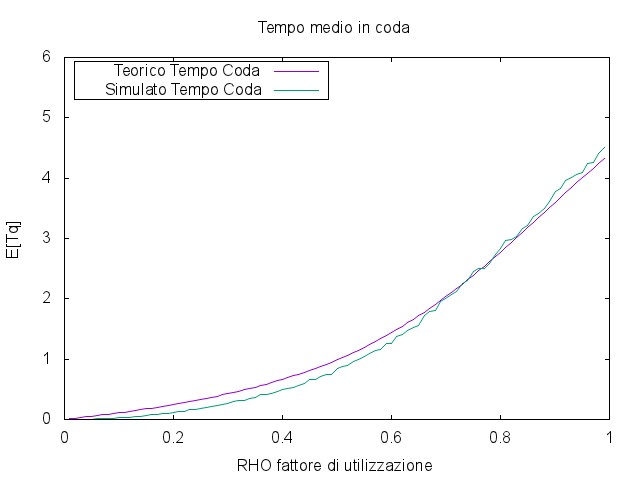


Figura 5.9 Tempo medio speso in coda con 100000 utenti

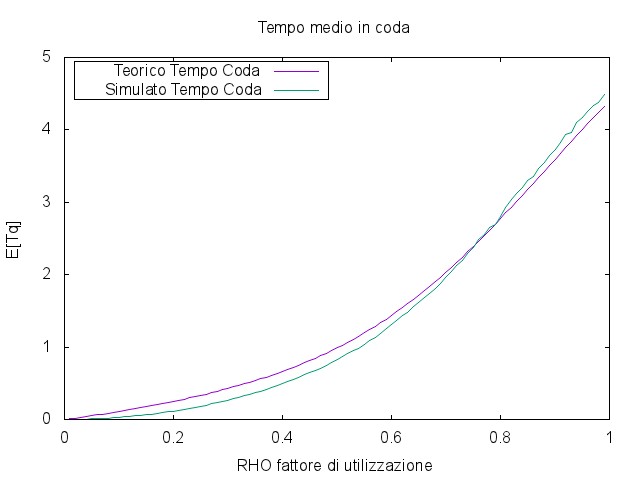


Figura 5.10 Tempo medio speso in coda con 1000000 utenti

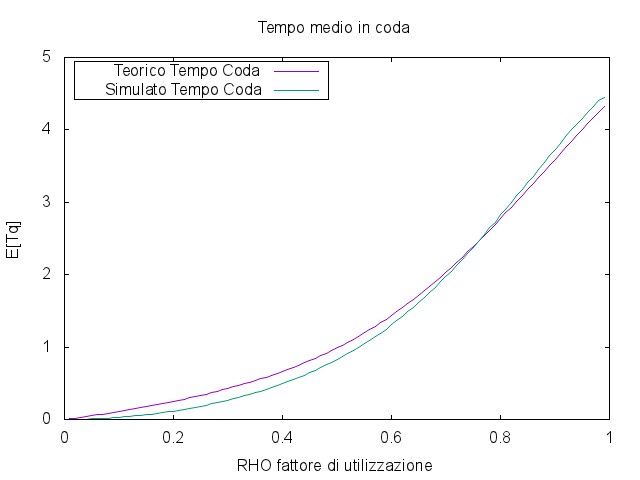


Figura 5.11 Tempo medio speso in coda con 5000000 utenti

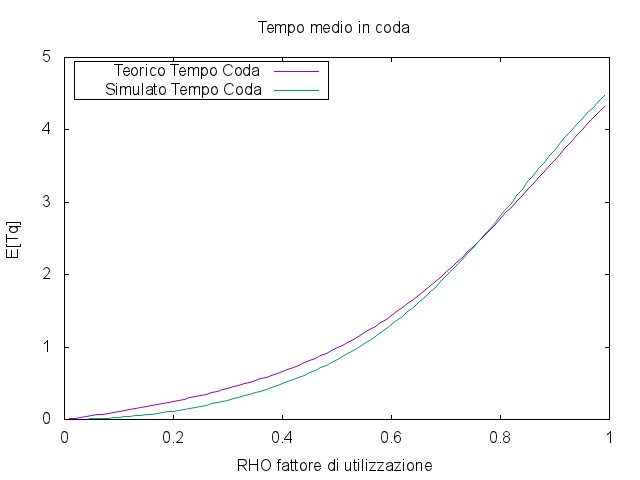


Figura 5.12 Tempo medio speso in coda con 10000000 utenti

### Numero medio di utenti nel sistema

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[k] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

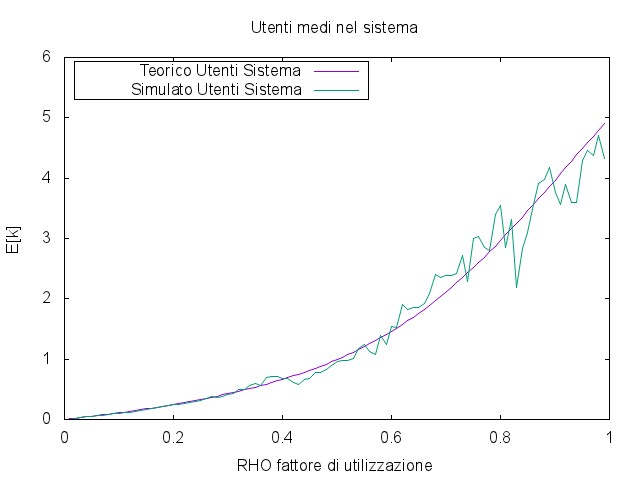


Figura 5.13 Numero medio di utenti nel sistema con 1000 utenti

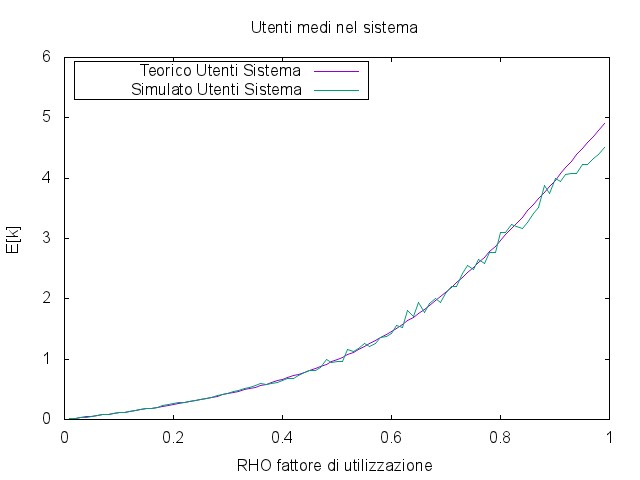


Figura 5.14 Numero medio di utenti nel sistema con 10000 utenti

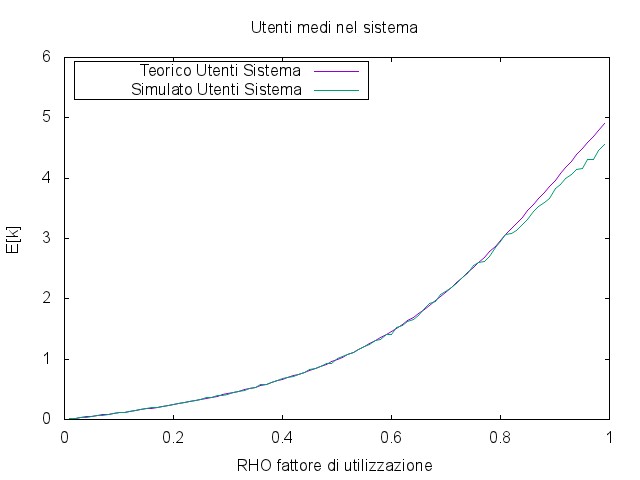


Figura 5.15 Numero medio di utenti nel sistema con 100000 utenti

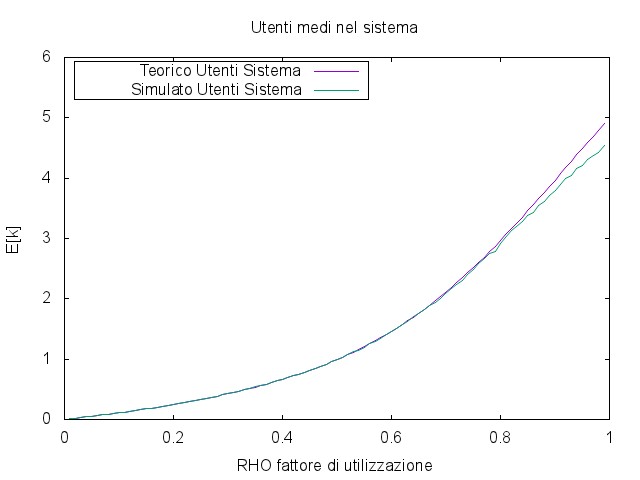


Figura 5.16 Numero medio di utenti nel sistema con 1000000 utenti

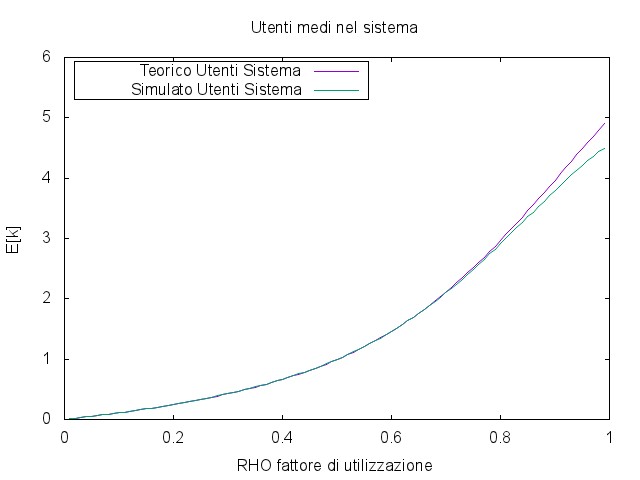


Figura 5.17 Numero medio di utenti nel sistema con 5000000 utenti

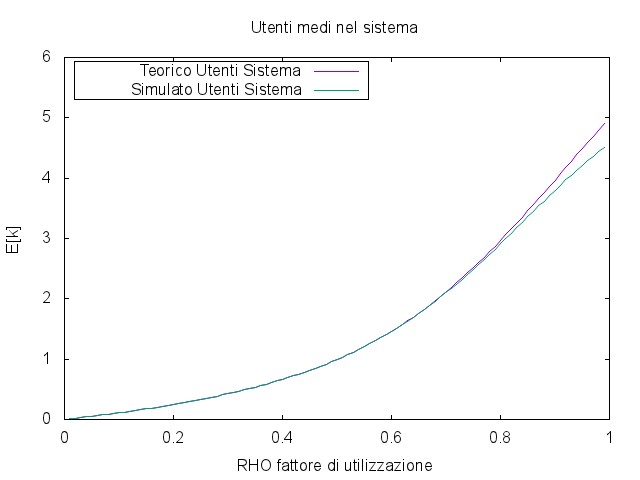


Figura 5.18 Numero medio di utenti nel sistema con 10000000 utenti

### Numero medio di utenti in coda

Vengono ora riportati i grafici generati dal programma che mettono a confronto i valori teorici di E[q] con i risultati ottenuti dalla simulazione, al variare del numero di utenti con il quale la simulazione viene effettuata.

