DIPARTIMENTO
DI INFORMATICA
SAPIENZA
UNIVERSITÀ DI ROMA

Progetto Ingegneria del Software

Insulin Pump

Federica Magliocca Matteo Missanelli

Indice

Descrizione del Sistema

Il progetto è stato sviluppato attraverso Modelica 1.7 e testato in Python 2.7 per simulare un gran numero di scenari diversi. Il sistema modella una pompa di insulina per pazienti iperglicemici che stabilizza la quantità di glucosio nel sangue. Il modello è quindi composto da un **Paziente** che è l'utente finale del sistema, dalla **Pompa** che rilascia insulina quando il glucosio nel sangue del paziente è troppo alto e da un **Generatore di Pasti** che simula la quantità di cibo che il paziente dovrebbe ingerire (nel nostro caso ogni 8 ore). Il sistema si occupa inoltre di controllare che i requisiti siano rispettati attraverso dei monitor. Questo sistema è un sistema safety-critical, infatti il suo obiettivo principale è assicurarsi che il glucosio nel sangue del paziente sia stabile il più possibile, altrimenti la sua salute potrebbe essere gravemente danneggiata.

Scenari Operativi

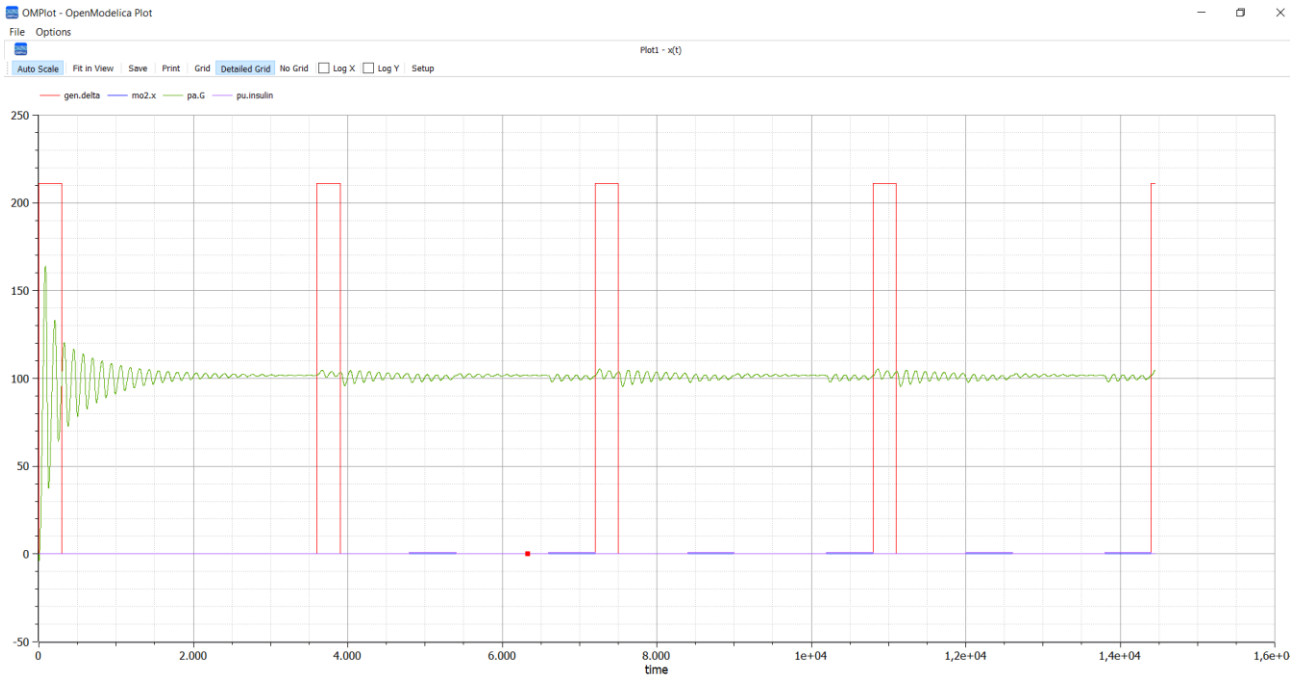
Nel nostro sistema i diversi scenari operativi sono rappresentati dai diversi pazienti che potrebbero farne uso e dai diversi pasti generati. La pompa di insulina deve essere in grado di funzionare correttamente su ogni paziente rilasciando la quantità di insulina necessaria per quel paziente, e deve far sì che il glucosio nel sangue sia stabile sia che il paziente non mangi per un periodo prolungato di tempo sia che mangi molto spesso, sia che faccia pasti molto piccoli sia che ne faccia di più grandi.

