# INDICE

# *I. INTRODUZIONE*

Nell’ambito di un tirocinio di 500 ore presso l’IS.I.B. (Istituto di Ingegneria Biomedica) del C.N.R. di Padova, sono stato inserito dagli Ing. A. Mari e A. Tura nell'attivita' di ricerca che riguarda lo studio del diabete. In particolare ho studiato due distinte applicazioni allo studio del metabolismo del glucosio, che hanno potenziale rilevanza per la ricerca sul diabete.

Tali applicazioni vengono illustrate in questa tesi:

- nella I parte verranno descritte alcune esperienze di laboratorio di misure di impedenza di soluzioni fisiologiche contenenti glucosio, rilevanti per lo sviluppo di un metodo di misura non invasiva della glicemia;

- la II parte tratterà della simulazione di modelli fisiologici per lo studio quantitativo del metabolismo, partendo da alcuni elementi di base per giungere ad applicazioni a modelli tratti da una libreria codificata nel Systems Biology Markup Language.

# *II. MISURE DI IMPEDENZA DI SOLUZIONI DI GLUCOSIO*

# II.I. INTRODUZIONE

In campo diabetologico esistono numerose apparecchiature in grado di misurare la glicemia di un paziente, anche in ambito domiciliare, sfruttando la reazione della glucosio – ossidasi su un piccolo campione di sangue del paziente: questa metodologia richiede però il prelievo di una goccia di sangue, ed è quindi di tipo invasivo. Per questa ragione, molti studi sono stati compiuti negli ultimi anni per raggiungere un metodo di misura della glicemia di tipo non invasivo. Un possibile approccio è quello basato sulla misura di impedenza, poiché vi sono indicazioni riguardo al fatto che variazioni di concentrazione della glicemia determinino variazioni misurabili dell'impedenza del sangue. Nel seguito si esplorano, in un contesto per il momento in vitro, le proprietà dell'impedenza di soluzioni aventi alcune similitudini con il sangue, al variare della concentrazione di glucosio disciolto all'interno delle soluzioni stesse.

# II.II. MISURE D'IMPEDENZA

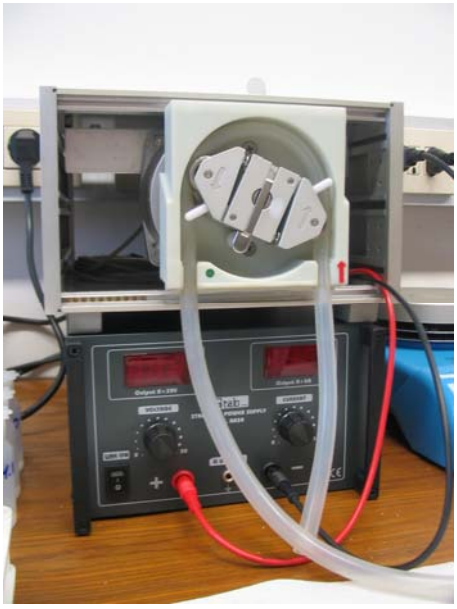
## II.II.I. DESCRIZIONE ATTIVITÀ

Al fine di trovare un legame misurabile tra la concentrazione di glucosio e l'impedenza della soluzione, ho eseguito numerose sessioni di misura del modulo dell'impedenza di soluzioni fisiologiche saline (sodio-cloruro 0,9%) a concentrazione di glucosio e condizioni variabili. Gli esperimenti sono stati effettuati mediante un circuito idraulico integrato con una sonda a due elettrodi e alimentato da una pompa.