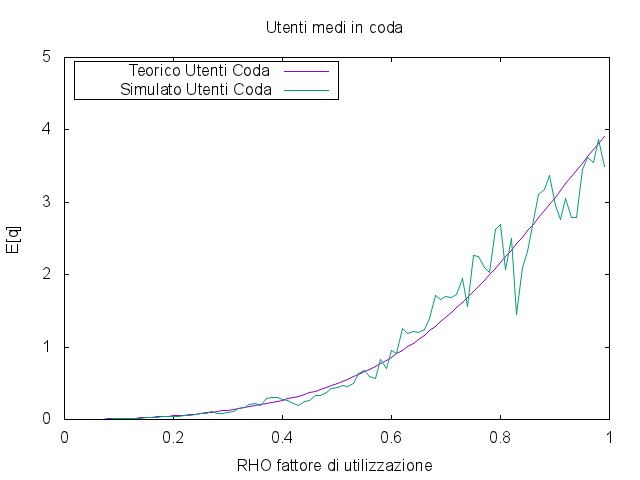


Figura 5.19 Numero medio di utenti in coda con 1000 utenti

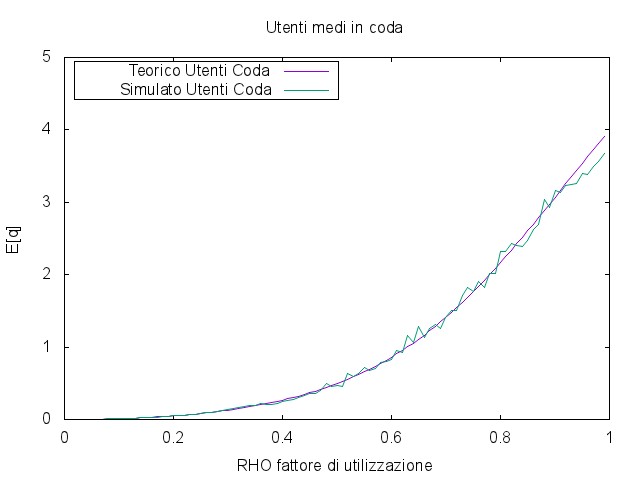


Figura 5.20 Numero medio di utenti in coda con 10000 utenti

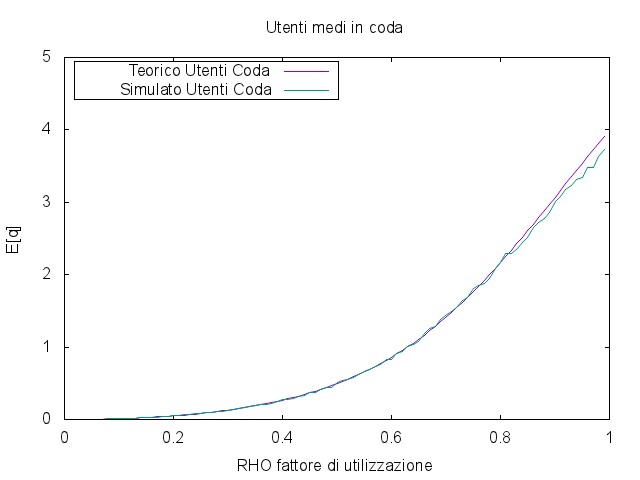


Figura 5.21 Numero medio di utenti in coda con 100000 utenti

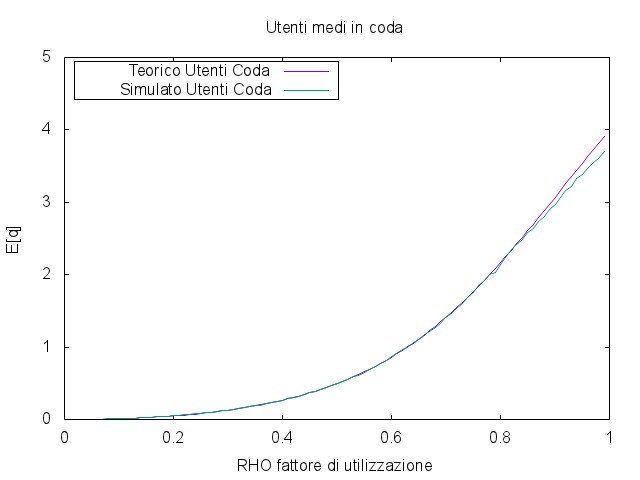


Figura 5.22 Numero medio di utenti in coda con 1000000 utenti

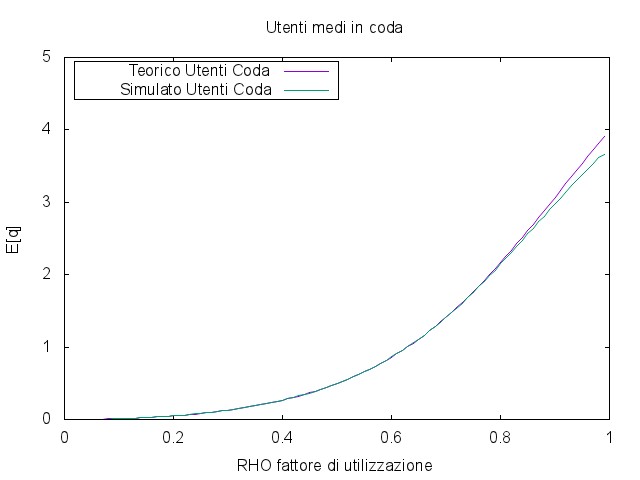


Figura 5.23 Numero medio di utenti in coda con 5000000 utenti

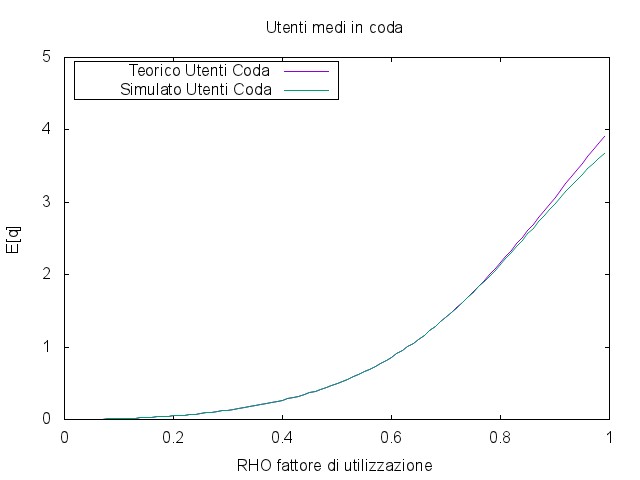


Figura 5.24 Numero medio di utenti in coda con 10000000 utenti

## Errori tra valori teorici e simulati nel sistema M/M/1/Y

### Tempo medio trascorso nel sistema

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[T], al variare del numero di utenti con il quale la simulazione viene effettuata.

