Insulin Pump Control System

Model Based Software Engineering

A.Y. 2020-2021

Tiziano Sammarone, Leonardo Colosi

Indice

1 System Description

Si vuole misurare l'andamento dei livelli di glucosio di un paziente malato di diabete tipo 1, collegato ad una pompa d'insulina in costante funzionamento. Il paziente effettuerà dei pasti durante la simulazione, e la pompa si occuperà di mantenere il suo livello di glucosio entro valori accettabili tramite l'iniezione d'insulina, evitando di iniettare dosi che farebbero scendere troppo il glucosio. Verranno generati dei pazienti con valori casuali accettabili e verrà verificato il funzionamento del sistema. Ciò viene fatto dai file .py, ovvero verify.py e sinth.py.

Premesse:

- Nei file .py si è scelto di caricare il modello di sistema ad ogni simulazione, a favore delle performance. Caricando il sistema una sola volta, sembra che non fossero sincronizzate le simulazioni con i trasferimenti dei parametri (scrittura/lettura su file), quindi alcune simulazioni consecutive risultavano duplicate. Risolvendo il problema inserendo un ritardo all'interno del codice (funzione time.sleep(x)), si andava a rallentare il sistema più di quanto non costasse la ricarica dei file, senza benefici.

  - Tempo medio di simulazione ricaricando il sistema ≈ 0.83
  - Tempo medio con sleep funzionante ≥ 1

  Per ogni evenienza, si è deciso di considerare il caso in cui, magari su un computer più veloce, dia problemi anche questo metodo. Quindi, il sistema è in grado di rilevare simulazioni duplicate consecutive, e di abilitare un time.sleep(1.0). Si è scelto un valore forte in grado di garantire il funzionamento su diversi tipi di macchine, a spese delle performance. Quest'ultimo caso non si è mai verificato durante nessun test.

- Per il sistema è stata scelta come tempo di funzionamento l'unità di tempo T = 0.1 Esempio: la pompa monitora il glucosio ed inietta insulina (se opportuno) ogni 0.1 unità di tempo. In alcune componenti del sistema si è mostrato, per esigenze di efficienza, necessario misurare il tempo tramite dei cicli, che però "agivano" ogni 1.0 unità di tempo. Per mantenere la sincronizzazione tra ogni componente, sono state cambiate delle variabili o dei parametri in proporzione (anche con semplici moltiplicazioni per 10).

- Alcuni controlli possono venir eseguiti a partire da tempi superiori a quelli di inizio simulazione. Viene rafforzata la correttezza del sistema, anche perché la durata di simulazione è stata effettivamente raddoppiata. Ciò viene fatto principalmente per due motivi:

  - Il glucosio del paziente presenta un picco discendente anomalo per breve tempo all'inizio della simulazione, dopodiché il sistema si stabilizza e funziona regolarmente. Non si ripete, neanche prolungando

la simulazione. Si risolve monitorando il paziente da tempi poco successivi all'inizio simulazione. In un contesto reale, accade quando la pompa è appena installata ed il paziente controllato da personale medico, quindi non in pericolo.

– Il secondo motivo dipende dal fatto che per gestire il requisito dei pazienti ipoglicemici, è necessario avere tempo in più per applicare le correzioni, e verificare che funzionino effettivamente (vedere sezione 4, System requrements, requisiti non funzionali).

2 Operational Scenarios

Il sistema modella l'utilizzo della pompa in questo modo: ad ogni simulazione avviata tramite lo script, verrà generato un paziente, avente collegata la pompa d'insulina. Questa é costantemente in funzione. Ha il compito di monitorare il glucosio presente nel sangue del paziente e di iniettare insulina in modo da non farlo alzare eccessivamente, il che rappresenterebbe un pericolo per il paziente, avendo tuttavia cura di non iniettare troppa insulina, evitando che il glucosio scenda eccessivamente, essendo questa un'altra ipotesi pericolosa. Durante la simulazione, il paziente effettuerà dei pasti, dai quali ne ricaverà glucosio, e la pompa dovrà reagire di conseguenza.

Il modello è ragionevolmente completo e corretto, in quanto il paziente ha dei parametri interni che lo caratterizzano, e questi, ad ogni simulazione avviata tramite verify o synth, vengono scelti casualmente all'interno di intervalli definiti che li mantengono verosimili, questi intervalli sono stati ricavati dalla documentazione fornita assieme ai requisiti di progetto. Quindi, si possono generare una gran varietà di pazienti diversi grazie a queste caratteristiche, e la documentazione indica che siano corretti e plausibili grazie agli intervalli specificati.

Questo in realtà ha dato luogo a delle "anomalie", comunque correttamente gestite. Il sistema è in grado di generare pazienti il quale livello di glucosio cala a valori critici anche senza l'intervento della pompa (pazienti ipoglicemici). Da test effettuati a "pompa spenta", i pazienti non potevano sopravvivere a causa del poco glucosio presente nel loro sangue. Più nello specifico, sembra avessero un rateo di diminuzione del glucosio troppo alto, quindi, tra un pasto e l'altro, il glucosio raggiungeva picchi decrescenti, alle volte vicini allo 0. Mentre un paziente sotto i 40 comincia ad essere in pericolo di vita. Una volta confermato che ciò era completamente indipendente dalla pompa, e che questa non potesse aiutare in nessun modo questi pazienti (la pompa può solo iniettare insulina, la quale diminuisce il livello di glucosio, non lo può aumentare), ci si è trovati di fronte ad un problema. Formalmente quei pazienti erano corretti, o almeno lo erano i loro parametri (singolarmente), ma rappresentavano casi che esulavano dagli scopi di questo progetto, andando di fatto a minare la correttezza totale. Si è pensato di escluderli, perdendo in completezza e performance del sistema, ma alla fine la soluzione migliore è risultata essere il gestire questo tipo di pazienti come requisito non funzionale, permettendo alla pompa di segnalare loro che dovevano mangiare di più. Questa parte sarà spiegata più nel dettaglio nella sezione 4, System requrements, requisiti non funzionali.

