Descrizione del sistema

Il sistema Prodigit è un sistema di prenotazione per i posti in aula dell'università "La Sapienza" di Roma. Il sistema permette ad uno studente di effettuare una prenotazione per una determinata aula e una specifica fascia oraria, e di poter cancellare le prenotazioni effettuate. Nel sistema sono presenti N aule, ogni aula ha una capienza massima e può essere agibile o inagibile. Affinché una prenotazione possa essere effettuata con successo, il numero di posti disponibili nell'aula per la fascia oraria per cui ci si vuole prenotare deve essere superiore alla capienza dell'aula stessa. Inoltre, l'aula deve essere agibile e lo studente non deve aver già effettuato altre prenotazioni per quella fascia oraria. Il sistema infatti prevede M time slots in cui uno studente può prenotare un'aula. Per ottenere le informazioni sulle aule, il sistema Prodigit utilizza il sistema GOMP che può essere up o down, quindi se è up il GOMP svolge il ruolo di interfaccia e mostra allo studente lo stato agibile/inagibile dell'aula. Una prenotazione può essere effettuata da uno studente solo se in quel momento il GOMP è online. Nel caso in cui invece il GOMP risulti offline per un periodo di tempo, Prodigit deve garantire che non tutte le prenotazioni che arrivano in quel lasso temporale vadano perse. Il sistema Prodigit deve poter essere manutenibile, tuttavia le funzionalità del sistema non devono essere interrotte per un intervallo di tempo troppo ampio.

Scenari operativi

(a) L'environmnet consiste dei seguenti elementi il cui comportamento non è controllabile dal nostro software:

• Aule: ad intervalli regolari un'aula può cambiare stato e passare da agibile ad inagibile o viceversa. Il sistema Prodigit acquisisce tale informazione solo se il GOMP è online;
• Studenti: uno studente può fare nulla, prenotare con il suo ID l'aula k per il time slot j e cancellare la sua prenotazione con il suo ID per l'aula k al time slot j.

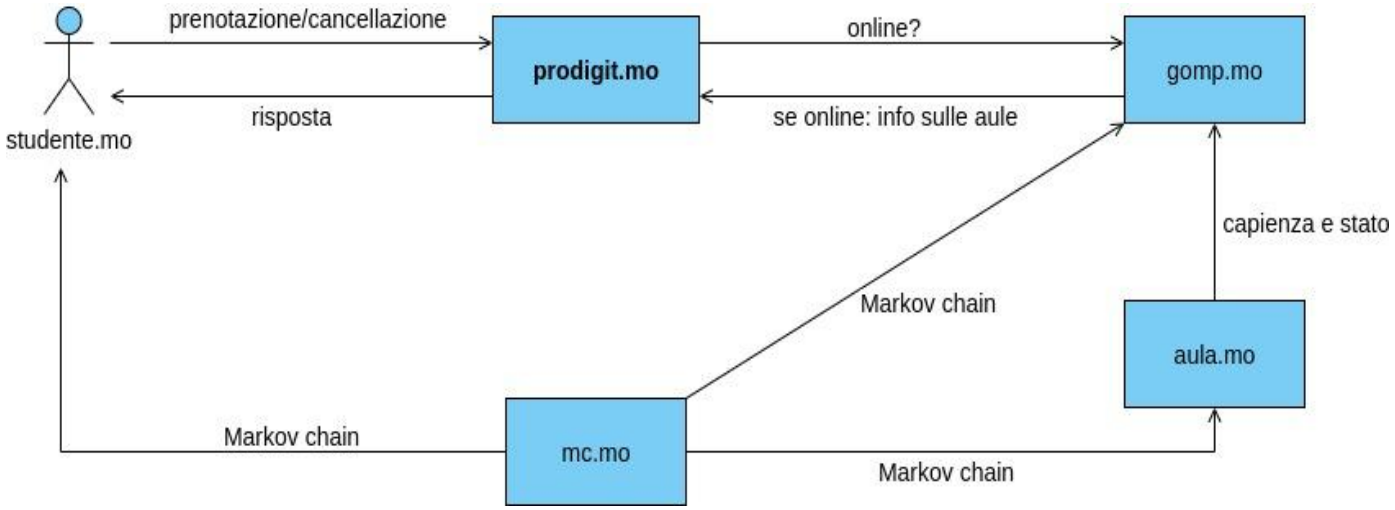
(b) L'aula è modellata come un block che ha come parameter una capienza e un periodo di sampling. Prende come input un intero che identifica lo stato in cui deve transitare e dà in output lo stato corrente (che può essere agibile o inagibile) e la capienza. L'intero preso in input (che controlla la transizione da agibile a inagibile) proviene da una Markov Chain.

Lo studente è modellato come un block, ha come parameter un periodo di sampling, prende in input un intero che controlla la transizione tra i suoi tre possibili stati (fare nulla, vuole prenotare, vuole cancellare) e restituisce in output il suo stato. Anche qui la transizione tra stati è controllata da una Markov Chain.

(c) I modelli Modelica per gli scenari operativi possono essere considerati completi perché prendono in considerazione tutto ciò di cui un sistema di prenotazione di posti in aula universitario necessita. Infatti gli unici attori coinvolti nella prenotazione sono gli studenti e le aule (con la mediazione del GOMP) e si tiene conto di tutti i loro possibili stati.

(d) I modelli Modelica per gli scenari possono essere considerati plausibili perché gli scenari presi in considerazione sono fedeli alla realtà e viene modellato solo ciò che è opportuno e naturale che un sistema di prenotazione di posto in aula universitario abbia.