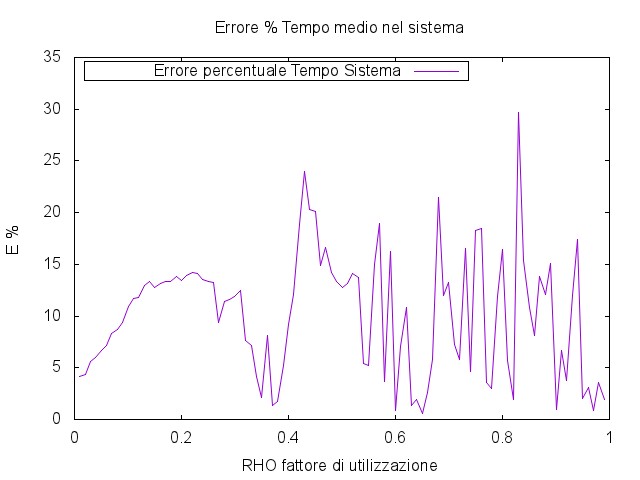


Figura 5.25 Errore percentuale E[T] con 1000 utenti

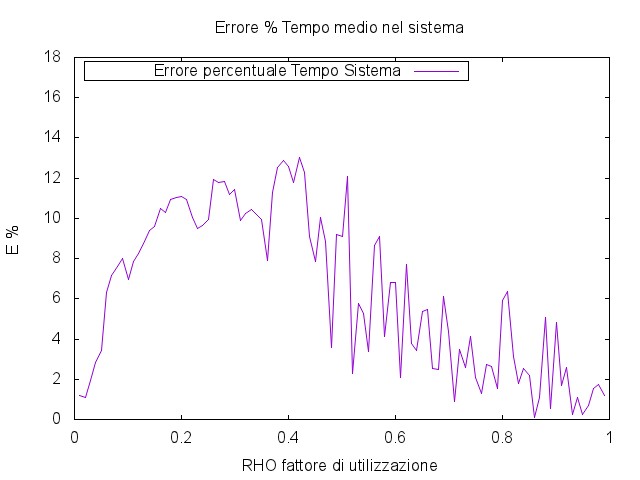


Figura 5.26 Errore percentuale E[T] con 10000 utenti

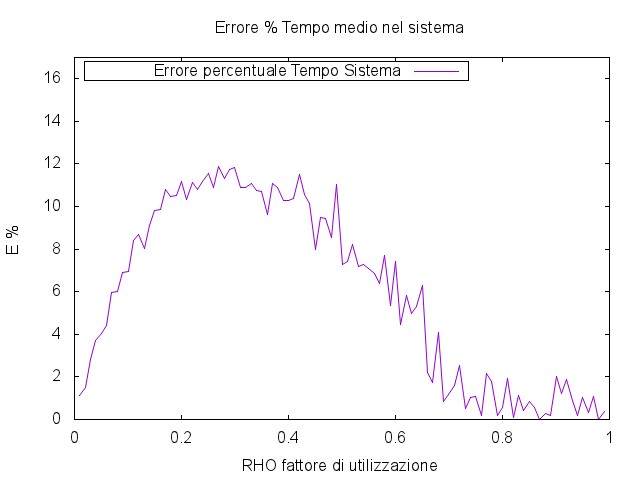


Figura 5.27 Errore percentuale E[T] con 100000 utenti

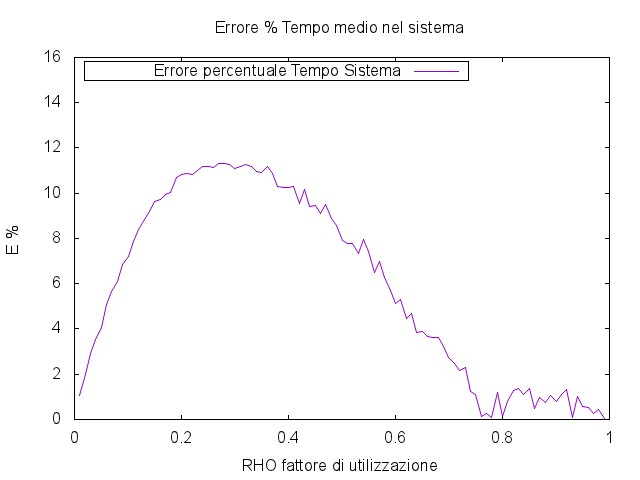


Figura 5.28 Errore percentuale E[T] con 1000000 utenti

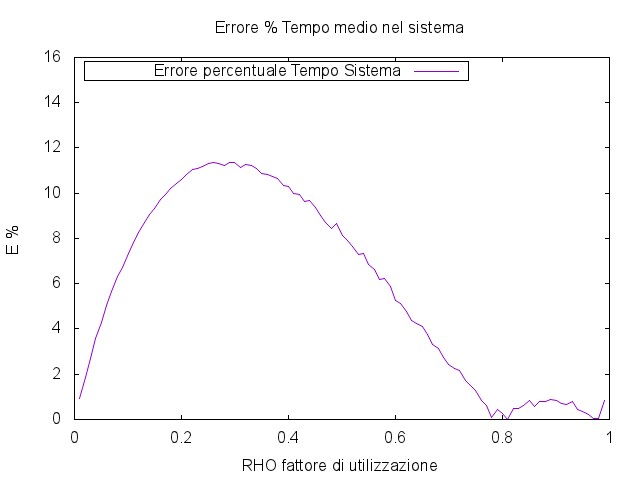


Figura 5.29 Errore percentuale E[T] con 5000000 utenti

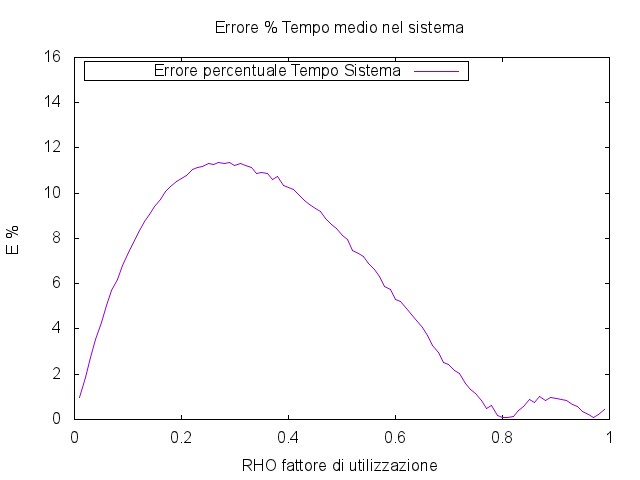


Figura 5.30 Errore percentuale E[T] con 10000000 utenti

### Tempo medio trascorso in coda

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[Tq], al variare del numero di utenti con il quale la simulazione viene effettuata.

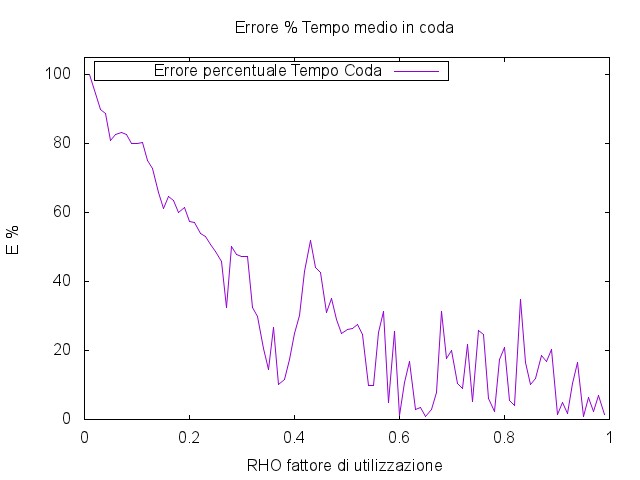


Figura 5.31 Errore percentuale E[Tq] con 1000 utenti

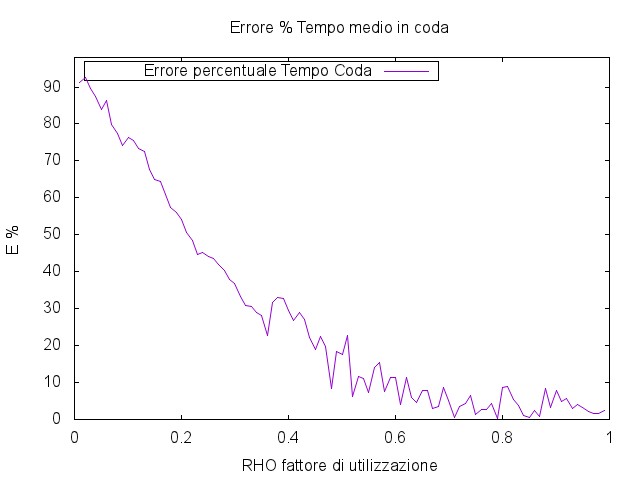


Figura 5.32 Errore percentuale E[Tq] con 10000 utenti

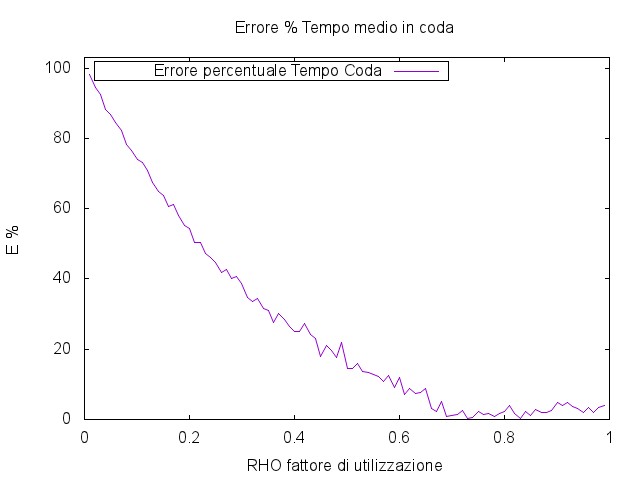


Figura 5.33 Errore percentuale E[Tq] con 100000 utenti

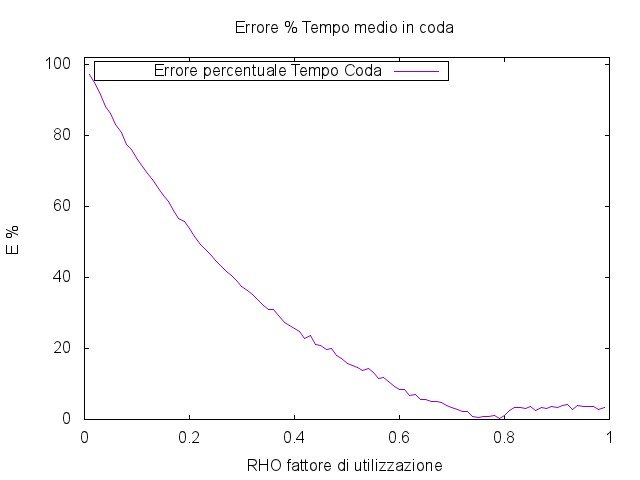


Figura 5.34 Errore percentuale E[Tq] con 1000000 utenti

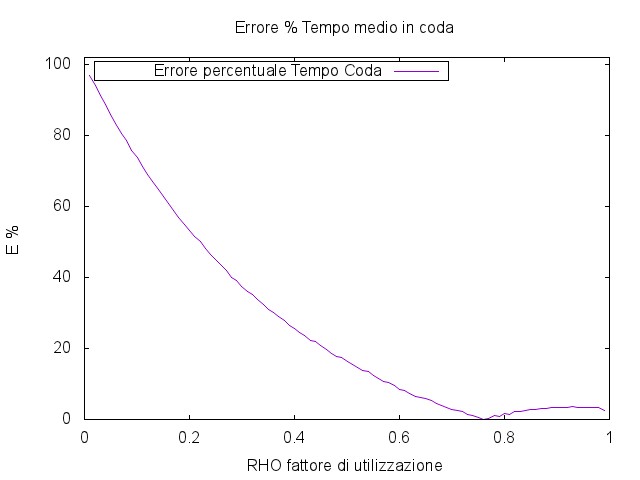


Figura 5.35 Errore percentuale E[Tq] con 5000000 utenti

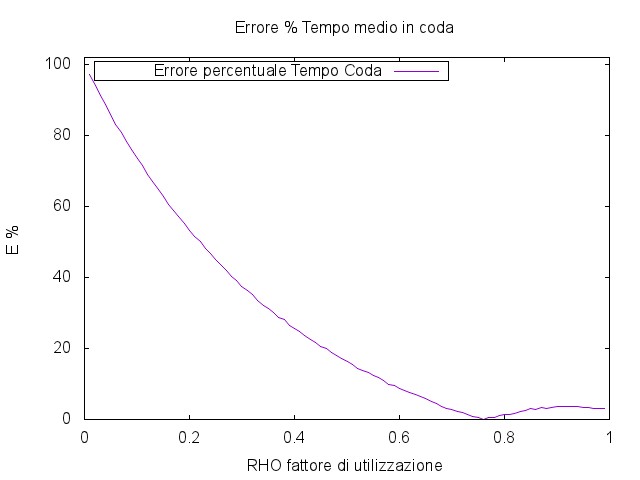


Figura 5.36 Errore percentuale E[Tq] con 10000000 utenti

### Numero medio di utenti nel sistema

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[k], al variare del numero di utenti con il quale la simulazione viene effettuata.