3 System Architecture

Qui vengono descritte le componenti del sistema, e come interagiscono l'una con l'altra.

Componenti strutturali del modello: Queste componenti sono quelle che più rappresentano l'ambiente e le entità che vogliamo simulare.

- mealgen.mo: Determina quando e quanto il paziente mangi, e quanto glucosio fornisca il pasto. È stato integrato in questa classe parte della gestione del requisito non funzionale dei pazienti ipoglicemici (si veda sezione 4, System requrements, requisiti non funzionali, pazienti ipoglicemici). Restituisce in output a rag_meal.mo il valore nutritivo del pasto.

- rag_meal.mo: Calcola il glucosio assorbito dal pasto in base ai parametri propri del paziente. Prende in input il valore nutritivo del pasto da mealgen.mo e restituisce il valore del glucosio assorbito a fake_patient.mo.

- fake_patient.mo: Modella il paziente. Prendendo in input il valore di glucosio assorbito da rag.mo e l'insulina iniettata dalla pompa da pump.mo, si occupa di calcolare e restituire in output il su valore di glucosio.

- pump.mo: Preso in input il glucosio del paziente da fake_patient.mo, si occupa di calcolare se vada iniettata insulina, ed in quali quantità, restituendola poi come output a fake_patient.mo. In questa classe i parametri a e b contribuiscono a determinare il calcolo dell'insulina da iniettare, e dopo vari test, si è giunti a determinare valori che permettano alla pompa di funzionare con ogni paziente generabile, cercando anche di non iniettare troppa insulina in più del necessario (quest'ultimo requisito verrà calcolato più attentamente nel synth.py)

Monitors: Queste componenti monitorano ciò che accede nel modello durante la simulazione, occupandosi principalmente di segnalare l'esito della stessa.

- monitor_pump.mo: Verifica il corretto funzionamento della pompa in base ai requisiti di funzionamento da noi stabiliti. Per fare questo controllo, monitora costantemente il glucosio del paziente, preso in input da fake_patient.mo, verifica che non assuma valori fuori dall'intervallo di valori sicuri. Se il glucosio assume valori critici, una variabile di controllo cambierà valore. Tale variabile viene fornita in output.

- monitor_average.mo: Calcola i valori di glucosio minimo e massimo durante la simulazione, l'insulina totale iniettata al paziente, fondamentale per il funzionamento del synth.py, e la media del glucosio. Verifica che la media non sia troppo alta o bassa, in qual caso una delle due variabili di controllo cambia valore. Prende in input il glucosio da fake_patient.mo e l'insulina iniettata da pump.mo.

- monitor_hypogly.mo: Verifica il requisito non funzionale dei pazienti ipoglicemici (vedere sezione 4, System requrements, requisiti non funzionali). Preso in input il glucosio dal paziente, verifica durante i primi tempi della simulazione se sia ipoglicemico e se vada applicata la correzione. Se si, la variabile di output che verrà poi recepita da mealgen.mo cambierà valore.

Altro: Gli altri file rappresentano ciò che fa funzionare il modello e attiva la simulazione, quinidi system.mo e connectors.mo che consentono la comunicazione tra le componenti, ed i file py e run.mo per avviare la simulazione e verificarne i risultati.

4 System Requirements

Requisiti funzionali:

- Il primo requisito funzionale richiede che il paziente non raggiunga valori di glucosio critici, né alti né bassi. In caso di glucosio eccessivamente alto, la pompa si deve attivare per iniettare insulina e mantenerlo il più vicino possibile al valore desiderato, avendo cura di non iniettare mai insulina eccessiva che abbassi il glucosio a valori critici. Questo controllo viene effettuato dal monitor_pump.mo. Come valore desiderato abbiamo 100, come valori critici abbiamo scelto 150 e 40.

- Il secondo requisito funzionale richiede che la media di glucosio del paziente per tutta la simulazione sia attorno ai 100. Questo controllo viene effettuato dal monitor_average.mo, il quale controlla che la media non sia sotto gli 80 o sopra i 120, che non sono valori critici, ma all'interno di questi limiti il paziente è più in salute. Il sistema si è mostrato perfettamente in grado di mantenere una media attorno ai 100 con svariate caratteristiche dei pazienti.

Requisiti non funzionali:

- Il primo requisito non funzionale viene calcolato dal file synth.py. Questo si occupa, preso un singolo paziente generato casualmente, di ottimizzare verso il basso la quantità totale di insulina iniettata durante la simulazione. Questo viene fatto modificando in maniera incrementale i valori di a e b presenti in pump.mo, i quali influiscono fortemente sull'insulina iniettata. Il synth.py continua ad effettuare simulazioni finché il sistema non fallisce, dopodiché restituisce i valori migliori di a e b per i quali ha iniettato complessivamente meno insulina. Il sistema fallisce se il paziente raggiunge valori critici, o se sorgono problemi aritmetici, da noi gestiti (come valori di a e b negativi o calcoli che portano ad errori perché ad esempio troppo elevati per la simulazione). Il sistema è resistente ai fallimenti dei requisiti funzionali, ed avendo deciso che i risultati migliori si ottengono aumentando a e diminuendo b, è probabile che le simulazioni si fermino per b negativo.