Architettura del sistema



Il sistema di prenotazione è modellato all'interno del block Prodigit. Quest'ultimo prende dagli altri modelli, con cui interagisce, tutte le informazioni che gli sono necessarie per controllare se una richiesta può essere correttamente processata o no. In particolare prende in input lo stato del gomp, lo stato dello studente, la capienza di ogni aula e lo stato dell'aula. Inoltre ha come parameter il numero di studenti che possono fare richieste al sistema, il numero di aule che sono a disposizione e il numero di fasce orarie per cui ci si può prenotare per le aule. Le prenotazioni effettuate dagli studenti sono modellate tramite un array bidimensionale con un numero di righe pari al numero degli studenti e un numero di colonne pari al numero di fasce orarie. Nella cella [i,j] ci sarà l'identificativo dell'aula per cui lo studente con ID i si è prenotato per la fascia oraria j. Si tiene traccia anche dei posti occupati per ogni fascia oraria di ogni aula e, nello specifico, si usa anche in questo caso un array bidimensionale con un numero di righe pari al numero di aule e un numero di colonne pari al numero di fasce. Nella cella [i,j] ci sarà il numero di prenotati per l'aula con ID i nella fascia oraria j. Queste due strutture dati vengono restituite in output poiché servono ai monitor per verificare che tutti i requisiti funzionali siano soddisfatti.
Gli altri output di Prodigt sono: il numero di prenotazioni che arrivano mentre il GOMP è down e quelle che si riescono a gestire mediante una waiting list, necessarie per la verifica del primo requisito non funzionale; il numero di richieste totali che arrivano durante la simulazione e quelle che il sistema riesce a soddisfare, che servono per monitorare il secondo requisito non funzionale.
La waiting list (o coda) è stata implementata in modo analogo alla struttura dati che serve per tenere traccia delle prenotazioni degli studenti, e permette a Prodigit di ristabilire una situazione coerente quando il GOMP torna ad essere online. La coda permette di salvare soltanto la prima prenotazione dello studente x per la fascia oraria y che arriva mentre il GOMP è down, e appena il GOMP torna up la coda sarà svuotata.
Nel file connectors.mo sono dichiarati i connectors. Nel file types.mo sono presenti le varie enumeration. Nel file mc.mo sono presenti le matrici che contengono le probabilità di transitare da uno stato all'altro per il GOMP, per lo studente e per l'aula. Nel file system.mo vengono effettuate tutte le connessioni necessarie tra i vari elementi.

Requisiti del sistema

Requisiti funzionali:

ID:	F1
Descrizione:	Il sistema deve fare in modo che per ogni aula e per ogni fascia oraria il numero di prenotazioni non superi la capienza delle aule e non sia minore di 0.

ID:	F2
Descrizione:	Il sistema deve garantire che ogni studente sia prenotato al più per un'aula in ogni fascia oraria. Questo perché uno studente non può trovarsi in due posti contemporaneamente nella stessa ora.

Requisiti non funzionali:

ID:	NF1
Descrizione:	Il sistema deve garantire che non andranno perse almeno il 25% delle prenotazioni che arrivano mentre il GOMP è down.

ID:	NF2
Descrizione:	È necessario prevedere un periodo di manutenzione di Prodigit (in cui non può prendere in carico alcuna richiesta). Il periodo deve essere il più largo possibile, ma tale da garantire che almeno il 90% delle richieste totali che arrivano al sistema siano processate.

Il requisito F1 viene rispettato facendo in modo che ogni volta che arriva una prenotazione questa viene effettuata solo se la capienza è maggiore dei posti disponibili; inoltre il numero di prenotazioni non è mai minore di 0 perché una cancellazione è consentita solo se lo studente era prenotato per quell'aula. Il file "monitorf1.mo" monitora questo requisito.

Il requisito F2 viene garantito dall'implementazione della struttura dati "prenotazioniStudenti" all'interno del file "prodigit.mo". Questo perché la struttura prevede una sola cella per ogni coppia studente-fascia oraria, in cui sarà contenuto l'ID dell'aula prenotata, oppure 0 se non c'è una prenotazione dello studente per quella fascia oraria. Il file "monitorf2.mo" monitora questo requisito.

Il requisito NF1 viene rispettato mediante la struttura dati "prenotazioniInCoda" che salva la prima prenotazione di ogni studente che arriva a partire dall'ultima volta che il GOMP è

andato down. Quando il GOMP ritorna up, tale struttura viene utilizzata per risolvere le prenotazioni rimaste in coda. Il file “monitornf1.mo” monitora questo requisito.

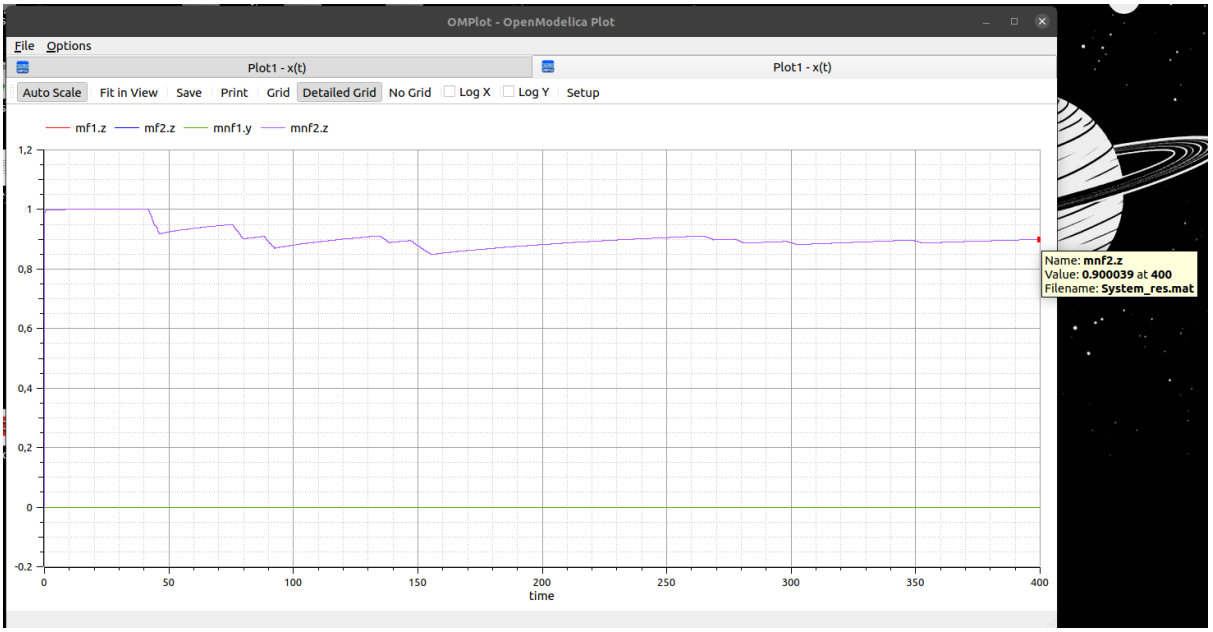
Il requisito NF2 è stato implementato trovando tramite il file “synth.py” un timestep T, che rappresenta ogni quanto Prodigit può cambiare stato. Questo per fare in modo che sia massimizzato il tempo in cui Prodigit può rimanere in manutenzione pur processando il 90% delle richieste che arrivano al sistema. Il file “monitornf2.mo” monitora questo requisito.