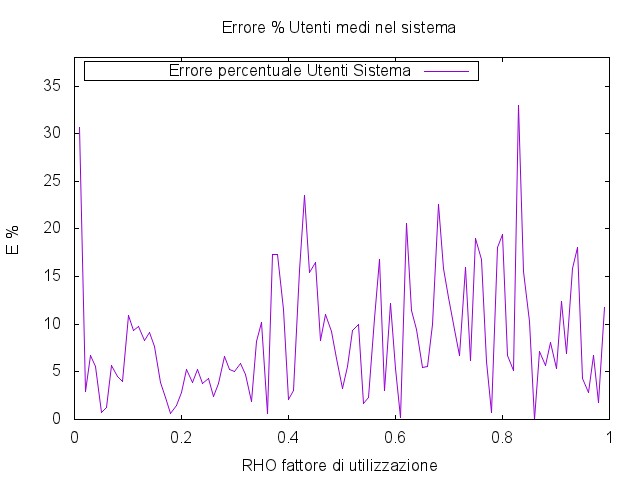


Figura 5.37 Errore percentuale E[k] con 1000 utenti

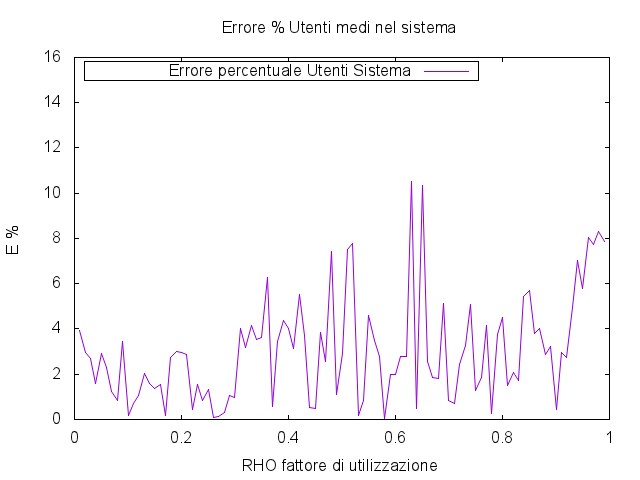


Figura 5.38 Errore percentuale E[k] con 10000 utenti

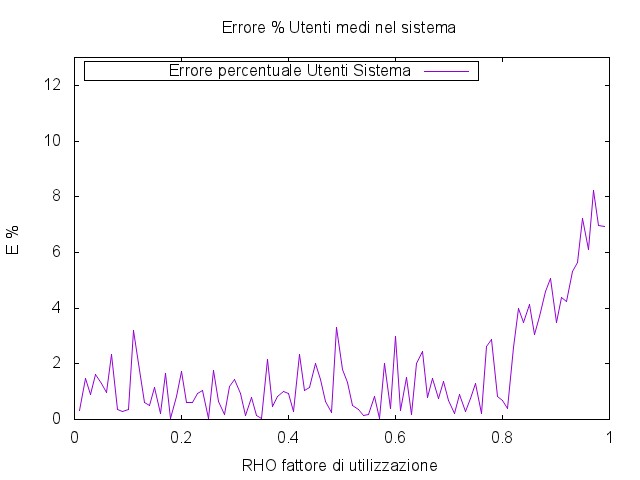


Figura 5.39 Errore percentuale E[k] con 100000 utenti

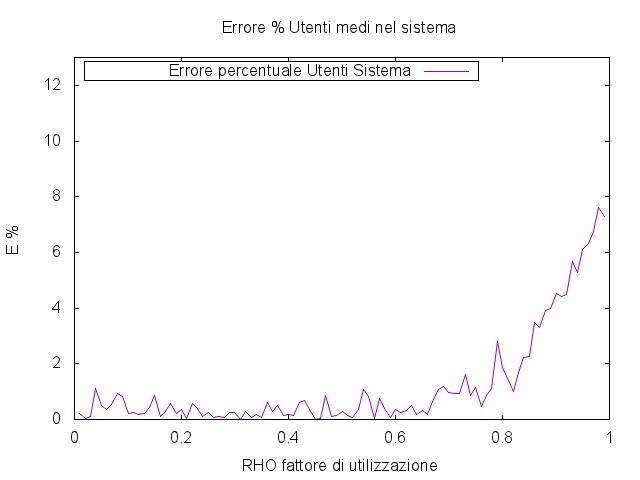


Figura 5.40 Errore percentuale E[k] con 1000000 utenti

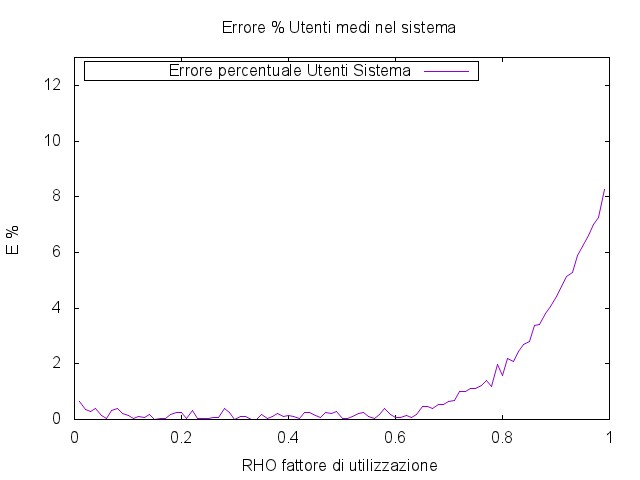


Figura 5.41 Errore percentuale E[k] con 5000000 utenti

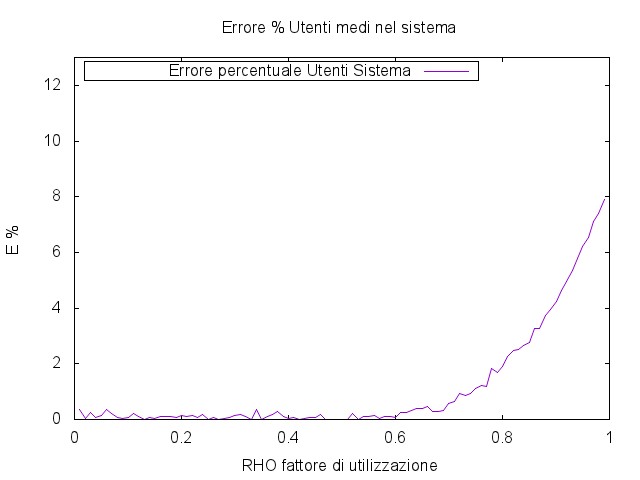


Figura 5.42 Errore percentuale E[k] con 10000000 utenti

### Numero medio di utenti in coda

Vengono ora riportati i grafici generati dal programma che illustrano l’errore percentuale (calcolato come illustrato nel Capitolo 3) riguardante E[q], al variare del numero di utenti con il quale la simulazione viene effettuata.