- Il secondo requisito non funzionale richiede la gestione dei pazienti ipoglicemici. (per la descrizione, vedere operational scenarios). Il monitor_hypogly.mo, ad un tempo poco dopo l'inizio della simulazione, cambierà il valore alla variabile di controllo se rileva che il glucosio minimo fino a quel momento è basso (e Non dipende dalla pompa, come spiegato in operational scenarios). Quindi, mealgen.mo aumenterà la frequenza, durata e nutrimento dei pasti, per far sopravvivere il paziente. In un contesto reale, è come se la pompa in un primo periodo effettuasse un monitoraggio delle condizioni del paziente, e se le trovasse insufficienti, un allarme lo spingerebbe a mangiare di più (oppure un medico che interpreta l'allarme). Questo ha portato ad aumentare i tempi di simulazione, ed a

ritardare le azioni di monitoraggio degli altri requisiti, in quanto questa correzione applicata richiede del tempo extra prima di stabilizzarsi definitivamente. Di default, la correzione viene applicata per glucosio inferiore a 70. La pompa si mostra perfettamente in grado di gestire il cibo extra, senza far aumentare eccessivamente i livelli di glucosio.

5 Experimental Results

Il sistema è dimostrato funzionante dal gran numero di simulazioni eseguite dai suoi script.
Il verify.py esegue 100, 1000 e 10000 simulazioni verificanti i due requisiti funzionali ed il requisito non funzionale dei pazienti ipoglicemici. Come si può vedere nei file di risultati, il sistema funziona in ogni caso.
Il synth.py esegue 100, 1000 simulazioni verificanti il requisito non funzionale dell'insulina iniettata per singolo paziente, avendo cura di rispettare gli altri requisiti. Per ogni paziente esegue circa 17 simulazioni, restituendo tra quelle i valori migliori trovati, come si può vedere dai file dei risultati.
In questa fase, si è dimostrato anche come le molte ottimizzazioni operate a livello di codice abbiano dato i loro frutti. Ad esempio, nel caso verify.py a 10000 pazienti, il sistema è passato dal richiedere tempo stimato pari a circa 33 ore all'inizio (tempo calcolato matematicamente), a completare il suo compito in poco più di 2 ore (tempo effettivo).

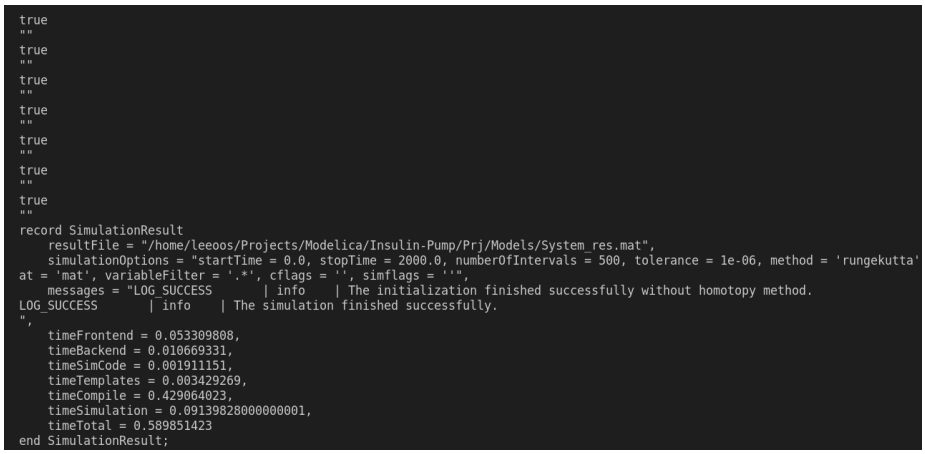

Figura 1: Tempo di esecuzione per una singola simulazione.