Abbiamo modellato questi scenari generando i pasti casualmente nel MealGenerator e modificando i parametri del paziente entro i range previsti dalle specifiche attraverso gli script python. Questo ci ha permesso di simulare l'azione della pompa su pazienti diversi. Attraverso i monitor abbiamo potuto constatare che il rilascio di insulina fosse corretto poiché il glucosio tende a stabilizzarsi intorno ai 100mg.

Inoltre per vedere più chiaramente come la pompa agisce per ristabilizzare la quantità di glucosio nel sangue dopo un pasto abbiamo generato con il run.mos casi estremi fuori dal range previsto.

Ovvero abbiamo modificato il periodo in cui è previsto che il pazienti mangi e aumentato considerevolmente la dimensione del pasto.

Esempio:



In questo grafico si può vedere come l'insulina agisca correttamente per stabilizzare il glucosio nel caso il paziente abbia fatto pasti molto sostanziosi molto frequentemente (abbiamo impostato il pasto a 210 ogni ora)

Architettura del Sistema

Come affermato precedentemente il sistema è composto dagli oggetti Paziente, Pompa e Generatore del pasto; sono state però aggiunte, per una maggiore leggibilità e chiarezza del codice, delle classi che rappresentano alcune funzioni richiamate dall'oggetto Paziente e una classe KPar che contiene tutti i parametri che definiscono il paziente.

Va inoltre specificato che poiché i valori del glucosio all'inizio della simulazione non sono verosimili, le classi come Pump e alcuni Monitor sono stati impostati per iniziare ad agire dopo alcuni minuti.

Vediamo più nel dettaglio l'implementazione dei vari oggetti:

-**Patient**: rappresenta l'oggetto Paziente definito nel file Patient.mo, prende in input il delta ovvero il pasto, che gli viene passato dal MealGenerator, e l'oggetto KPar che contiene tutti i suoi parametri per calcolare il glucosio nel sangue del paziente. L'informazione sul glucosio verrà poi passata alla pompa che dopo aver calcolato la dose di insulina da rilasciare la trasmetterà al Paziente che calcolerà il nuovo glucosio.

-**MealGenerator**: rappresenta il Generatore di pasti definito nel file MealGenerator.mo. Questa classe non ha input e ha un solo output, il pasto, che genererà casualmente (con valori compresi tra 10 e 30) utilizzando la libreria Modelica.Math.Random.Generators e che poi passerà al Paziente. Il pasto viene generato ogni 8 (T=28800) ore per la durata di un'ora(T = 3600) .

-Kpar: è una classe record per questo non ha né input né output ma contiene soltanto tutti i parametri del Paziente ed è definita nel file Kpar.mo.

-Pump: rappresenta il vero e proprio sistema, la pompa, definita nel file Pump.mo. Questa classe prende in input il glucosio passatogli da Paziente e calcola la dose di insulina da rilasciare tenendo conto della quantità di glucosio negli stati precedenti. L'insulina viene rilasciata ogni 10 minuti come definito nelle specifiche. La pompa inoltre calcola la quantità di insulina totale rilasciata fino a quel momento passando l'informazione al Monitor1NotFun per controllare che l'insulina iniettata sia sempre minimizzata.

-Monitor: i monitor si occupano di controllare che i requisiti del sistema siano rispettati, ne parleremo nel dettaglio più avanti.

-Funzioni: come già accennato sono state aggiunte due classi, k_empt.mo e f.mo, che rappresentano delle funzioni utilizzate dalla classe Paziente.