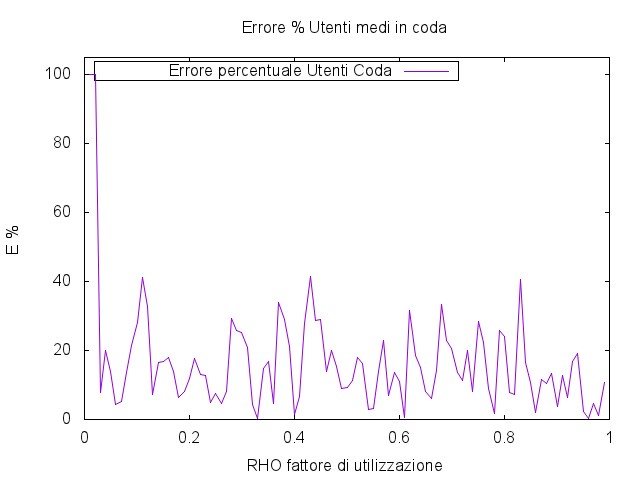


Figura 5.43 Errore percentuale E[q] con 1000 utenti

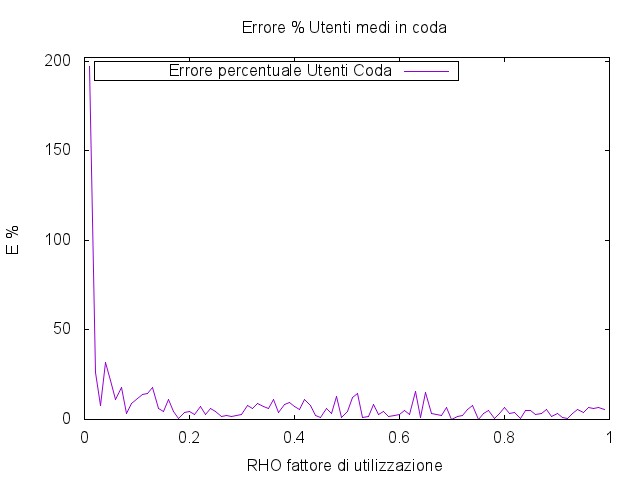


Figura 5.44 Errore percentuale E[q] con 10000 utenti

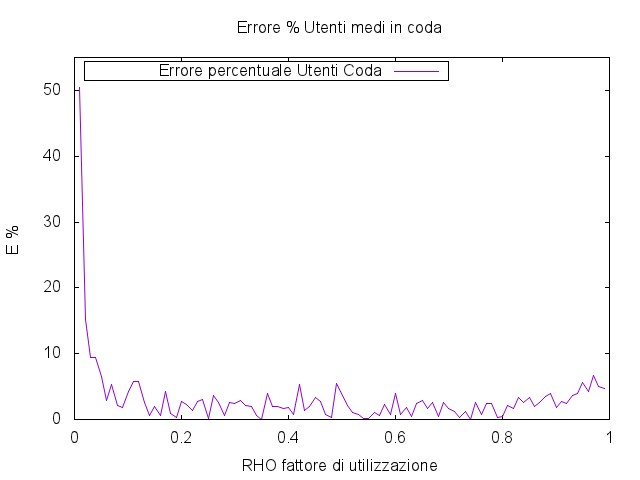


Figura 5.45 Errore percentuale E[q] con 100000 utenti

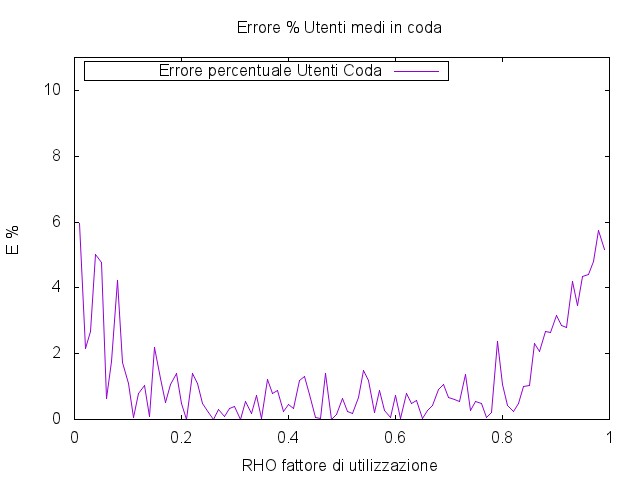


Figura 5.46 Errore percentuale E[q] con 1000000 utenti

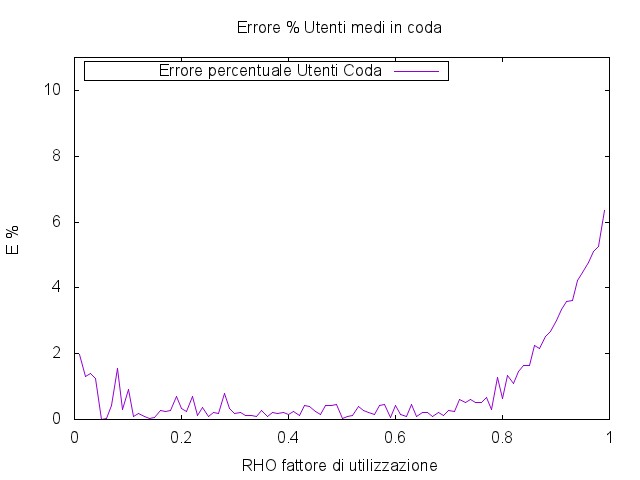


Figura 5.47 Errore percentuale E[q] con 5000000 utenti

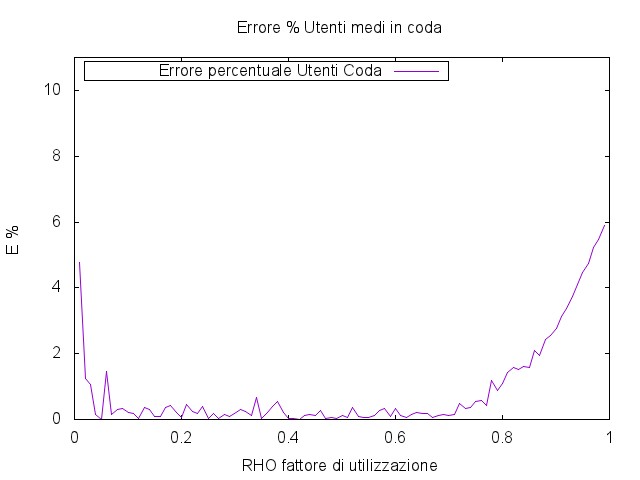


Figura 5.48 Errore percentuale E[q] con 10000000 utenti

# Conclusioni

I risultati simulati confermano quelli teorici con un discreto grado di precisione. Miglioramenti sarebbero ottenibili aumentando il numero di utenti generati per collezionare dati statistici, con conseguente aumento del tempo reale di simulazione. Per questo motivo è possibile impostare il passo di incremento di , in modo da poter diminuire il tempo richiesto dalla simulazione al crescere del numero di utenti. Esiste tuttavia un limite inferiore oltre il quale gli errori relativi non potranno mai scendere, limite imposto dalle ipotesi esemplificative introdotte nel modello teorico.

# Appendice A – Variabile aleatoria distribuita secondo Poisson

La variabile aleatoria X con funzione di densità di probabilità è stata calcolata adottando la tecnica di trasformazione inversa.

1. Si calcola la funzione di distribuzione di probabilità di . Tale funzione (qualora sia possibile calcolarla in forma chiusa) è continua, monotona crescente ed è sempre compresa tra 0 e 1 (per definizione ).
2. Si pone , con u numero casuale distribuito uniformemente nell'intervallo , ossia .
3. Si risolve , ottenendo la variabile aleatoria X distribuita secondo la desiderata , ossia .

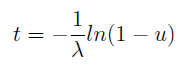
Nel nostro caso volendo una variabile casuale distribuita secondo Poisson, abbiamo considerato la funzione di distribuzione di probabilità:



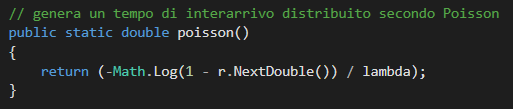
quindi ponendo:



che invertita:



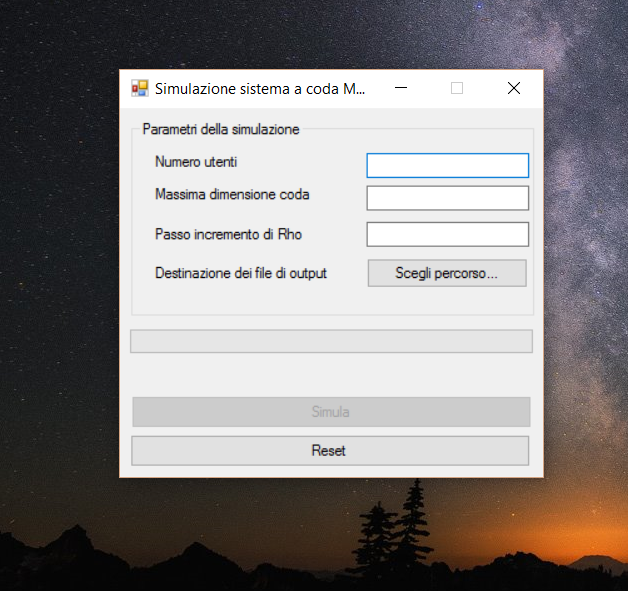
Il codice C# che consente di generare t è allora:

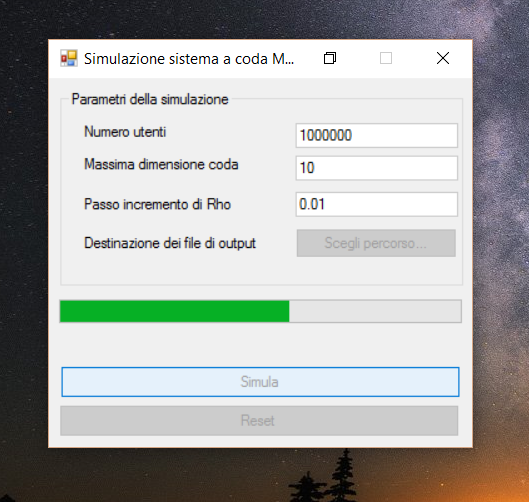


dove r è un oggetto Random, che tramite il metodo NextDouble() ritorna un numero casuale in virgola mobile, con 15 cifre decimali.

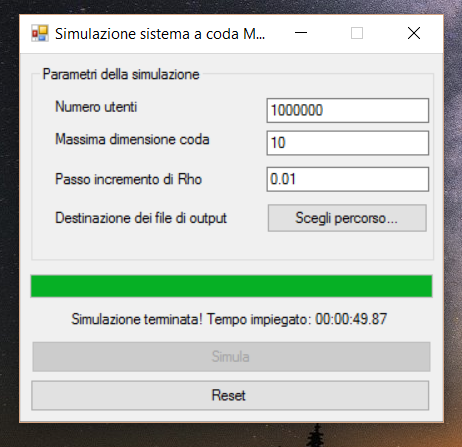
# Appendice B – Breve guida all’uso del simulatore

Una volta lanciato il programma, si aprirà la finestra riportata in figura:



Il pulsante simula non risulterà abilitato finché l’utente non avrà compilato tutti i campi necessari alla simulazione. La textBox contraddistinta dalla label “Numero utenti” indica il campo in cui è necessario inserire il numero di utenti con cui la simulazione verrà effettuata. La textBox contraddistinta dalla label “Massima dimensione coda” indica il campo in cui è necessario inserire la massima dimensione della coda del sistema. Combinando questi due parametri è possibile simulare un sistema M/M/1 o M/M/1/Y. La textBox contraddistinta dalla label “Passo di incremento di Rho” indica il passo di incremento di con cui la simulazione deve essere effettuata. Questo parametro consente di diminuire il tempo richiesto dalla simulazione al crescere del numero di utenti. Infine, il pulsante “Scegli percorso”, apre una finestra di dialogo che consente all’utente di selezionare il percorso in cui il programma creerà i file di output. Una volta impostati questi parametri, il pulsante “Simula” verrà abilitato, e la simulazione può essere avviata. Una progressBar indica l’avanzamento temporale della simulazione, come mostrato in figura:

Al termine della simulazione, comparirà la schermata:



I risultati (compresi i grafici riportati in questa relazione) saranno nel percorso specificato. Premendo il pulsante “Reset” sarà possibile resettare il programma per effettuare una nuova simulazione.