Risultati sperimentali

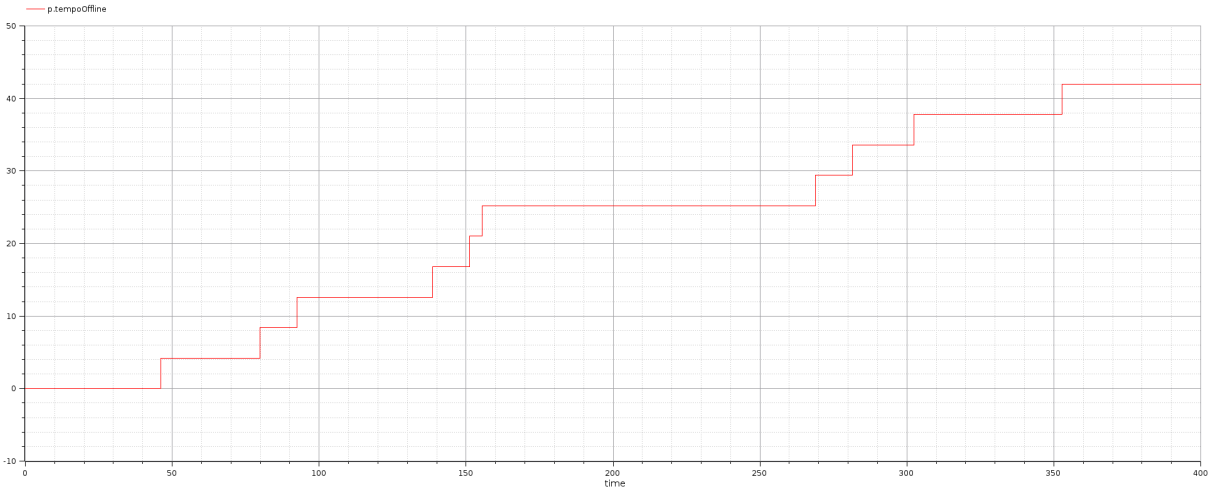
(a) Run del sistema usando lo script run.mos.
Simulazione con 120 studenti, 3 aule e 3 fasce orarie:

timeCompile = 1.412146964
timeSimulation = 0.276894495

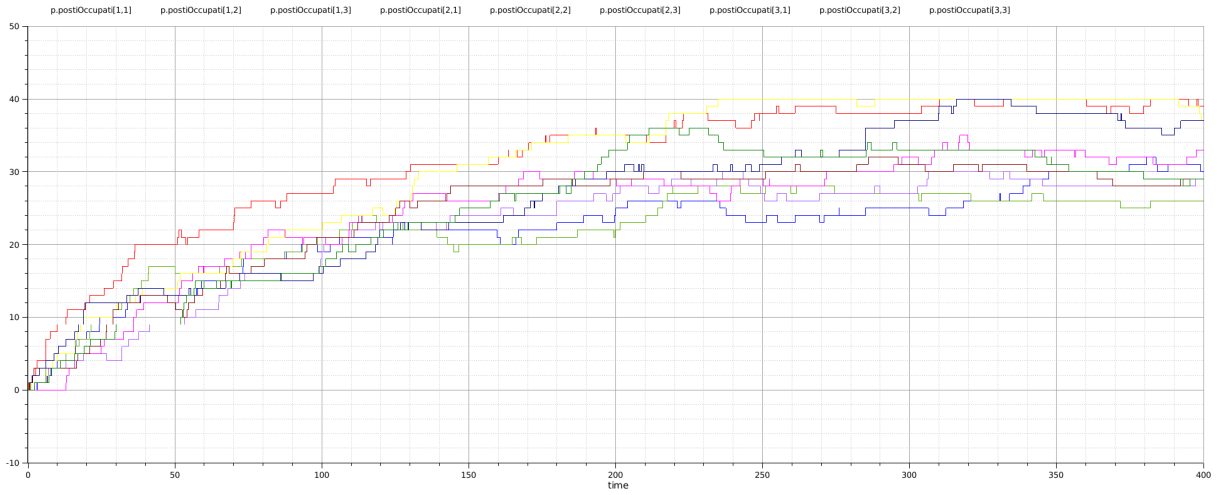
Monitors:



Tempo offline:



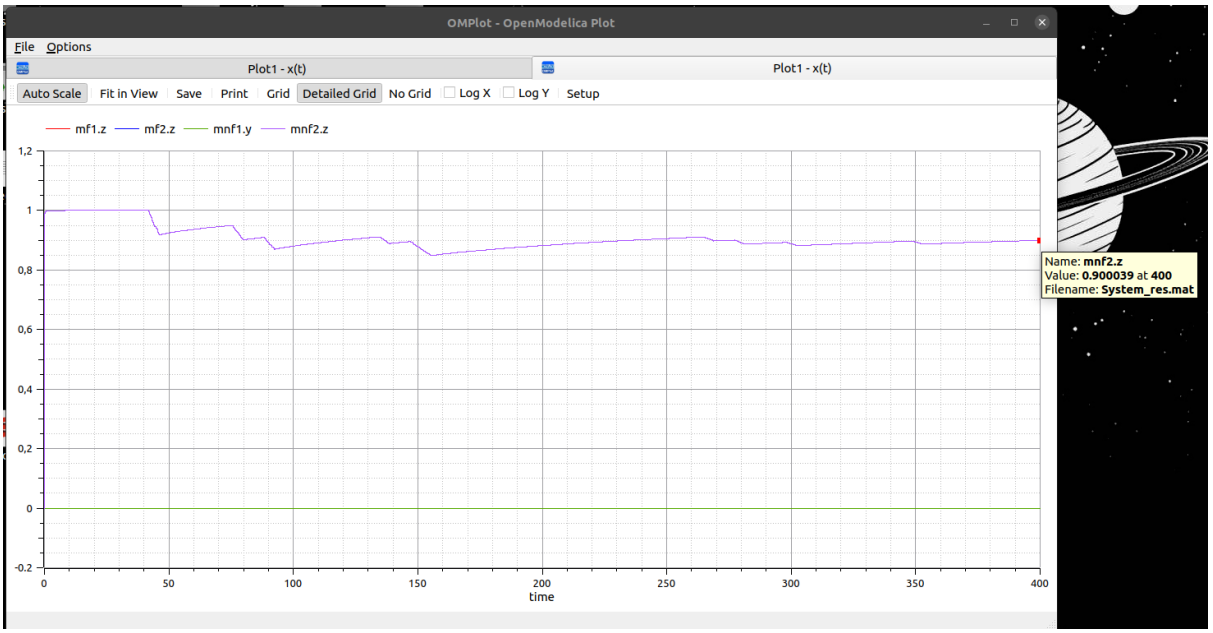
Posti occupati:



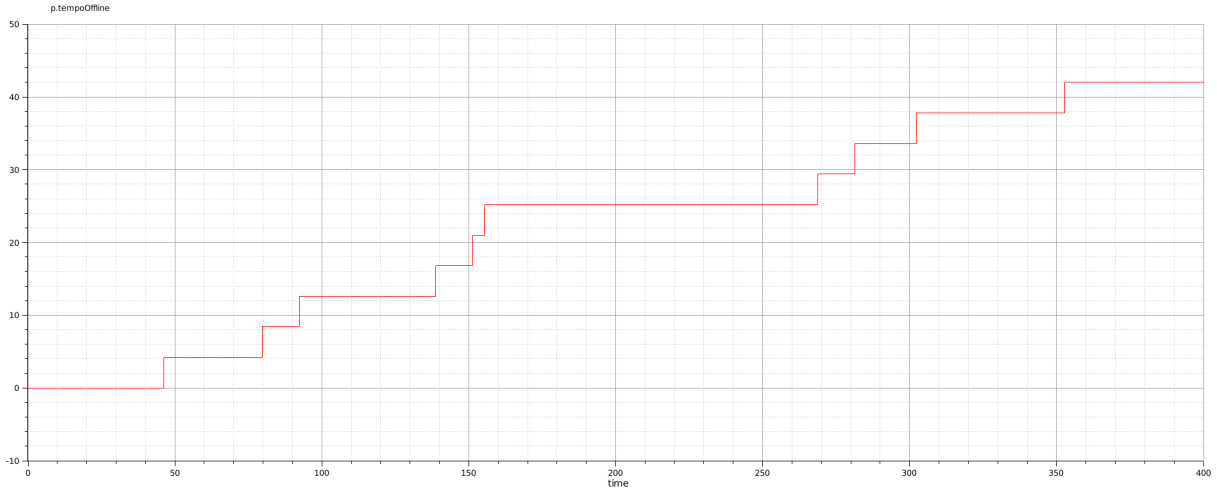
Simulazione con 200 studenti, 5 aule e 3 fasce orarie:

timeCompile = 1.984070761
timeSimulation = 0.380084077

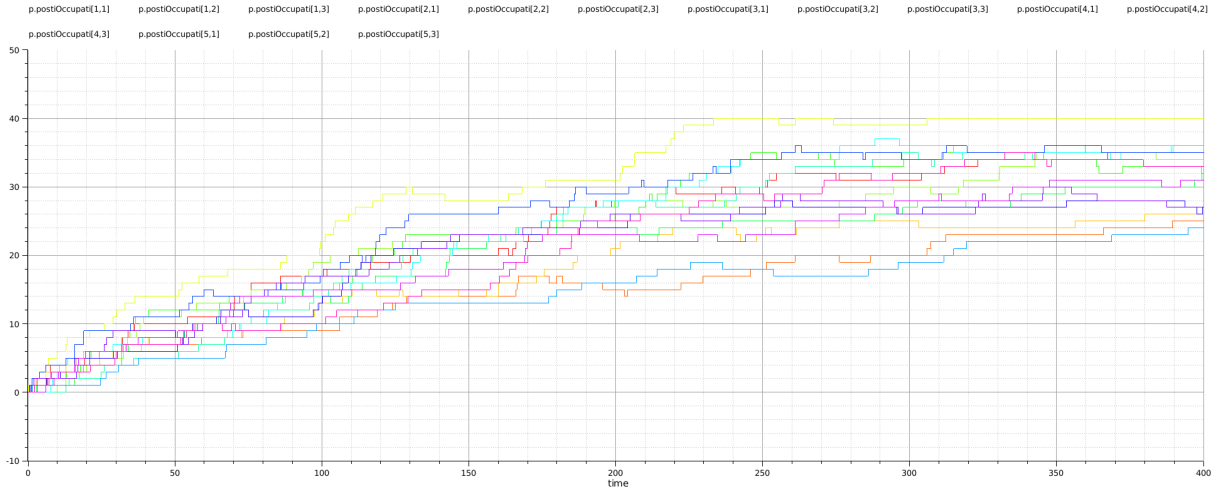
Monitors:



Tempo offline:



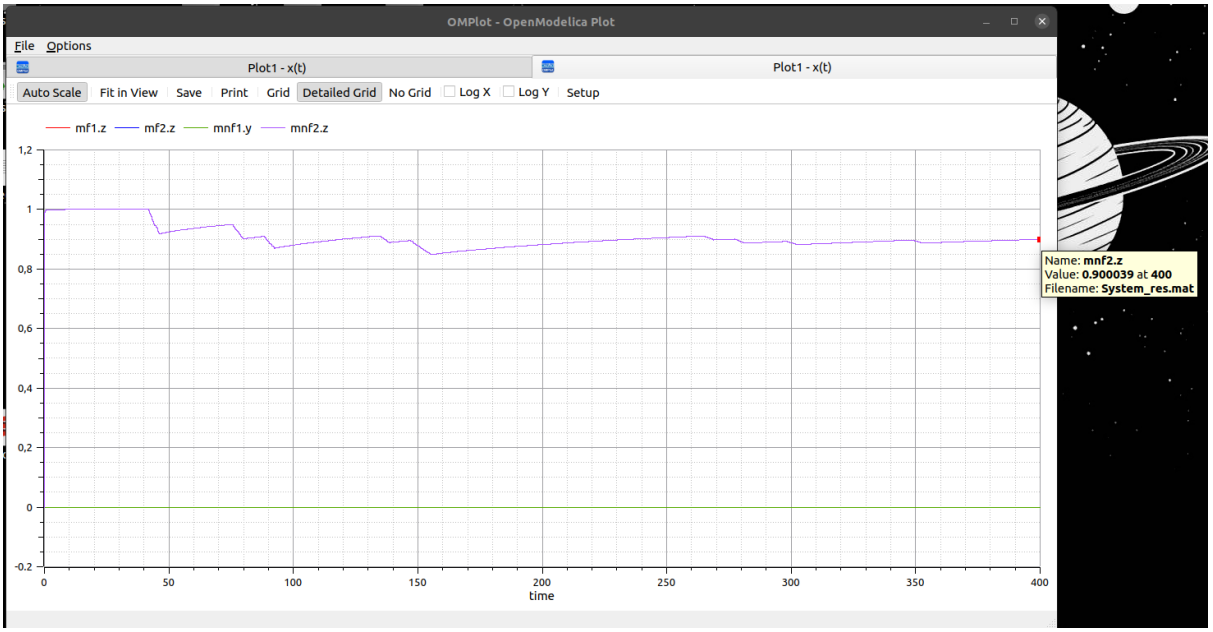
Posti occupati:



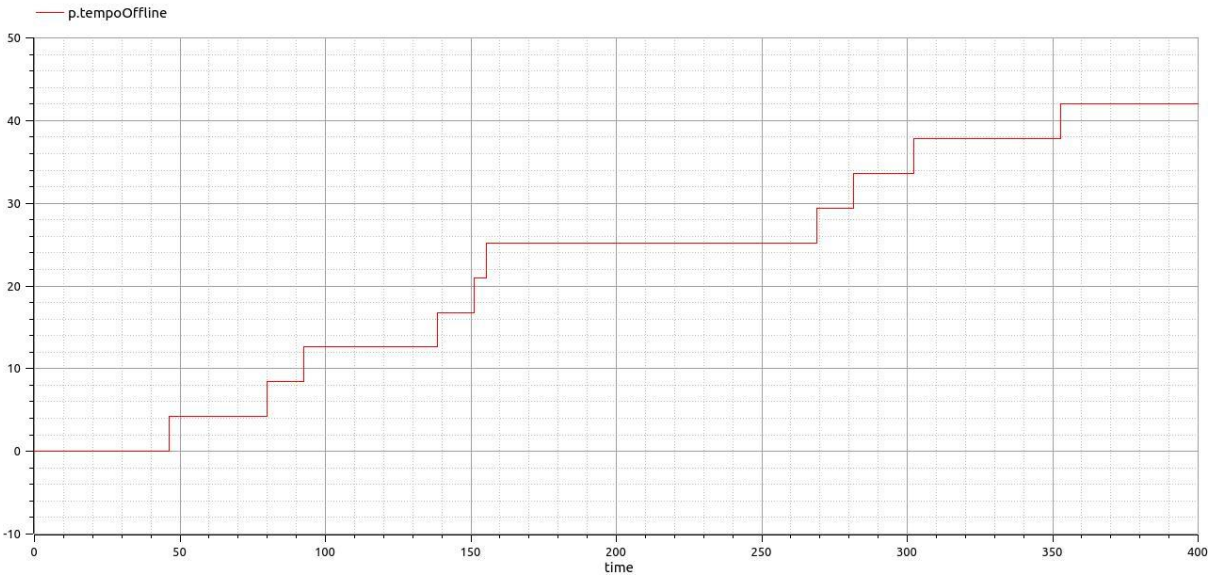
Simulazione con 500 studenti, 3 aule e 3 fasce orarie:

- timeCompile = 4.208198006
- timeSimulation = 0.961298469

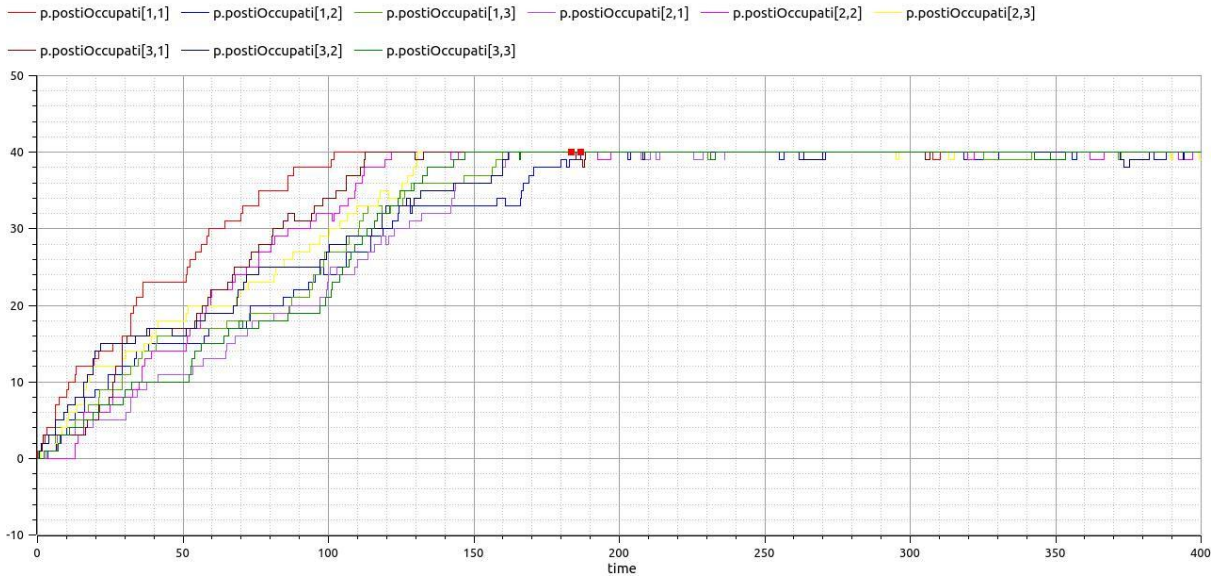
Monitors:



Tempo offline:



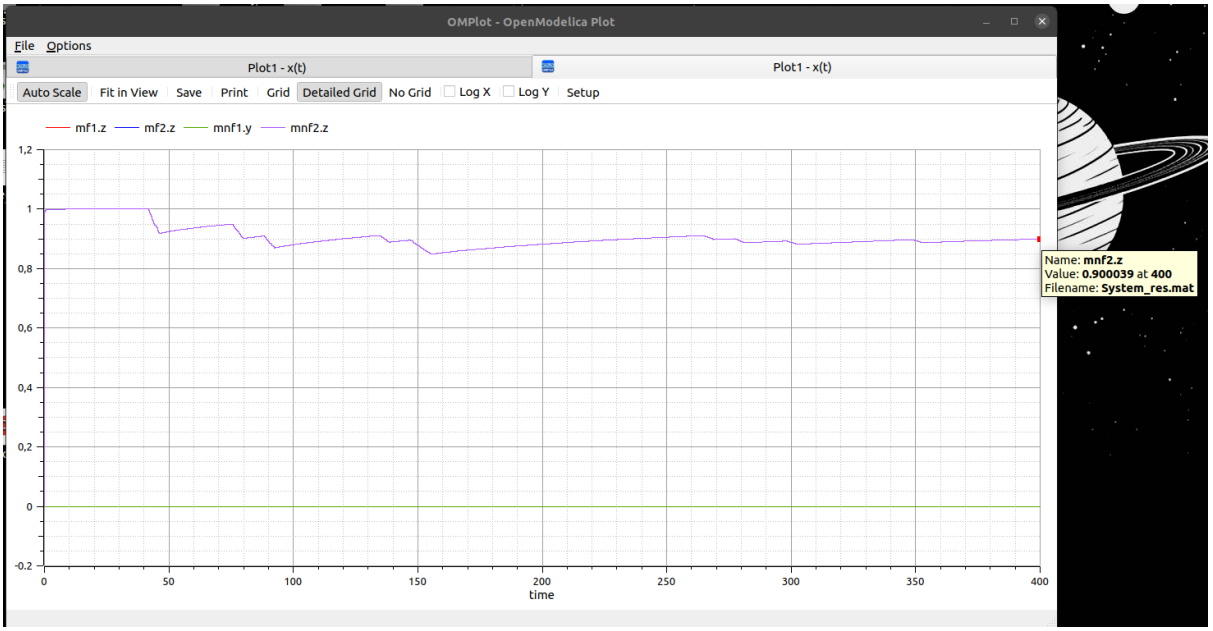
Posti occupati:



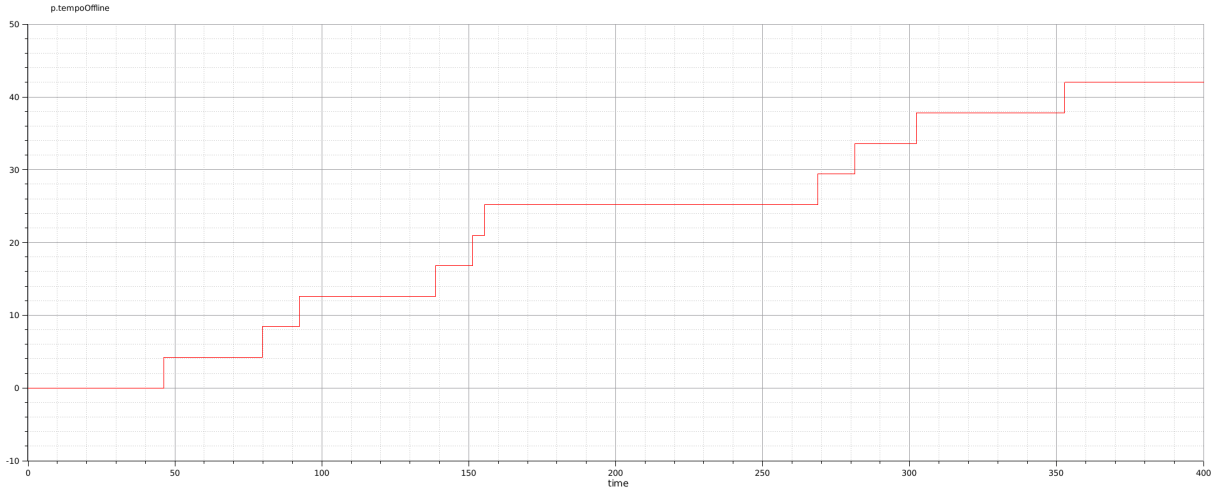
Simulazione con 500 studenti, 5 aule e 3 fasce orarie:

timeCompile = 4.410880786
timeSimulation = 0.8875326609999999

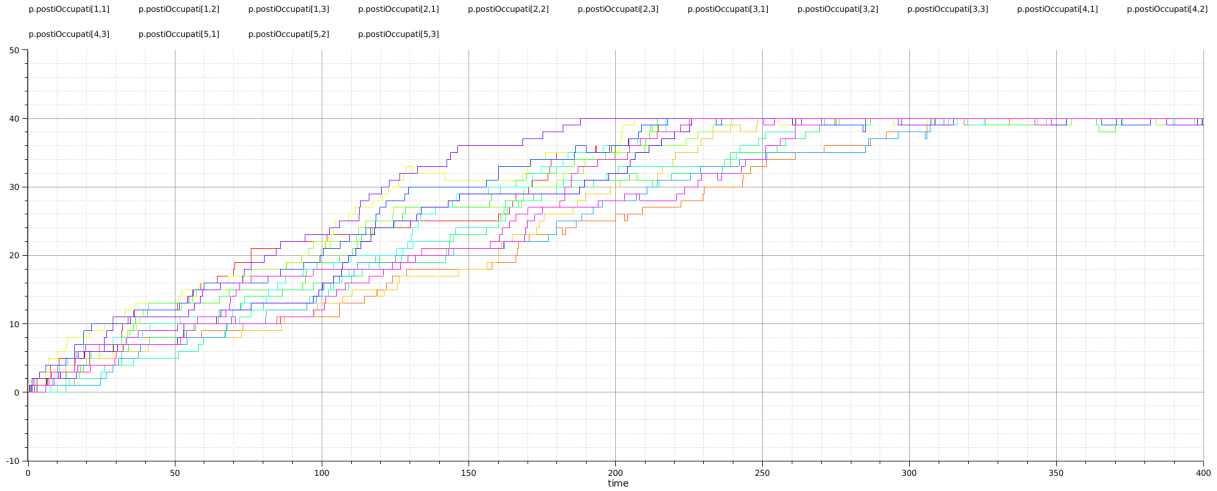
Monitors:



Tempo offline:

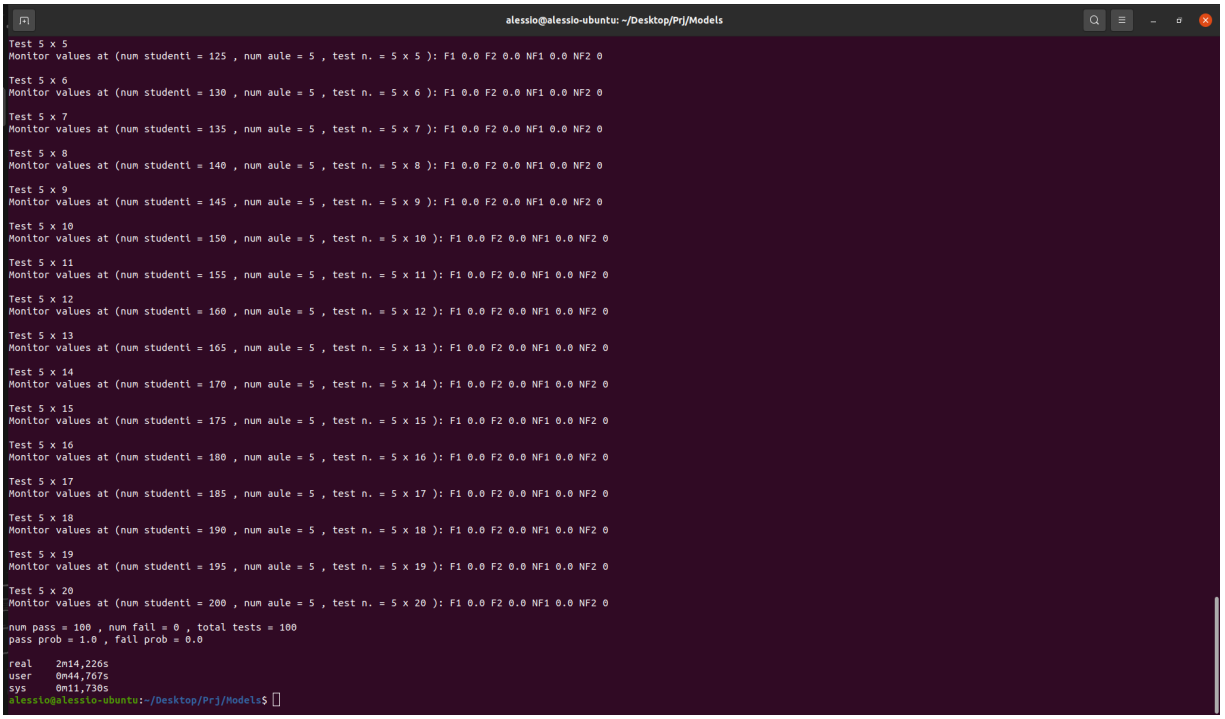


Posti occupati:

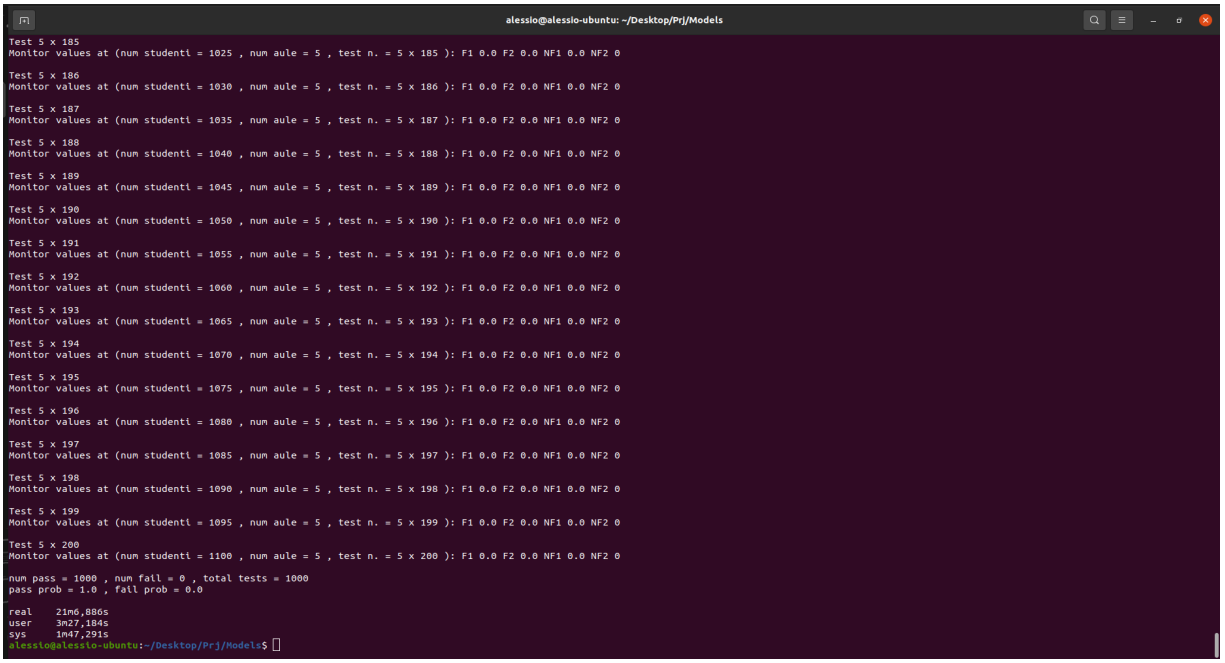


(b) Run dello script verify.py

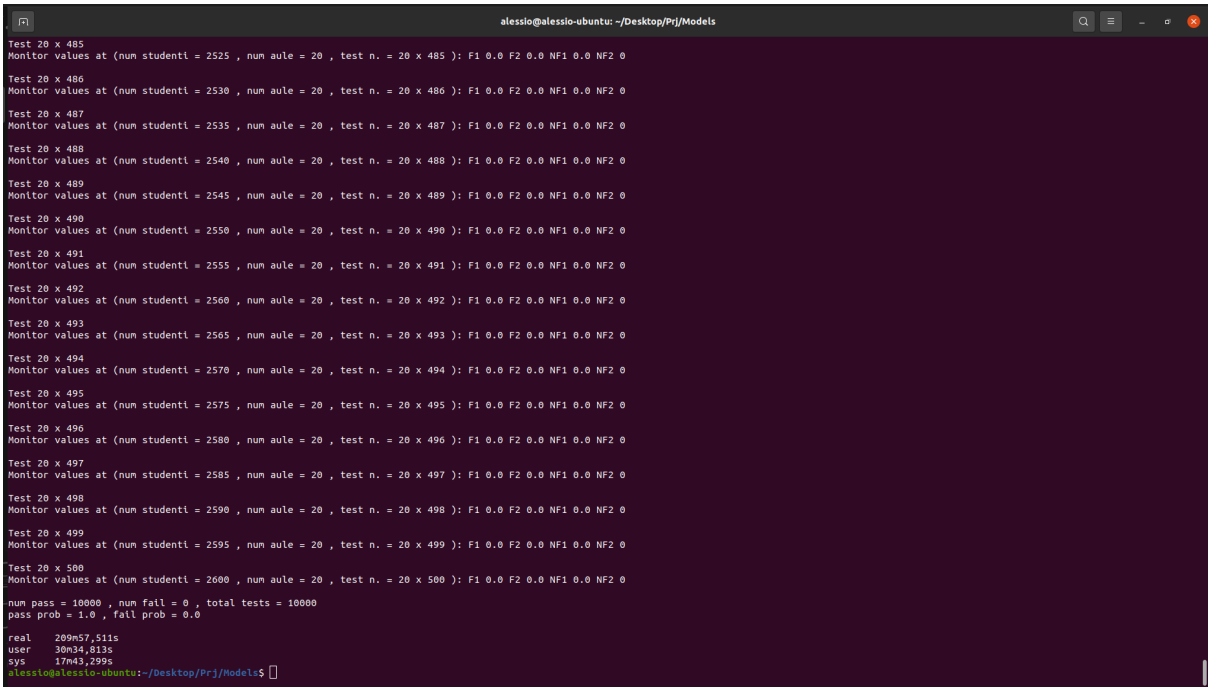
100 samples (da 1 a 5 aule, per ogni numero di aule sono stati effettuati 20 sampling di studenti):



1000 samples (da 1 a 5 aule, per ogni numero di aule sono stati effettuati 200 sampling di studenti):

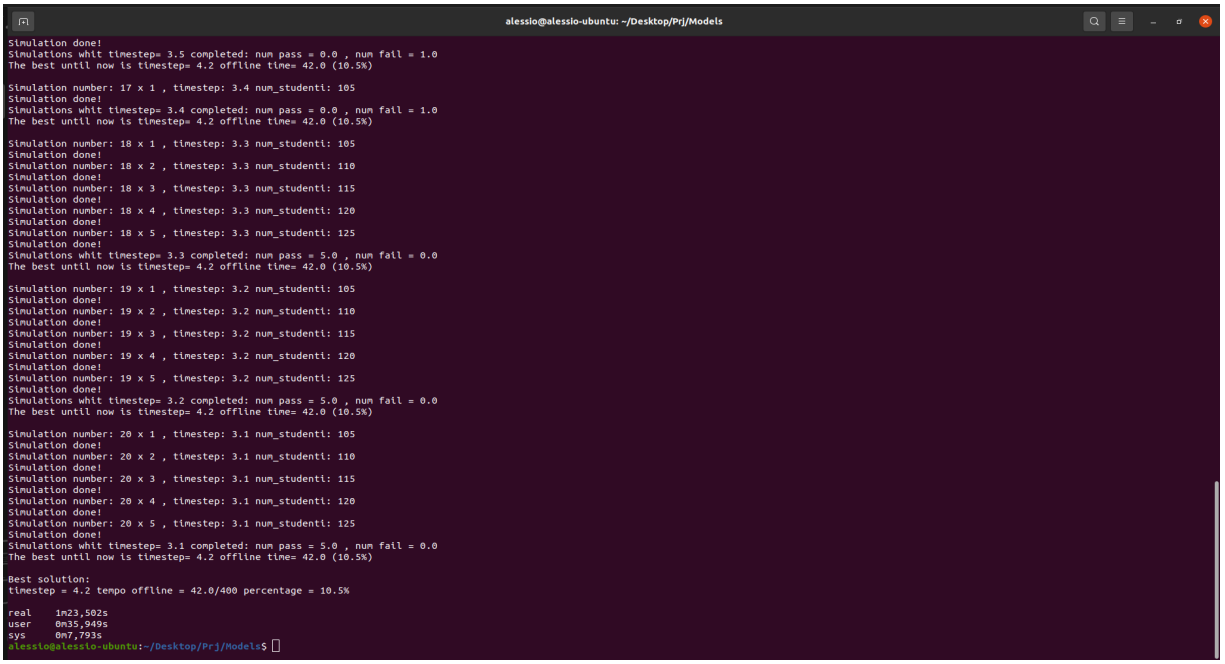


10000 samples (da 1 a 20 aule, per ogni numero di aule sono stati effettuati 500 sampling di studenti):



(c) Run dello script synth.py

100 samples, vengono testato 20 timestep T all'interno del range [3.0, 5.0], per ogni timestep si fanno 5 test con numeri di studenti diversi (se per un certo numero di studenti il test fallisce si passa immediatamente al timestep successivo):



1000 samples, vengono testato 50 timestep T all'interno del range [1.0, 6.0], per ogni timestep si fanno 20 test con numeri di studenti diversi (se per un certo numero di studenti il test fallisce si passa immediatamente al timestep successivo):