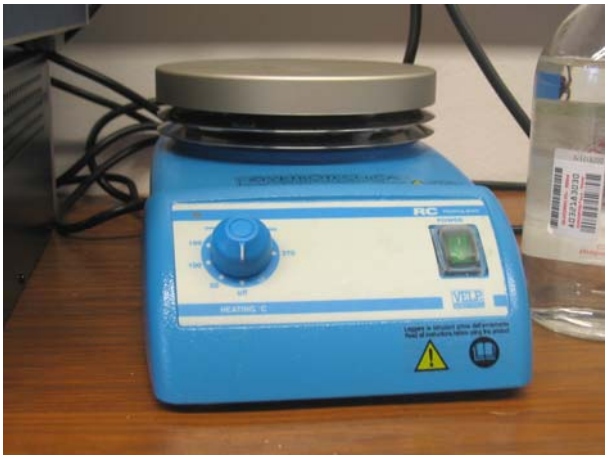
## II.II.II. STRUMENTAZIONE E MATERIALI

- Analizzatore di impedenza/guadagno – fase Solartron Schlumberger mod. 1260

- Pompa peristaltica
- Alimentatore stabilizzato, mod. AR30 (Stab)



- Rc heating plate (piastra riscaldante) (Velp Scientifica)



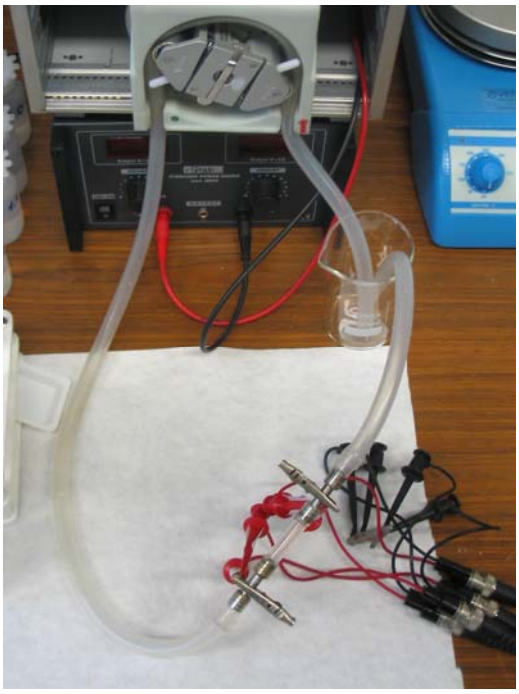
- Sonda a due elettrodi

- MICROSTIRRER (Velp Scientifica)
- Sodium chloride, (Sigma)
- D-(+)-Glucose anhydrous (Fluka, BioChemika)
- Sodio Cloruro 0,9% soluzione per infusione (Baxter)

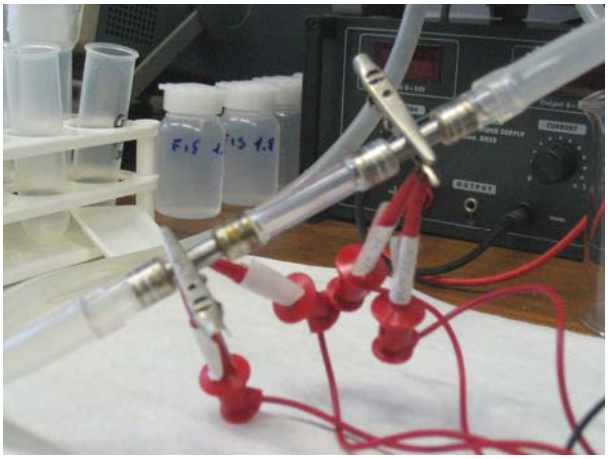
## II.II.III. PROCEDURA

Scopo degli esperimenti è stato quello di osservare il variare del modulo dell'impedenza di vari tipi di soluzioni in funzione del valore della concentrazione di glucosio. L'obbiettivo principale era la determinazione di una correlazione tra la concentrazione di glucosio in soluzione fisiologica al 0,9% e la variazione del modulo dell'impedenza della soluzione stessa, al mutare di determinati parametri.
L'apparecchiatura utilizzata era costituita da:

- Un circuito idraulico così composto: una pompa, collegata a un alimentatore a corrente continua, che generava un flusso di soluzione all'interno di due tratti di tubo in silicone, il primo dei quali attingeva la soluzione stessa da un becker per condurla fino al primo elettrodo della sonda, al cui secondo elettrodo era unito il secondo tratto di tubo che terminava nel becker per chiudere il circuito.