# Appendice C - Codice sorgente del simulatore

Riportiamo ora il codice sorgente del simulatore.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SimulatoreTLC

{

public partial class Form1 : Form

{

static string percorso = "";

static TimeSpan durataSimulazione = new TimeSpan(0, 0, 0);

//variabili della simulazione

static double NUTENTI = 1000000; // numero di utenti del sistema

static double Y = 1000000; // massima dimensione della coda

static double lambda; // tasso di nascita degli utenti

static double MU = 1; // tasso di morte degli utenti

static double INCR\_LAMBDA = 0.01; // incremento sul tasso di nascita degli utenti

static int x; // contatore del numero di utenti attualmente in servizio

static int q; // contatore del numero di utenti attualmente in coda

static int k; // contatore del numero di utenti attualmente nel sistema

static double accumula\_tx; // accumula il tempo trascorso in servizio

static double accumula\_tq; // accumula il tempo trascorso in coda

static double accumula\_tk; // accumula il tempo trascorso nel sistema

static double accumula\_x; // accumula il numero di utenti attualmente in servizio

static double accumula\_q; // accumula il numero di utenti attualmente in coda

static double accumula\_k; // accumula il numero di utenti attualmente nel sistema

static double nUtEntratiServ; // contatore del numero di utenti entrati in servizio

static double nUtEntratiCoda; // accumula il numero di utenti entrati in coda

static double nUtEntratiSist; // accumula il numero di utenti entrati nel sistema

static double nUtUscitiServ; // contatore del numero di utenti usciti dal servizio

static double nUtUscitiCoda; // accumula il numero di utenti usciti in coda

static double nUtUscitiSist; // accumula il numero di utenti usciti nel sistema

static double nUtentiGenerati; // contatore del numero di utenti generati

static double nUtentiPersi; // contatore del numero di utenti persi per coda piena

static double occorrenzeCodaPiena; // numero di volte in cui la coda è piena

static double tInterarrivo; // tempo trascorso fra due arrivi consecutivi di utenti

static double tSimulazione; // accumula il tempo di simulazione

static double tServitoreLibero; // accumula il tempo in cui il servitore è libero

static double tServitoreOccupato; // accumula il tempo in cui il servitore è occupato

static Random r;

static List<Utente> coda;

static Utente utente;

// genera un tempo di interarrivo distribuito secondo Poisson

public static double poisson()

{

return (-Math.Log(1 - r.NextDouble()) / lambda);

}

// genera un tempo di interservizio distribuito secondo Poisson

public static double txUtente()

{

return (-Math.Log(1 - r.NextDouble()) / MU);

}

public static void inizializzaSimulazione()

{

r = new Random(DateTime.Now.Millisecond);

x = 0; // contatore del numero di utenti attualmente in servizio

q = 0; // contatore del numero di utenti attualmente in coda

k = 0; // contatore del numero di utenti attualmente nel sistema

accumula\_tx = 0; // accumula il tempo trascorso in servizio

accumula\_tq = 0; // accumula il tempo trascorso in coda

accumula\_tk = 0; // accumula il tempo trascorso nel sistema

accumula\_x = 0; // accumula il numero di utenti attualmente in servizio

accumula\_q = 0; // accumula il numero di utenti attualmente in coda

accumula\_k = 0; // accumula il numero di utenti attualmente nel sistema

nUtEntratiServ = 0; // contatore del numero di utenti entrati in servizio

nUtEntratiCoda = 0; // accumula il numero di utenti entrati in coda

nUtEntratiSist = 0; // accumula il numero di utenti entrati nel sistema

nUtUscitiServ = 0; // contatore del numero di utenti usciti dal servizio

nUtUscitiCoda = 0; // accumula il numero di utenti usciti in coda

nUtUscitiSist = 0; // accumula il numero di utenti usciti nel sistema

nUtentiGenerati = 0; // contatore del numero di utenti generati

nUtentiPersi = 0; // contatore del numero di utenti persi per coda piena

occorrenzeCodaPiena = 0; // numero di volte in cui la coda è piena

tInterarrivo = 0; // tempo trascorso fra due arrivi consecutivi di utenti

tSimulazione = 0; // accumula il tempo di simulazione

tServitoreLibero = 0; // accumula il tempo in cui il servitore è libero

tServitoreOccupato = 0; // accumula il tempo in cui il servitore è occupato

coda = new List<Utente>();

utente = new Utente();

}

// gestisce l’accodamento di un utente in coda alla lista

public static void inserisciInCoda(Utente utente)

{

coda.Add(utente);

}

// ritorna l’elemento di testa della lista, cancellandolo. // il chiamante si è già assicurato che la coda NON sia vuota

public static Utente prelevaInTesta()

{

Utente u = coda[0];

coda.RemoveAt(0);

return u;

}

// aggiunge agli utenti in coda il tempo time

public static void aggiorna\_tq\_tkCoda(double time)

{

foreach (var u in coda)

{

u.TempoInCoda += time;

}

}

// gestisce l’entrata in servizio dell’utente

public static bool serviUtente(ref Utente utente, ref int x, ref double nUtEntratiServ)

{

// incremento il contatore degli utenti entrati in servizio

nUtEntratiServ++;

x++; // l’utente entra in servizio

utente.TempoServizio = txUtente();

// starà in servizio per un certo tempo txUtente

utente.TempoServizioRimanente = utente.TempoServizio; // setto il tempo di servizio rimanente

return true; // l’utente arrivato è stato gestito

}

// gestisce l’entrata in servizio di un utente quando il sistema è vuoto

public static bool gestisciArrivoUtenteConSistemaVuoto(ref Utente utente, ref int x, ref int k, ref double nUtEntratiServ, ref double nUtEntratiSist)

{

// incremento il contatore degli utenti entrati nel sistema

nUtEntratiSist++;

// accumulo il numero di utenti attualmente nel sistema

accumula\_k = accumula\_k + k;

// accumulo il numero di utenti attualmente in coda

accumula\_q = accumula\_q + q;

k++;

// arriva l’utente che entra nel sistema

utente.TempoServizio = 0;// non è ancora stato messo in servizio

utente.TempoInCoda = 0; // non ha atteso in coda perchè il sistema è vuoto

utente.TempoNelSistema = 0; // è appena entrato nel sistema

utente.TempoServizioRimanente = 0; // non è ancora stato messo in servizio

// gestisce l’entrata in servizio dell’utente

return (serviUtente(ref utente, ref x, ref nUtEntratiServ));

}

// gestisce l’eventuale (MM1Y) accodamento dell’utente

public static bool gestisciAccodamentoUtenteMM1Y(ref int q, ref int k, ref double nUtEntratiCoda, ref double nUtEntratiSist, ref double nUtentiPersi)

{

Utente utDaAccodare = new Utente();

// se la coda NON è piena

if (q < Y)

{

// incremento il contatore del numero totale di utenti entrati nel sistema

nUtEntratiSist++;

// incremento il contatore del numero totale di utenti entrati in coda

nUtEntratiCoda++;

// accumulo il numero di utenti attualmente nel sistema

accumula\_k = accumula\_k + k;

// accumulo il numero di utenti attualmente in coda

accumula\_q = accumula\_q + q;

k++; // arriva l’utente che entra nel sistema

q++; // incremento il numero di utenti in coda

utDaAccodare.TempoServizio = 0; // non è ancora stato messo in servizio

utDaAccodare.TempoInCoda = 0; // entrerà ora in coda

utDaAccodare.TempoNelSistema = 0; // è appena entrato nel sistema

utDaAccodare.TempoServizioRimanente = 0; // non è ancora stato messo in servizio

inserisciInCoda(utDaAccodare); // gestisce l’accodamento

// se la coda è piena aggiorno il contatore di coda piena

if (q == Y)

occorrenzeCodaPiena++;

} // se la coda è piena

else

nUtentiPersi++;// arriva l’utente che trovando la coda piena sarà perso

return true; // l’utente arrivato è stato gestito

}

// gestisce l’uscita dell’utente attualmente in servizio

public static void gestisciUscitaUtente(Utente utente, ref int x, ref int k, ref double nUtUscitiServ, ref double nUtUscitiSist)

{

// incremento il contatore del numero totale di utenti usciti dal servizio

nUtUscitiServ++;

// incremento il contatore del numero totale di utenti usciti dal sistema

nUtUscitiSist++;

// accumulo i tempi di servizio

accumula\_tx = accumula\_tx + utente.TempoServizio;

// accumulo i tempi di permanenza nel sistema

accumula\_tk = accumula\_tk + utente.TempoNelSistema;

x--; // l’utente esce dal servizio

k--; // ed esce dal sistema

}

// gestisce l’entrata in servizio del primo utente in coda (FIFO)

public static void gestisciServiUtenteInCoda(ref Utente utente, ref int x, ref int q, ref double nUtEntratiServ, ref double nUtUscitiCoda)

{

// incremento il contatore del numero totale di utenti usciti dalla coda

nUtUscitiCoda++;

// accumulo i tempi di attesa in coda

accumula\_tq = accumula\_tq + utente.TempoInCoda;

q--; // l’utente esce dalla coda

utente = prelevaInTesta(); // prelevo il primo utente dalla coda

// gestisco l’entrata in servizio dell’utente

serviUtente(ref utente, ref x, ref nUtEntratiServ);

}

public static void simulazione()

{

bool arrivato; // 1 se è arrivato un utente, 0 se NON è arrivato alcun utente

// arriva il primo utente che trova il sistema vuoto

arrivato = gestisciArrivoUtenteConSistemaVuoto(ref utente, ref x, ref k, ref nUtEntratiServ, ref nUtEntratiSist);

nUtentiGenerati = 0;

while (nUtentiGenerati < NUTENTI)

{

// se l’utente è già arrivato

if (arrivato)

{

nUtentiGenerati++; // genero un arrivo

// il prossimo utente arriverà fra un tempo tInterarrivo

tInterarrivo = poisson();

arrivato = false; // poichè NON è ancora arrivato

}

// se il sistema è vuoto

if (k == 0)

{

// la simulazione avanza temporalmente fino all’istante di arrivo

tSimulazione = tSimulazione + tInterarrivo;

// accumulo il tempo di servitore libero

tServitoreLibero = tServitoreLibero + tInterarrivo;

// arriva l’utente che trova il sistema vuoto

arrivato = gestisciArrivoUtenteConSistemaVuoto(ref utente, ref x, ref k, ref nUtEntratiServ, ref nUtEntratiSist);

} // se il sistema NON è vuoto

else

{

// se l’utente arriverà prima che termini il servizio di quello attualmente in servizio

if (tInterarrivo < utente.TempoServizioRimanente)

{

// la simulazione avanza temporalmente fino all’istante di arrivo

tSimulazione = tSimulazione + tInterarrivo;

// trascorre il tempo tInterarrivo per l’utente in servizio

utente.TempoNelSistema = utente.TempoNelSistema + tInterarrivo;

// aggiorno il tempo di servizio rimanente

utente.TempoServizioRimanente = utente.TempoServizioRimanente - tInterarrivo;

// trascorre il tempo tInterarrivo per gli EVENTUALI utenti in coda

aggiorna\_tq\_tkCoda(tInterarrivo);

// gestisco l’eventuale (MM1Y) accodamento dell’utente

arrivato = gestisciAccodamentoUtenteMM1Y(ref q, ref k, ref nUtEntratiCoda, ref nUtEntratiSist, ref nUtentiPersi);

} // se l’utente arriverà dopo che termini il servizio di quello attualmente in servizio

else

{

// la simulazione avanza temporalmente fino all’istante di uscita

tSimulazione = tSimulazione + utente.TempoServizioRimanente;

// trascorre il tempo txRimanente per l’utente in servizio

utente.TempoNelSistema = utente.TempoNelSistema + utente.TempoServizioRimanente;

// trascorre il tempo txRimanente per gli EVENTUALI utenti in coda

aggiorna\_tq\_tkCoda(utente.TempoServizioRimanente);