Paziente Non Ipoglicemico

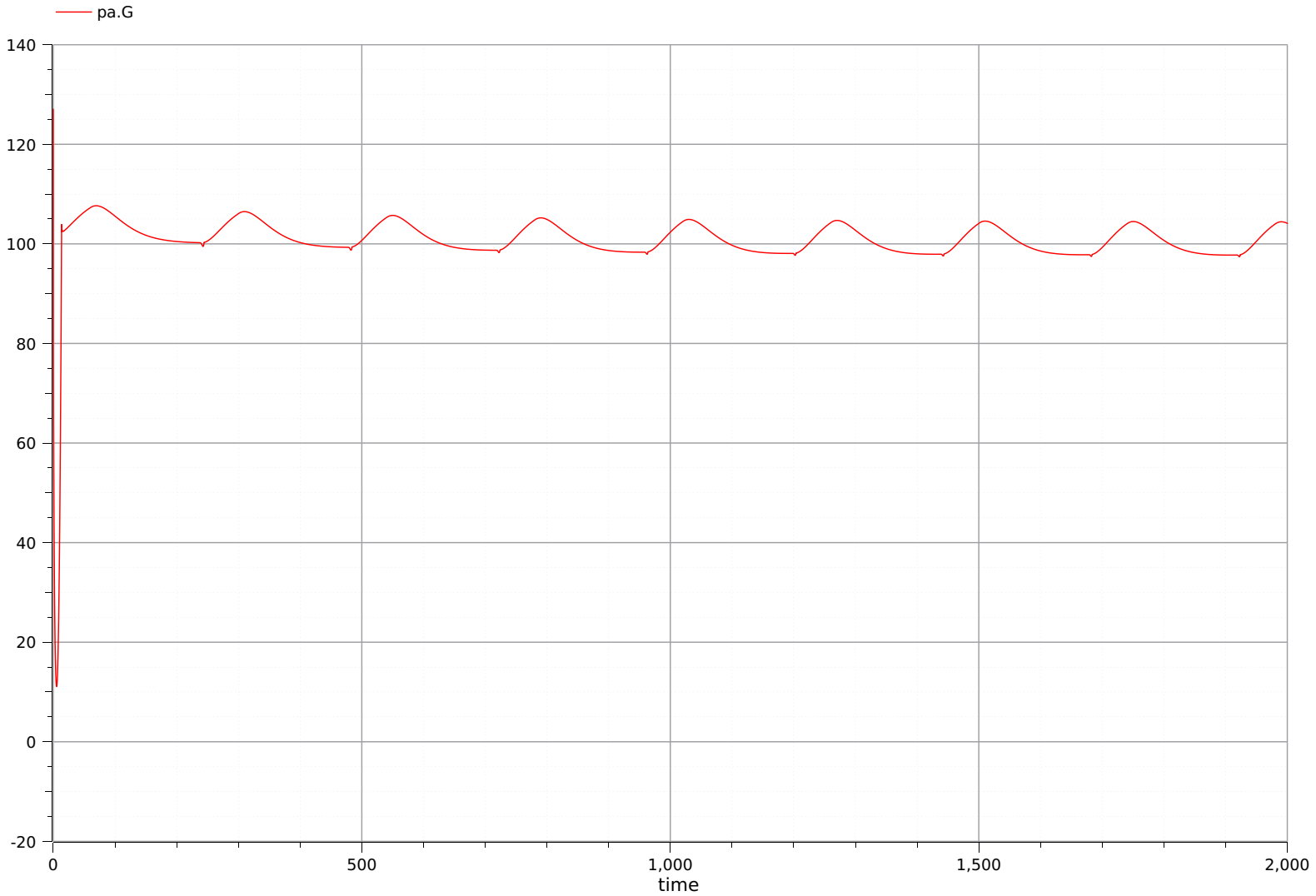

Grafico 1: Andamento del livello di glucosio in un paziente non ipoglicemico.

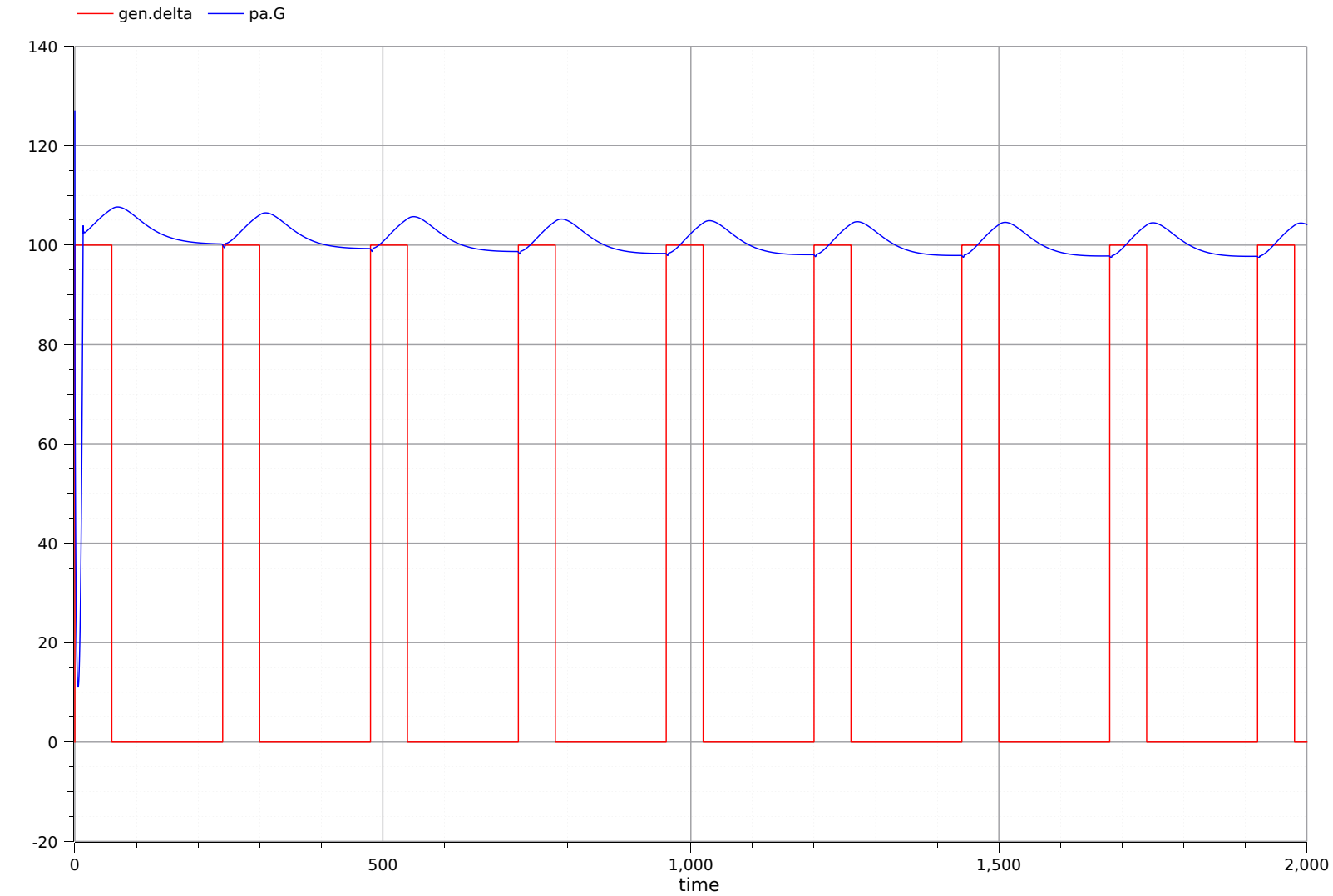

Grafico 2: Andamento del livello di glucosio e frequenza dei pasti di un paziente non ipoglicemico.