-System: questa classe, definita nel file System.mo, rappresenta il sistema nel suo complesso, cioè connette tutti gli oggetti del sistema collegando gli input e gli output delle varie classi.

Requisiti del Sistema

Abbiamo modellato il sistema facendo sì che rispettasse 2 requisiti funzionali e 2 requisiti non funzionali.

Requisiti funzionali:

-**Safety**: vogliamo assolutamente evitare che il paziente vada in ipoglicemia (cioè che abbia un considerevole decremento dei livelli di glucosio nel sangue). In particolare ci preoccupiamo che la quantità di glucosio nel paziente non scenda mai sotto i 50 mg. Questo requisito è modellato dalla pompa che non rilascia mai insulina se il glucosio è inferiore ai 50 mg e verificato dal monitor nel file Monitor1Fun.mo.

-**Liveness**: il nostro obbiettivo è che il glucosio sia stabile il più possibile intorno ai 100 mg. Anche in questo caso è la pompa a far sì che questo accada, utilizzando il glucosio negli stati precedenti per capire se e quanta insulina sia necessaria per stabilizzare il glucosio. Il tutto è verificato dal secondo monitor contenuto nel file Monitor2Fun.mo.

Requisiti non funzionali:

-vogliamo che l'insulina che viene somministrata al paziente in un certo periodo di tempo sia la minima necessaria. Per questo motivo è stato aggiunto nella pompa un parametro di correzione che viene moltiplicato all'insulina per testare con gli script python varie dosi di

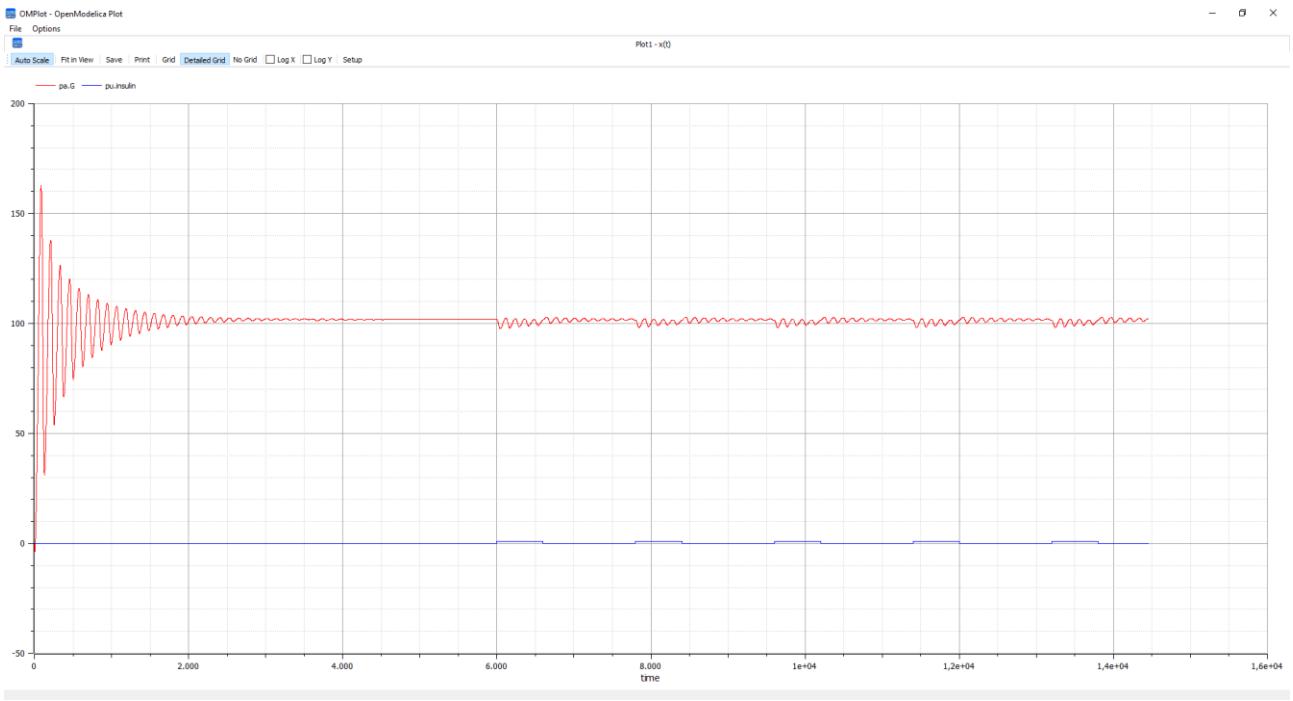
insulina sui pazienti. Una volta trovato il parametro di correzione per cui la pompa funzioni correttamente e la dose rilasciata sia minima viene trasmesso alla pompa. Inoltre la pompa calcola l'insulina totale rilasciato fino a quel momento per far si che il monitor, nel file Monitor1NonFun.mo, verifichi l'insulina rilasciata sia minimizzata.