// trascorre il tempo txRimanente per l’utente in arrivo

tInterarrivo = tInterarrivo - utente.TempoServizioRimanente;

// gestisce l’uscita dell’utente attualmente in servizio

gestisciUscitaUtente(utente, ref x, ref k, ref nUtUscitiServ, ref nUtUscitiSist);

// se c’è coda

if (q > 0)

{

// gestisce l’entrata in servizio del primo utente in coda (FIFO)

gestisciServiUtenteInCoda(ref utente, ref x, ref q, ref nUtEntratiServ, ref nUtUscitiCoda);

}

}// se l’utente arriverà dopo che termini il servizio di quello attualmente in servizio

}// se il sistema NON è vuoto

}// while (nUtentiGenerati < NUTENTI)

}

public static void stampaRisultatiSimulati(string percorso, List<double> r, List<double> tq, List<double> tk, List<double> k, List<double> q, List<double> x)

{

StreamWriter ftempoCoda = new StreamWriter(percorso + "\\simulati\\tempoCoda.txt", false);

StreamWriter ftempoSistema = new StreamWriter(percorso + "\\simulati\\tempoSistema.txt", false);

StreamWriter fUtentiCoda = new StreamWriter(percorso + "\\simulati\\utentiCoda.txt", false);

StreamWriter fUtentiSistema = new StreamWriter(percorso + "\\simulati\\utentiSistema.txt", false);

for (int i = 0; i < tq.Count; i++)

{

ftempoCoda.Write(r[i].ToString().Replace(',', '.'));

ftempoCoda.Write(' ');

ftempoCoda.Write(tq[i].ToString().Replace(',', '.'));

ftempoCoda.WriteLine();

ftempoSistema.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), tk[i].ToString().Replace(',', '.'));

fUtentiSistema.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), k[i].ToString().Replace(',', '.'));

fUtentiCoda.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), q[i].ToString().Replace(',', '.'));

}

ftempoCoda.Close();

ftempoSistema.Close();

fUtentiCoda.Close();

fUtentiSistema.Close();

}

public static void teoricoMM1(out double teoricoUtentiSistema, out double teoricoUtentiCoda, out double teoricoTempoCoda, out double teoricoTempoSistema, out double teoricoTempoServizio)

{

double rho = lambda / (double)MU;

teoricoUtentiSistema = rho / (1 - rho);

teoricoUtentiCoda = Math.Pow(rho, 2) / (1 - rho);

teoricoTempoSistema = teoricoUtentiSistema \* (1 / lambda);

teoricoTempoServizio = 1 / MU;

teoricoTempoCoda = teoricoTempoSistema - teoricoTempoServizio;

}

public static void teoricoMM1Y(out double teoricoUtentiSistema, out double teoricoUtentiCoda, out double teoricoTempoCoda, out double teoricoTempoSistema, out double teoricoTempoServizio)

{

double a = lambda / (double)MU;

double py = ((1 - a) / (1 - Math.Pow(a, Y + 1))) \* Math.Pow(a, Y);

double lambdaMedio = lambda \* (1 - py);

teoricoUtentiSistema = (a / (1 - a)) - (((Y + 1) \* Math.Pow(a, Y + 1)) / (1 - Math.Pow(a, Y + 1)));

teoricoUtentiCoda = (Math.Pow(a, 2) / (1 - a)) - (((Y + 1) \* Math.Pow(a, Y + 1)) / (1 - Math.Pow(a, Y + 1)));

teoricoTempoSistema = teoricoUtentiSistema / lambdaMedio;

teoricoTempoServizio = 1 / MU;

teoricoTempoCoda = teoricoUtentiCoda / lambdaMedio;

}

public static void stampaTeorici(string percorso, List<double> r, List<double> tq, List<double> tk, List<double> k, List<double> q, List<double> x)

{

StreamWriter fTtempoCoda = new StreamWriter(percorso + "\\teorici\\tempoCoda.txt", false);

StreamWriter fTtempoSistema = new StreamWriter(percorso + "\\teorici\\tempoSistema.txt", false);

StreamWriter fTUtentiCoda = new StreamWriter(percorso + "\\teorici\\utentiCoda.txt", false);

StreamWriter fTUtentiSistema = new StreamWriter(percorso + "\\teorici\\utentiSistema.txt", false);

for (int i = 0; i < tq.Count; i++)

{

fTtempoCoda.Write(r[i].ToString().Replace(',', '.'));

fTtempoCoda.Write(' ');

fTtempoCoda.Write(tq[i].ToString().Replace(',', '.'));

fTtempoCoda.WriteLine();

fTtempoSistema.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), tk[i].ToString().Replace(',', '.'));

fTUtentiSistema.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), k[i].ToString().Replace(',', '.'));

fTUtentiCoda.WriteLine("{0} {1}", r[i].ToString().Replace(',', '.'), q[i].ToString().Replace(',', '.'));

}

fTtempoCoda.Close();

fTtempoSistema.Close();

fTUtentiCoda.Close();

fTUtentiSistema.Close();

}

public static void CalcolaErrore(List<double> stk, List<double> stq, List<double> sk, List<double> sq, List<double> ttk, List<double> ttq, List<double> tk, List<double> tq, out List<double> erroreTk, out List<double> erroreTq, out List<double> erroreK, out List<double> erroreQ)

{

erroreK = new List<double>();

erroreQ = new List<double>();

erroreTk = new List<double>();

erroreTq = new List<double>();

for (int i = 0; i < sk.Count; i++)

{

erroreK.Add((Math.Abs(sk[i] - tk[i]) / tk[i]) \* 100);

}

for (int i = 0; i < stk.Count; i++)

{

erroreTk.Add((Math.Abs(stk[i] - ttk[i]) / ttk[i]) \* 100);

}

for (int i = 0; i < stq.Count; i++)

{

erroreTq.Add((Math.Abs(stq[i] - ttq[i]) / tk[i]) \* 100);

}

for (int i = 0; i < sq.Count; i++)

{

erroreQ.Add((Math.Abs(sq[i] - tq[i]) / tq[i]) \* 100);

}

}

public static string CreaCartelle(string percorso)

{

string newPath;

if(Directory.Exists(percorso + "\\risultati"))

{

int i = 0;

do

{

i++;

} while (Directory.Exists(percorso + "\\risultati(" + i.ToString() + ")"));

percorso = percorso + "\\risultati(" + i.ToString() + ")";

Directory.CreateDirectory(percorso + "\\errore");

Directory.CreateDirectory(percorso + "\\teorici");

Directory.CreateDirectory(percorso + "\\simulati");

Directory.CreateDirectory(percorso + "\\grafici");

Directory.CreateDirectory(percorso + "\\script");

}

else

{

Directory.CreateDirectory(percorso + "\\risultati\\errore");

Directory.CreateDirectory(percorso + "\\risultati\\teorici");

Directory.CreateDirectory(percorso + "\\risultati\\simulati");

Directory.CreateDirectory(percorso + "\\risultati\\grafici");

Directory.CreateDirectory(percorso + "\\risultati\\script");

percorso = percorso + "\\risultati";

}

newPath = percorso;

return newPath;

}

public static double TrovaMax(List<double> lista)

{

double max = 0;

foreach(var i in lista)

{

if (max < i)

max = i;

}

return max;

}

public static double TrovaMax(double a, double b)

{

if (a < b)

return b;

else

return a;

}

public static void CreaScript(List<double> erroreTq, List<double> erroreTk, List<double> erroreK, List<double> erroreQ, List<double> teoricoTq, List<double> teoricoTk, List<double> teoricoQ, List<double> teoricoK, List<double> simulatoTk, List<double> simulatoTq, List<double> simulatoK, List<double> simulatoQ)

{

//massimi errori

double maxETq = TrovaMax(erroreTq);

double maxETk = TrovaMax(erroreTk);

double maxEQ = TrovaMax(erroreQ);

double maxEK = TrovaMax(erroreK);

//massimi teorici

double maxTTq = TrovaMax(teoricoTq);

double maxTTk = TrovaMax(teoricoTk);

double maxTQ = TrovaMax(teoricoQ);

double maxTK = TrovaMax(teoricoK);

//massimi simulati

double maxSTq = TrovaMax(simulatoTq);

double maxSTk = TrovaMax(simulatoTk);

double maxSQ = TrovaMax(simulatoQ);

double maxSK = TrovaMax(simulatoK);

double maxTq = TrovaMax(maxSTq, maxTTq);

double maxTk = TrovaMax(maxSTk, maxTTk);

double maxQ = TrovaMax(maxSQ, maxTQ);

double maxK = TrovaMax(maxSK, maxTK);

StreamWriter script = new StreamWriter(percorso + "\\script\\script.txt", false);

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Tempo medio in coda""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\tempo\_coda.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E[Tq]'");

script.WriteLine(@"set xrange[0:1]");

script.WriteLine(@"set yrange[0:" + Math.Round(maxTq + 1, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\teorici" + @"\\tempoCoda.txt' title ""Teorico Tempo Coda"" with lines lt 1 lc 1, '" + percorso + "\\simulati" + @"\\tempoCoda.txt' title ""Simulato Tempo Coda"" with lines lt 2 lc 2");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Tempo medio nel sistema""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\tempo\_sistema.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E[Tk]'");

script.WriteLine(@"set xrange[0:1]");

script.WriteLine(@"set yrange[0:" + Math.Round(maxTk + 1, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\teorici" + @"\\tempoSistema.txt' title ""Teorico Tempo Sistema"" with lines lt 1 lc 1, '" + percorso + "\\simulati" + @"\\tempoSistema.txt' title ""Simulato Tempo Sistema"" with lines lt 2 lc 2");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Utenti medi in coda""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\utenti\_coda.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E[q]'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxQ + 1, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\teorici" + @"\\utentiCoda.txt' title ""Teorico Utenti Coda"" with lines lt 1 lc 1, '" + percorso + "\\simulati" + @"\\utentiCoda.txt' title ""Simulato Utenti Coda"" with lines lt 2 lc 2");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Utenti medi nel sistema""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\utenti\_sistema.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E[k]'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxK + 1, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\teorici" + @"\\utentiSistema.txt' title ""Teorico Utenti Sistema"" with lines lt 1 lc 1, '" + percorso + "\\simulati" + @"\\utentiSistema.txt' title ""Simulato Utenti Sistema"" with lines lt 2 lc 2");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Errore % Utenti medi nel sistema""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\errore\_utenti\_sistema.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E %'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxEK + 5, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\errore" + @"\\utentiSistema.txt' title ""Errore percentuale Utenti Sistema"" with lines lt 1 lc 1");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Errore % Utenti medi in coda""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\errore\_utenti\_coda.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E %'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxEQ + 5, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\errore" + @"\\utentiCoda.txt' title ""Errore percentuale Utenti Coda"" with lines lt 1 lc 1");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Errore % Tempo medio nel sistema""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\errore\_tempo\_sistema.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E %'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxETk + 5, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\errore" + @"\\tempoSistema.txt' title ""Errore percentuale Tempo Sistema"" with lines lt 1 lc 1");

script.WriteLine(@"unset multiplot");

script.WriteLine(@"set terminal jpeg");

script.WriteLine(@"set title ""Errore % Tempo medio in coda""");

script.WriteLine(@"set output '" + percorso + "\\grafici" + "\\errore\_tempo\_coda.jpeg'");

script.WriteLine(@"set key left box");

script.WriteLine(@"set multiplot");

script.WriteLine(@"set xlabel 'RHO fattore di utilizzazione'");

script.WriteLine(@"set ylabel 'E %'");

script.WriteLine(@"set xrange[0:1] ");

script.WriteLine(@"set yrange[0:" + Math.Round(maxETq + 5, 0).ToString() + "]");

script.WriteLine(@"set style data lines");

script.WriteLine(@"plot '" + percorso + "\\errore" + @"\\tempoCoda.txt' title ""Errore percentuale Tempo Coda"" with lines lt 1 lc 1");

script.WriteLine(@"unset multiplot");

script.Close();