- L' analizzatore di guadagno e fase collegato al polo positivo della sonda mediante i morsetti "Gen output" e "Input V1 HI", all'anodo con i morsetti "Input" e "Input V1 LO",



- e con i morsetti di massa connessi ad un oggetto metallico

- Un computer in collegamento con l'analizzatore, dotato del software di interfacciamento con l'analizzatore "SMaRT", che permette di controllare lo strumento via calcolatore
- Una bilancia elettronica per pesare le sostanze da sciogliere nella soluzione fisiologica e un miscelatore per rendere omogenea la soluzione stessa.

Le procedure seguite per svolgere le misure sono state diverse, in base al tipo di misurazione effettuata; tuttavia alcuni aspetti risultano comuni a tutte: ove non diversamente specificato, come solvente è stata utilizzata soluzione fisiologica sodio – cloruro 0,9%, come soluto D-glucosio, la tensione di alimentazione della pompa è stata impostata a 10V, le rilevazioni sono state effettuate nel range di frequenze 10 MHz – 1 Hz (espressamente in ordine decrescente per non provocare a inizio misurazioni fenomeni di polarizzazione degli elettrodi) con 5 rilevamenti per decade in scala logaritmica, tensione di sollecitazione esclusivamente alternata del valore di 100 mV, tra misurazioni consecutive il circuito idraulico è stato pulito facendovi scorrere all'interno acqua distillata e successivamente una piccola quantità della soluzione da misurare poi scartata. Inoltre, la temperatura di tutte le soluzioni prese in considerazione è stata monitorata durante il flusso all'interno del circuito mediante un termometro all'altezza del becker.

*Variazione concentrazione glucosio*

Le misure sono state eseguite su campioni di soluzione a diverse concentrazioni di glucosio con diluizione seriale 1:2 (5000, 2500, 1250, 625, 312, 156, 76 mg/dl).
Per ciascuna soluzione, è stato versato nel becker abbastanza liquido da riempire il circuito (vuotandolo da eventuali sacche d'aria che avrebbero potuto influenzare i valori ottenuti) che veniva poi avviato e lasciato funzionare

liberamente per pochi minuti così da rendere omogeneo il gradiente di concentrazione in ogni zona del tubo e della sonda. Inserito il termometro nel becker per monitorare la temperatura della soluzione esaminata, venivano avviate due misure consecutive dello stesso campione per verificare la ripetibilità dei risultati mediante i grafici dei rispettivi moduli d'impedenza visualizzabili in SMaRT.

*Variazione temperatura*

Sono stati presi in considerazione tanti campioni di soluzione a 1250 mg/dl quante le temperature della soluzione a cui effettuare le misure: testato dapprima un campione a temperatura della stanza (circa 21,5 °C), i successivi (rispettivamente a 31 °C, ~37 °C, 45 °C) sono stati scaldati mediante una piastra riscaldante inserita nel circuito sotto il becker.

*Basse frequenze*

Si è indagato il comportamento di alcune soluzioni anche alle basse frequenze (1 – 0,01 Hz), procedendo sempre dalla frequenza più elevata: per campioni di fisiologica pura e soluzione di fisiologica e glucosio da 156 mg/dl sono stati condotti test alle basse frequenze sia con 5 che con 15 punti di misura per decade, risultando la prima misura contraria alle aspettative, mentre la coppia di soluzioni da 156 e 76 mg/dl è stata analizzata solo nella configurazione a 15 punti per decade.