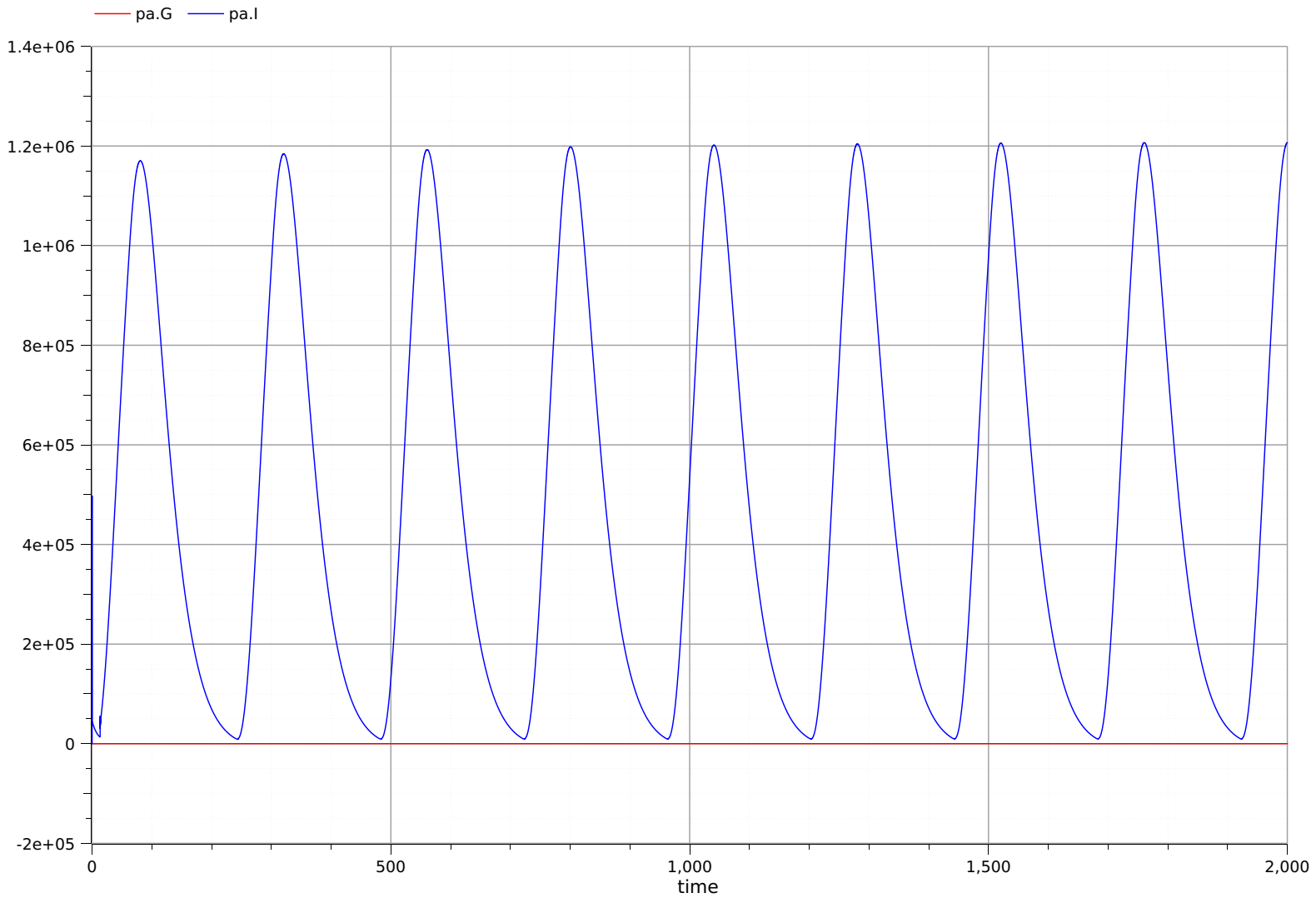

Grafico 3: Andamento del livello di glucosio e insulina nel paziente durante la simulazione.