-vogliamo far sì che il periodo in cui la pompa agisce sia massimizzato poiché dobbiamo cercare di rilasciare meno insulina possibile. Allo stesso modo abbiamo simulato con gli script python vari periodi in cui la pompa rilascia insulina fino a trovare quello massimo per cui la pompa rispetti i requisiti funzionali.

Risultati Sperimentali

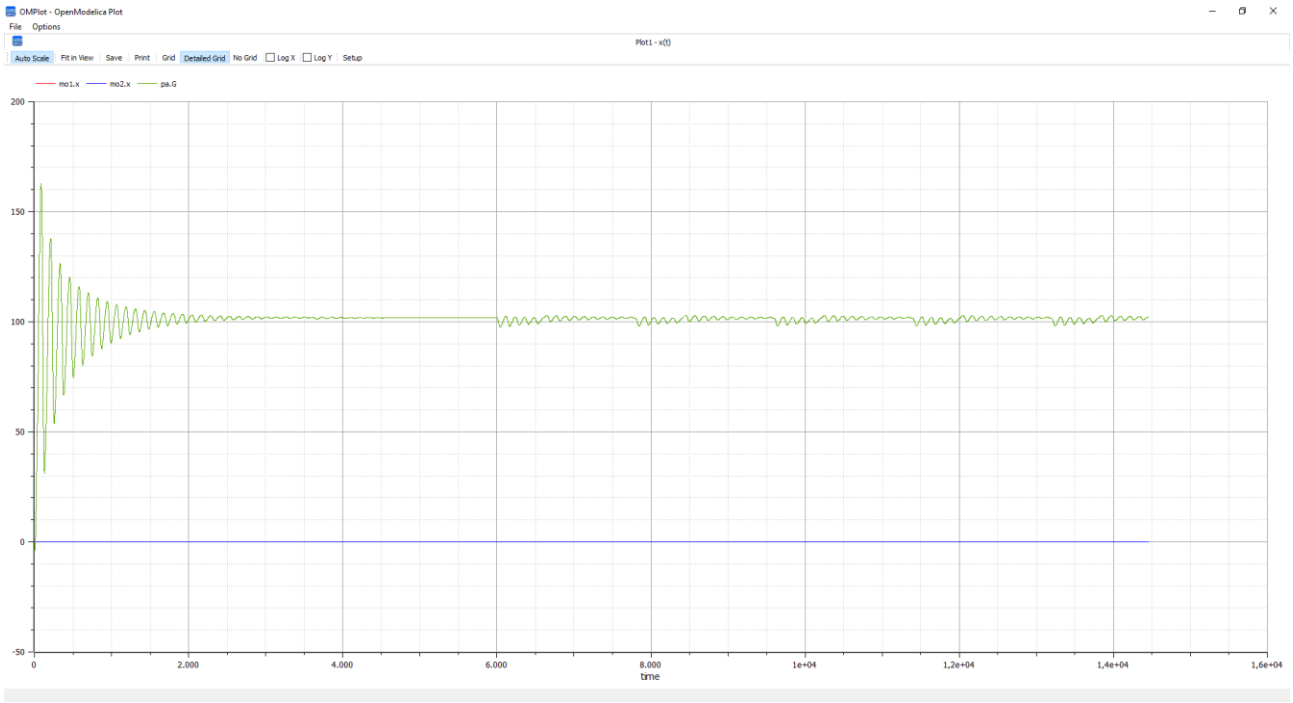
Vengono riportate diverse simulazioni eseguite con il run.mos