*Variazione concentrazione sodio*

Per misurare l'impedenza di soluzione fisiologica caricata (con aggiunta di cloruro di sodio)
all'1,0%, all'1,1% e all'1,2% , è stata seguita la medesima procedura vista per le soluzioni a diverse concentrazioni di glucosio, fatta eccezione per la creazione delle soluzioni stesse: in questo caso, i campioni sono stati creati l'uno indipendentemente dall'altro, rispettivamente aggiungendo 1, 2 e 3 grammi di sodio per litro di soluzione allo 0,9% di partenza.

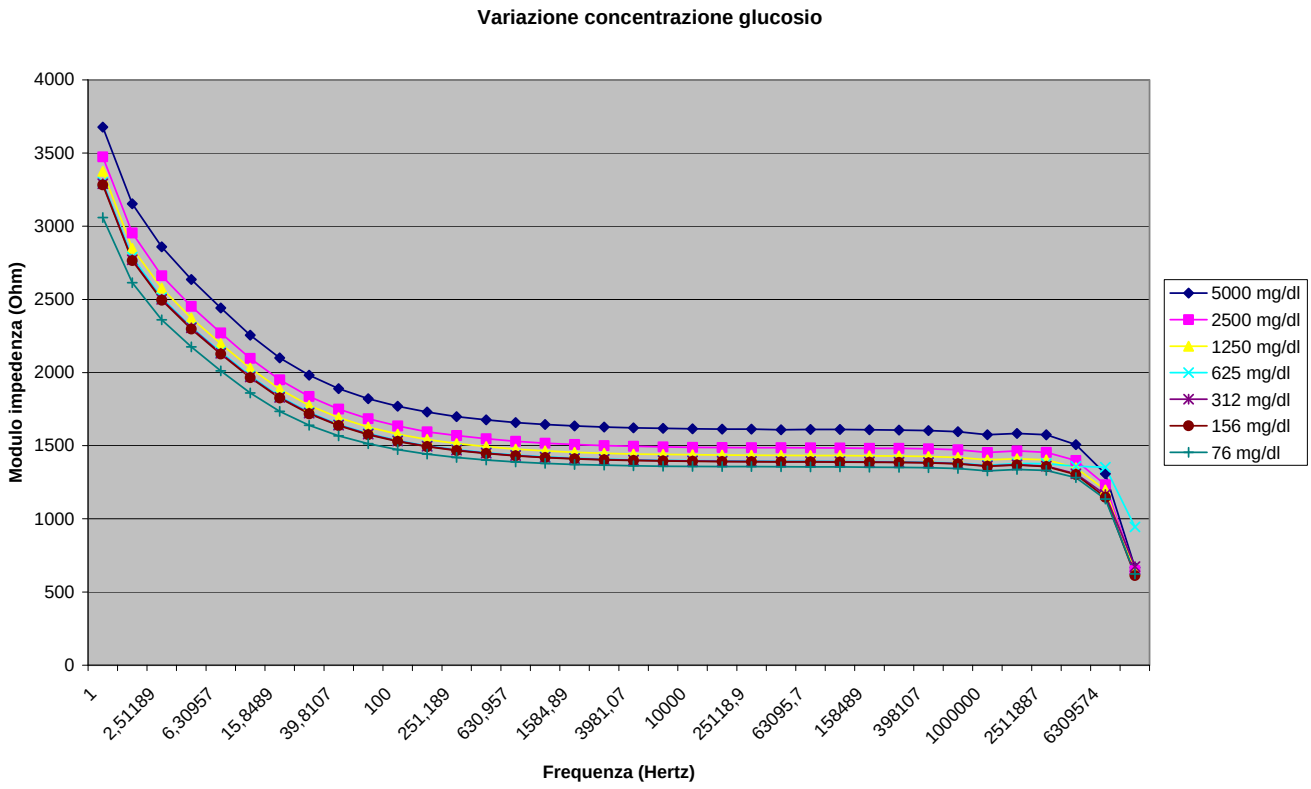
*Variazione di flusso*

Per soluzioni a 5000, 2500, 1250 mg/dl è stata misurata l'impedenza in condizioni di varie velocità di flusso all'interno del circuito idraulico, regolando la tensione fornita alla pompa dall'alimentatore; le tensioni considerate sono state: 0V (pompa ferma), 5V, 10V e 15V.