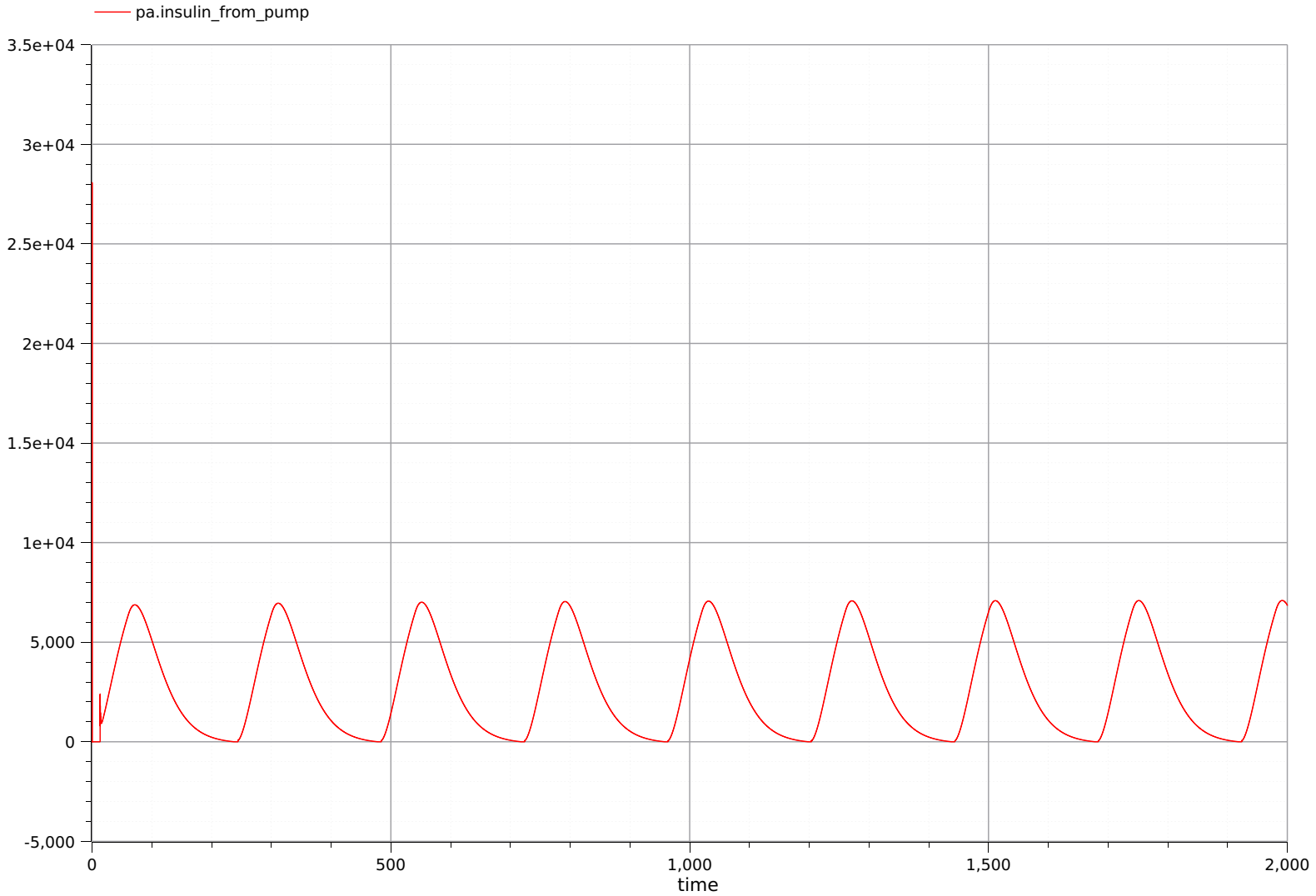

Grafico 4: Andamento del livello di insulina iniettata dalla pompa durante la simulazione.

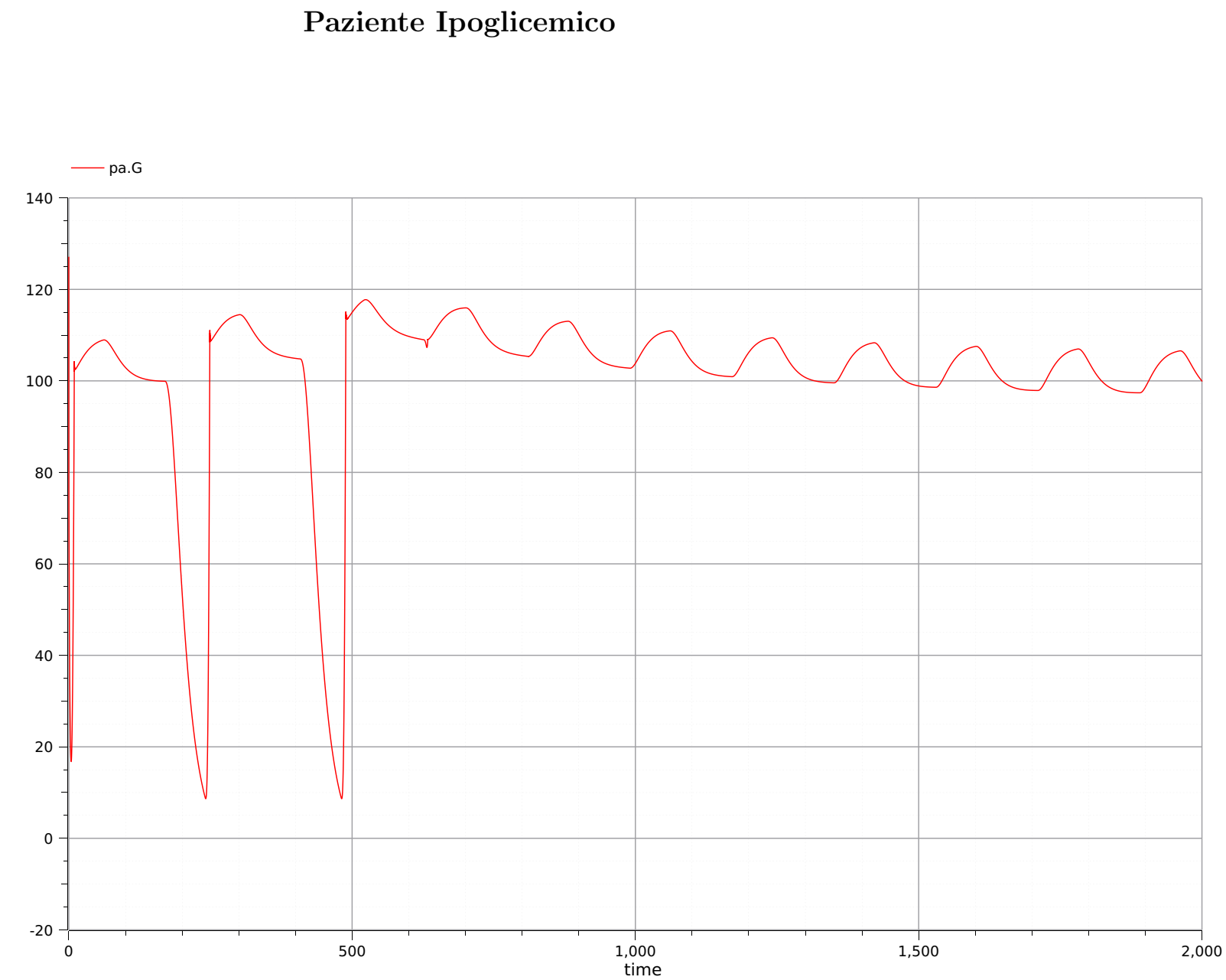

Grafico 5: Andamento del livello di glucosio in un paziente ipoglicemico al quale è stata applicata la correzione.