Simulazione1:



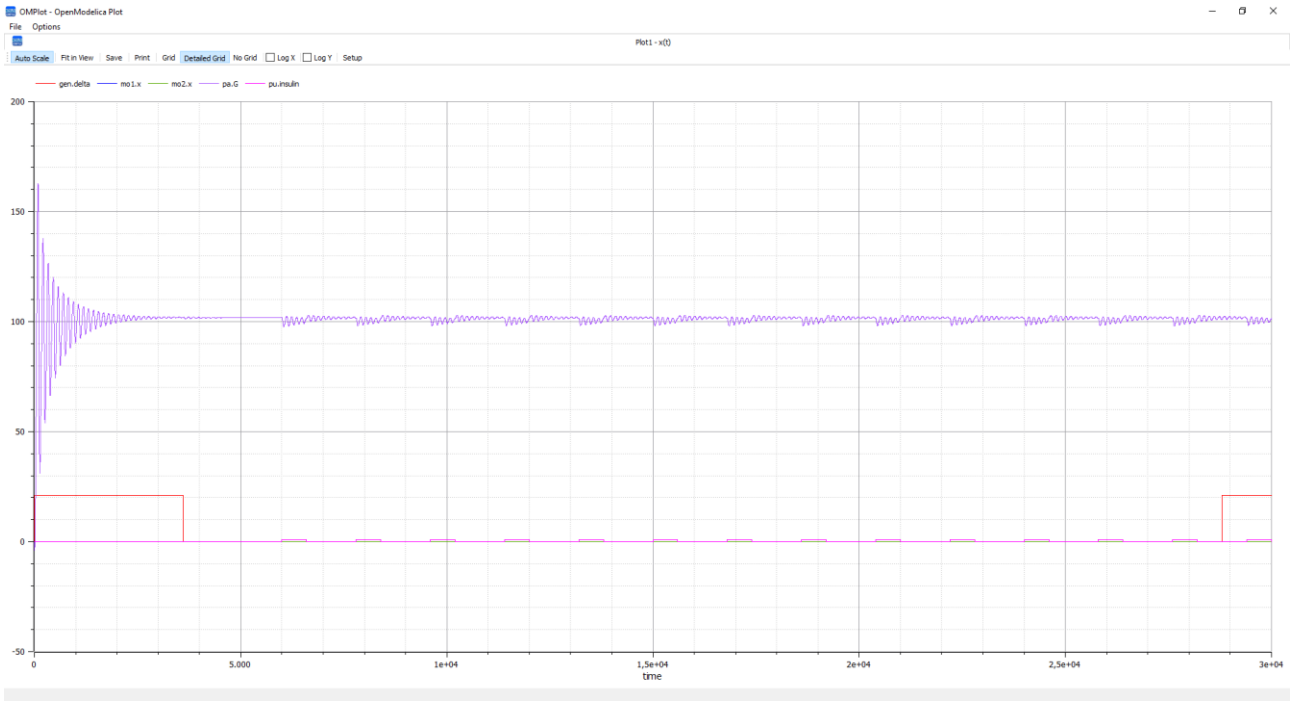
In questo grafico viene riportato l'andamento del glucosio in base all'insulina rilasciata.

Simulazione2:



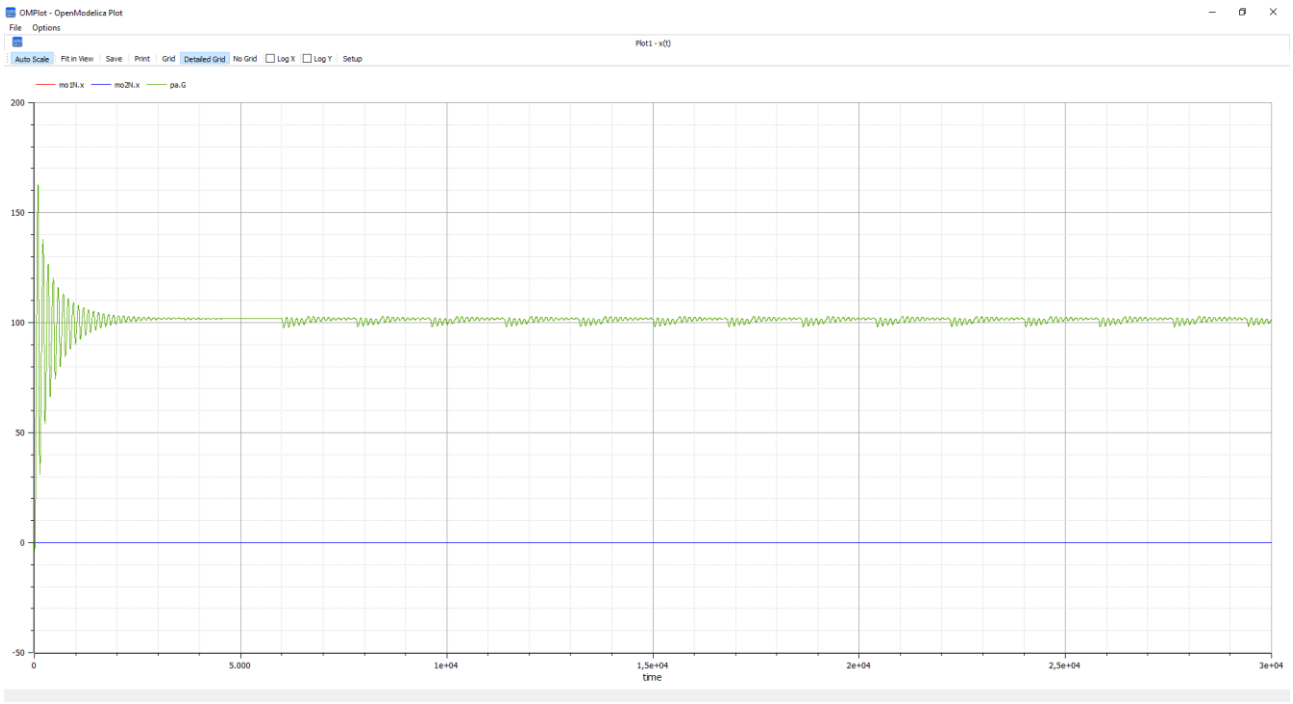
In questo grafico si può notare come i requisiti funzionali in relazione al glucosio sono sempre rispettati, poiché i due monitor hanno sempre valore zero.

Simulazione3:



Il grafico mostra tutti gli output del sistema, oltre all'andamento del glucosio si possono notare i monitor, i pasti generati e l'insulina rilasciata nell'arco di circa 8 ore.

Simulazione4:



In questo grafico si può notare come i requisiti non funzionali in relazione all'insulina rilasciata sono sempre rispettati, poiché i due monitor hanno sempre valore zero.

Simulazioni più dettagliate sono state eseguite attraverso degli script python : verify.py e synth.py

- Nel verify.py abbiamo testato il sistema su diversi pazienti modificando i parametri del file Kpar.mo per verificare che i requisiti funzionali fossero sempre rispettati. Le simulazioni sono state eseguite 100,1000 e 10000 volte e gli output sono contenuti nei rispettivi file outputFun100.txt, outputFun1000.txt, outputFun10000.txt.
- Nel synth.py abbiamo in primo luogo simulato diverse dosi di insulina rilasciata modificando i parametri della pompa per trovare l'insulina minima necessaria per ogni paziente. Allo stesso modo abbiamo simulato vari periodi in cui l'insulina viene

rilasciata modificando il parametro T della pompa per trovare il tempo massimo. Una volta ottenuti i risultati abbiamo effettuato nuove simulazioni con i dati aggiornati per verificare che i requisiti non funzionali fossero rispettati. Le simulazioni sono state eseguite 100 e 1000 volte e gli output sono contenuti nei rispettivi file outputNotFun100.txt e outputNotFun1000.txt.