}

public static void StampaErrore(string percorso, List<double> erroreK, List<double> erroreQ, List<double> erroreTq, List<double> erroreTk, List<double> rho)

{

StreamWriter fErroretempoCoda = new StreamWriter(percorso + "\\errore\\tempoCoda.txt", false);

StreamWriter fErroretempoSistema = new StreamWriter(percorso + "\\errore\\tempoSistema.txt", false);

StreamWriter fErroreUtentiCoda = new StreamWriter(percorso + "\\errore\\utentiCoda.txt", false);

StreamWriter fErroreUtentiSistema = new StreamWriter(percorso + "\\errore\\utentiSistema.txt", false);

for (int i = 0; i < rho.Count; i++)

{

fErroretempoCoda.WriteLine("{0} {1}", rho[i].ToString().Replace(',', '.'), erroreTq[i].ToString().Replace(',', '.'));

fErroretempoSistema.WriteLine("{0} {1}", rho[i].ToString().Replace(',', '.'), erroreTk[i].ToString().Replace(',', '.'));

fErroreUtentiCoda.WriteLine("{0} {1}", rho[i].ToString().Replace(',', '.'), erroreQ[i].ToString().Replace(',', '.'));

fErroreUtentiSistema.WriteLine("{0} {1}", rho[i].ToString().Replace(',', '.'), erroreK[i].ToString().Replace(',', '.'));

}

fErroretempoCoda.Close();

fErroretempoSistema.Close();

fErroreUtentiCoda.Close();

fErroreUtentiSistema.Close();

}

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

buttonSimula.Enabled = false;

}

private void textBoxUtenti\_Leave(object sender, EventArgs e)

{

if(textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void textBoxCoda\_Leave(object sender, EventArgs e)

{

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void buttonOutput\_Click(object sender, EventArgs e)

{

folderBrowserDialogFileOutput.ShowDialog();

percorso = folderBrowserDialogFileOutput.SelectedPath;

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void CreaGrafici()

{

string Pgm = @"C:\Program Files (x86)\gnuplot\bin\gnuplot.exe";

Process extPro = new Process();

extPro.StartInfo.FileName = Pgm;

extPro.StartInfo.UseShellExecute = false;

extPro.StartInfo.RedirectStandardInput = true;

extPro.Start();

StreamWriter gnupStWr = extPro.StandardInput;

string path = percorso + "\\script\\script.txt";

gnupStWr.WriteLine("load '" + path + "'");

gnupStWr.Flush();

gnupStWr.WriteLine("quit");

gnupStWr.Flush();

extPro.Close();

}

private void buttonSimula\_Click(object sender, EventArgs e)

{

//disabilito il form

textBoxRho.Enabled = false;

textBoxCoda.Enabled = false;

textBoxUtenti.Enabled = false;

buttonOutput.Enabled = false;

buttonReset.Enabled = false;

buttonSimula.Enabled = false;

//imposto i parametri

NUTENTI = Convert.ToInt32(textBoxUtenti.Text);

Y = Convert.ToInt32(textBoxCoda.Text);

percorso = folderBrowserDialogFileOutput.SelectedPath;

INCR\_LAMBDA = Convert.ToDouble(textBoxRho.Text.Replace('.',','));

percorso = CreaCartelle(percorso);

//inizio simulazione

List<double> tempoSistemaMedio = new List<double>();

List<double> tempoCodaMedio = new List<double>();

List<double> utentiSistemaMedio = new List<double>();

List<double> utentiCodaMedio = new List<double>();

List<double> tempoServizioMedio = new List<double>();

List<double> r = new List<double>();

List<double> TtempoSistemaMedio = new List<double>();

List<double> TtempoCodaMedio = new List<double>();

List<double> TutentiSistemaMedio = new List<double>();

List<double> TutentiCodaMedio = new List<double>();

List<double> TtempoServizioMedio = new List<double>();

DateTime inizio = DateTime.Now;

double tqTeorico, tkTeorico, kTeorico, qTeorico, txTeorico;

for (lambda = 0.01; lambda <= 1; lambda += INCR\_LAMBDA)

{

inizializzaSimulazione();

simulazione();

if( NUTENTI <= Y)

teoricoMM1(out kTeorico, out qTeorico, out tqTeorico, out tkTeorico, out txTeorico);

else

teoricoMM1Y(out kTeorico, out qTeorico, out tqTeorico, out tkTeorico, out txTeorico);

r.Add(Math.Round(lambda / (double)MU, 2));

TtempoCodaMedio.Add(Math.Round(tqTeorico, 15));

TtempoSistemaMedio.Add(Math.Round(tkTeorico, 15));

TutentiCodaMedio.Add(Math.Round(qTeorico, 15));

TutentiSistemaMedio.Add(Math.Round(kTeorico, 15));

TtempoServizioMedio.Add(Math.Round(txTeorico, 15));

//if (i == 80)

// Console.ReadLine();

tempoCodaMedio.Add(Math.Round(MU \* accumula\_tq / (double)nUtentiGenerati, 15));

tempoSistemaMedio.Add(Math.Round(MU \* (accumula\_tx + accumula\_tq) / (double)nUtentiGenerati, 15));

utentiCodaMedio.Add(Math.Round(accumula\_q / (double)nUtentiGenerati, 15));

utentiSistemaMedio.Add(Math.Round(accumula\_k / (double)nUtentiGenerati, 15));

tempoServizioMedio.Add(Math.Round(accumula\_tx / (double)nUtentiGenerati, 15));

if(lambda == 0)

progressBarAvanzamentoSImulazione.Value = Convert.ToInt32(1);

else

progressBarAvanzamentoSImulazione.Value = Convert.ToInt32(lambda \* 100);

}

List<double> erroreK, erroreQ, erroreTk, erroreTq;

erroreK = new List<double>();

erroreQ = new List<double>();

erroreTk = new List<double>();

erroreTq = new List<double>();

CalcolaErrore(tempoSistemaMedio, tempoCodaMedio, utentiSistemaMedio, utentiCodaMedio, TtempoSistemaMedio, TtempoCodaMedio, TutentiSistemaMedio, TutentiCodaMedio, out erroreTk, out erroreTq, out erroreK, out erroreQ);

StampaErrore(percorso, erroreK, erroreQ, erroreTq, erroreTk, r);

stampaRisultatiSimulati(percorso, r, tempoCodaMedio, tempoSistemaMedio, utentiSistemaMedio, utentiCodaMedio, tempoServizioMedio);

stampaTeorici(percorso, r, TtempoCodaMedio, TtempoSistemaMedio, TutentiSistemaMedio, TutentiCodaMedio, TtempoServizioMedio);

CreaScript(erroreTq, erroreTk, erroreK, erroreQ, TtempoCodaMedio, TtempoSistemaMedio, TutentiCodaMedio, TutentiSistemaMedio, tempoSistemaMedio, tempoCodaMedio, utentiSistemaMedio, utentiCodaMedio);

CreaGrafici();

DateTime fine = DateTime.Now;

TimeSpan durataSimulazione = fine.Subtract(inizio);

labelTempoSimulazione.Text = "Simulazione terminata! Tempo impiegato: " + durataSimulazione.ToString().Substring(0, 11);

//riabilito il form

textBoxRho.Enabled = true;

textBoxCoda.Enabled = true;

textBoxUtenti.Enabled = true;

buttonOutput.Enabled = true;

buttonReset.Enabled = true;

buttonSimula.Enabled = false;

folderBrowserDialogFileOutput.SelectedPath = "";

}

private void buttonReset\_Click(object sender, EventArgs e)

{

buttonSimula.Enabled = false;

percorso = "";

textBoxCoda.Text = "";

textBoxUtenti.Text = "";

progressBarAvanzamentoSImulazione.Value = 0;

labelTempoSimulazione.Text = "";

textBoxRho.Text = "";

}

private void textBoxRho\_Leave(object sender, EventArgs e)

{

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void textBoxUtenti\_TextChanged(object sender, EventArgs e)

{

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void textBoxCoda\_TextChanged(object sender, EventArgs e)

{

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

private void textBoxRho\_TextChanged(object sender, EventArgs e)

{

if (textBoxCoda.Text != "" && textBoxUtenti.Text != "" && percorso != "" && textBoxRho.Text != "")

buttonSimula.Enabled = true;

else

buttonSimula.Enabled = false;

}

}

public class Utente

{

double tempoServizio;

double tempoNelSistema;

double tempoInCoda;

//tempo rimanente prima dell'esaurimento del tempo di servizio assegnato

double tempoServizioRimanente;

public double TempoServizio

{

get

{

return tempoServizio;

}

set

{

tempoServizio = value;

}

}

public double TempoInCoda

{

get

{

return tempoInCoda;

}

set

{

tempoInCoda = value;

}

}

public double TempoNelSistema

{

get

{

return tempoNelSistema;

}

set

{

tempoNelSistema = value;

}

}

public double TempoServizioRimanente

{

get

{

return tempoServizioRimanente;

}

set

{

tempoServizioRimanente = value;

}

}

}

}