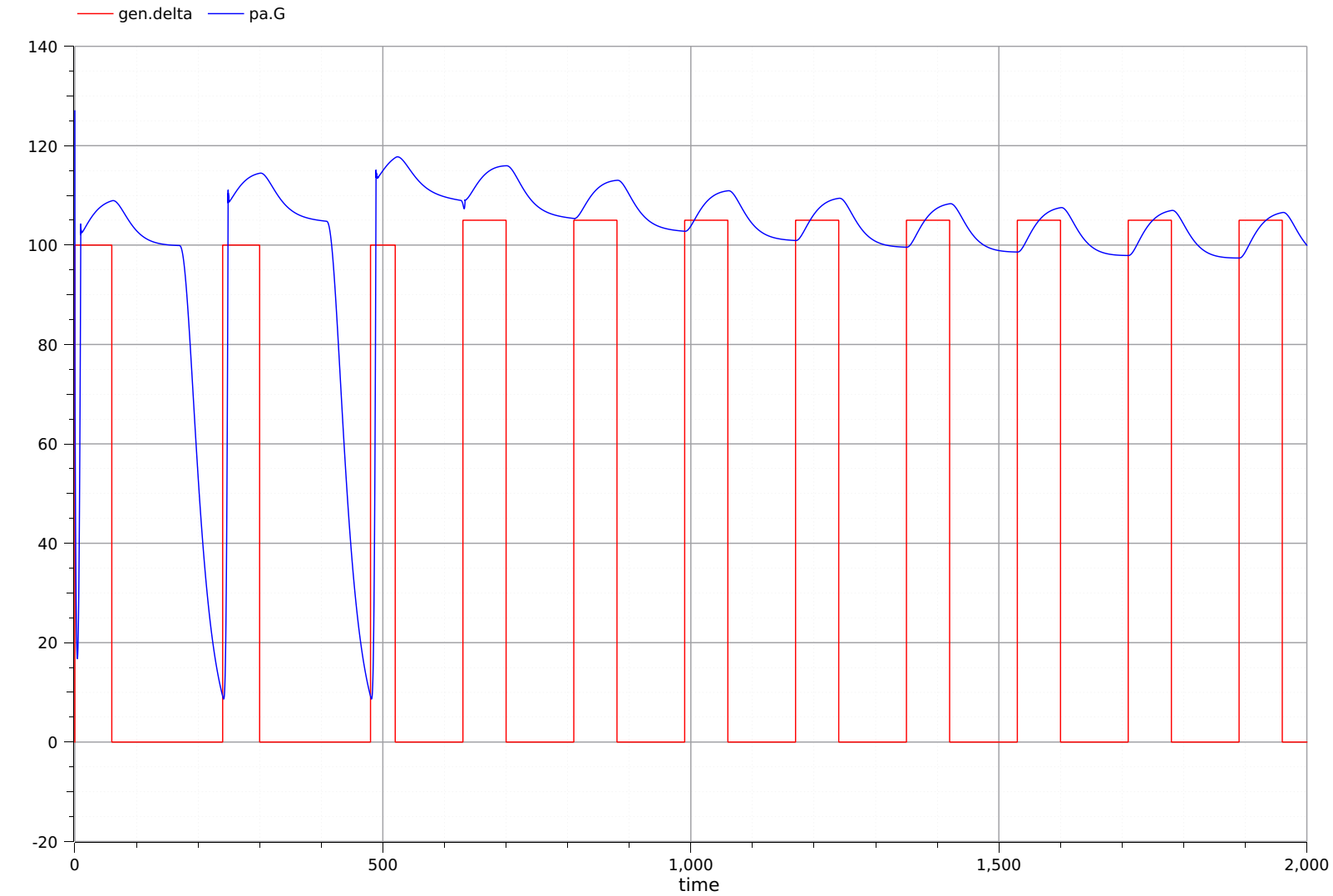

Grafico 6: Andamento del livello di glucosio e frequenza dei pasti di un paziente al quale è stata applicata la correzione.

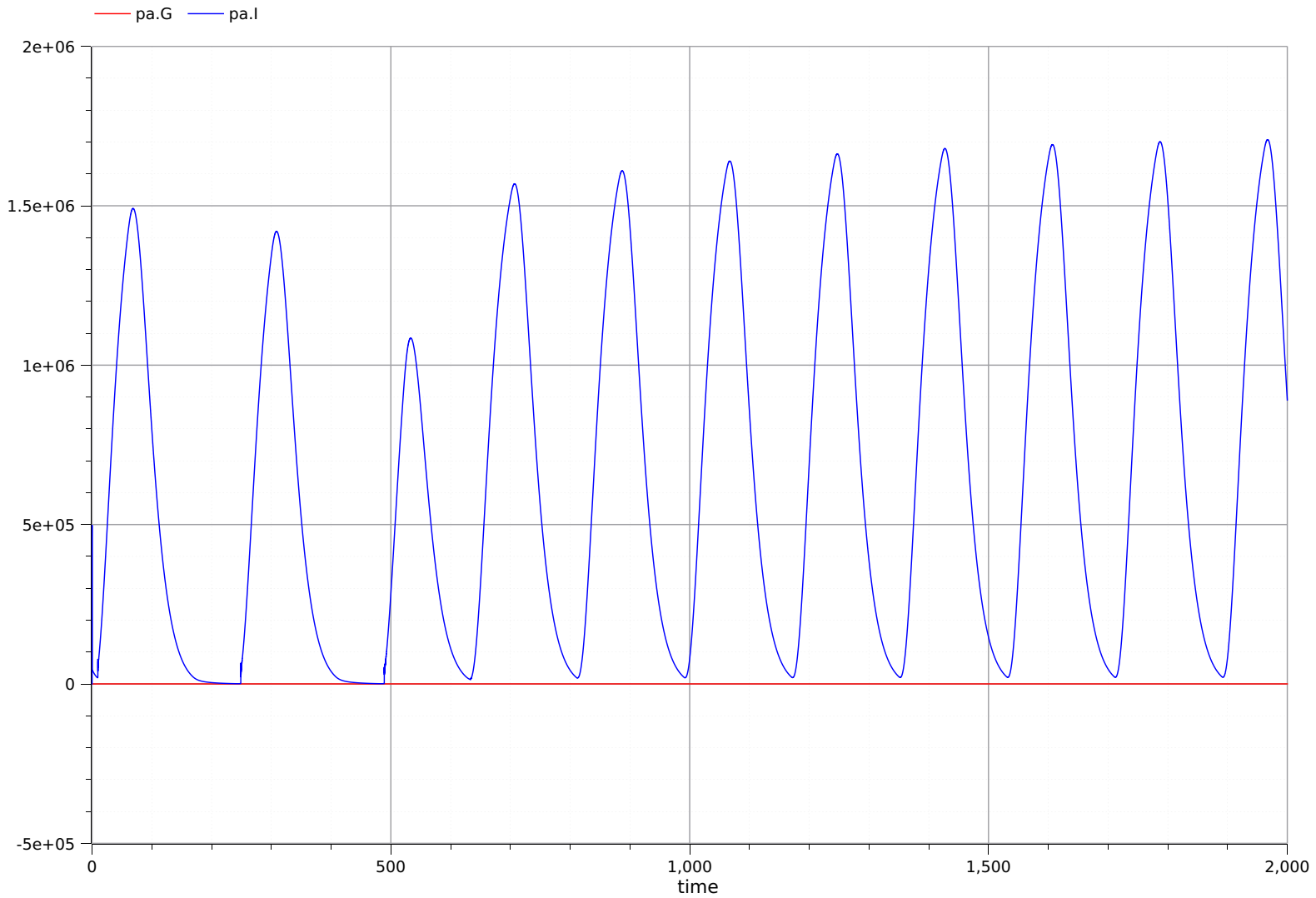

Grafico 7: Andamento del livello di glucosio e insulina di un paziente al quale è stata applicata la correzione.

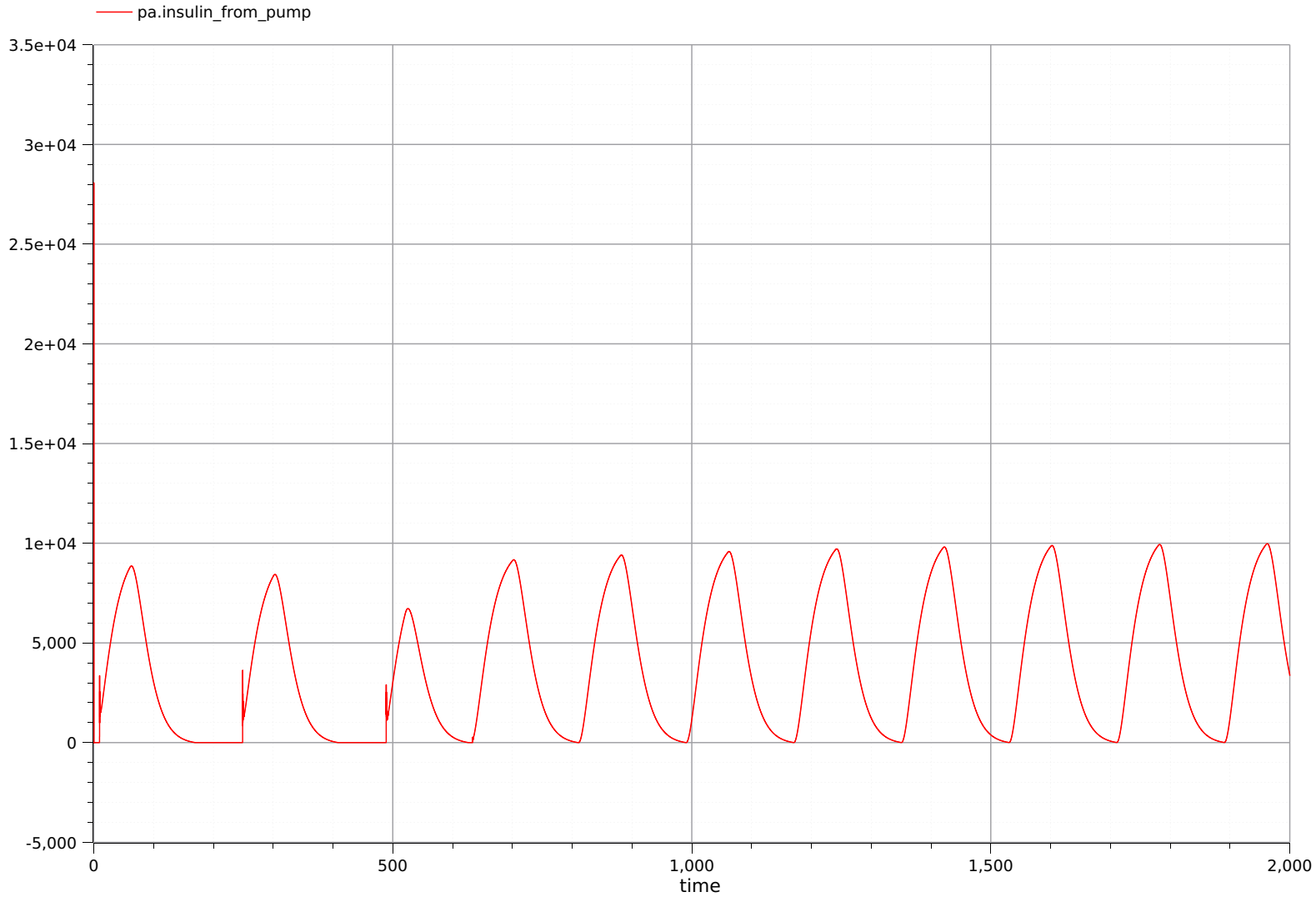

Grafico 8: Andamento del livello di insulina iniettata dalla pompa durante la simulazione ad un paziente ipoglicemico.