## II.II.IV. RISULTATI

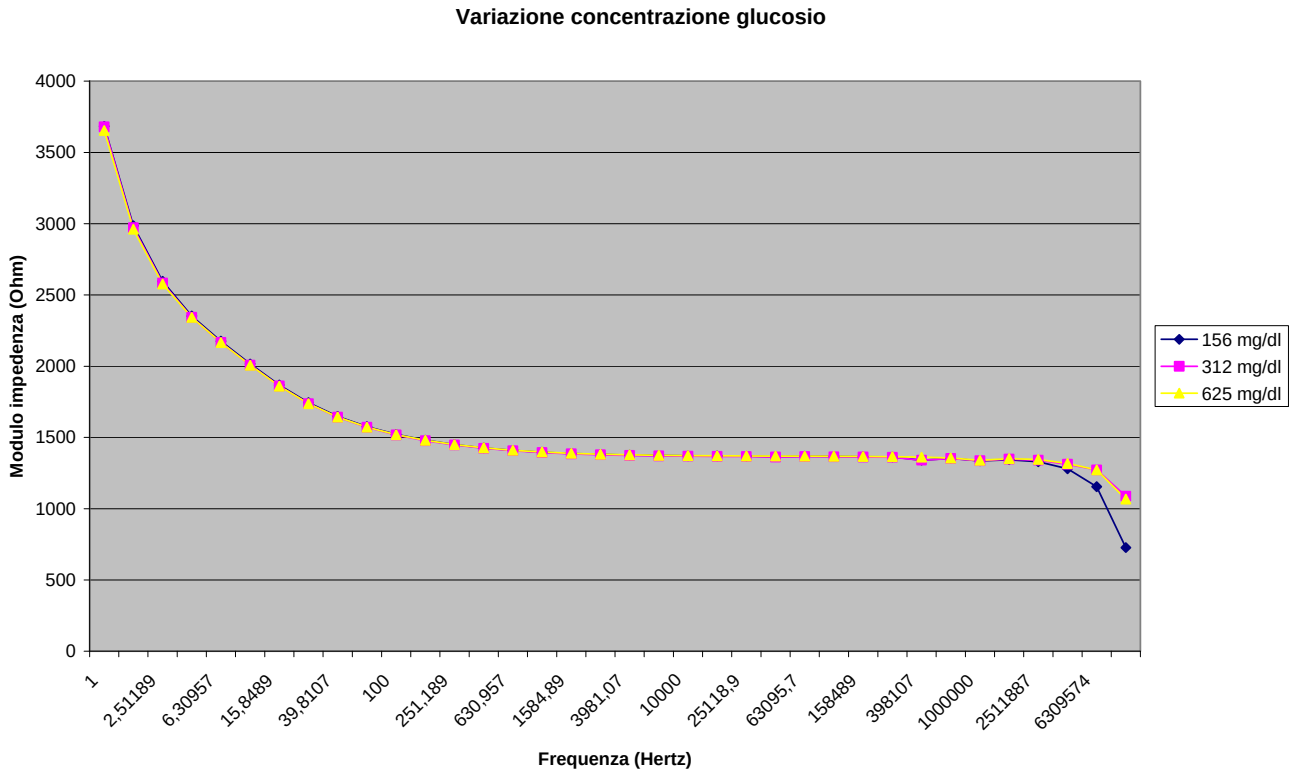
Effettuate le analisi, SMaRT consente di esportare i valori ottenuti per ogni misurazione, salvati in appositi file con estensione ".dat", nel formato Excel "csv": ho proceduto, quindi, ad unire tutte le misurazioni relative a una stessa tipologia di esperimento in uno stesso file "xls" generando il relativo grafico.

*Variazione concentrazione glucosio*



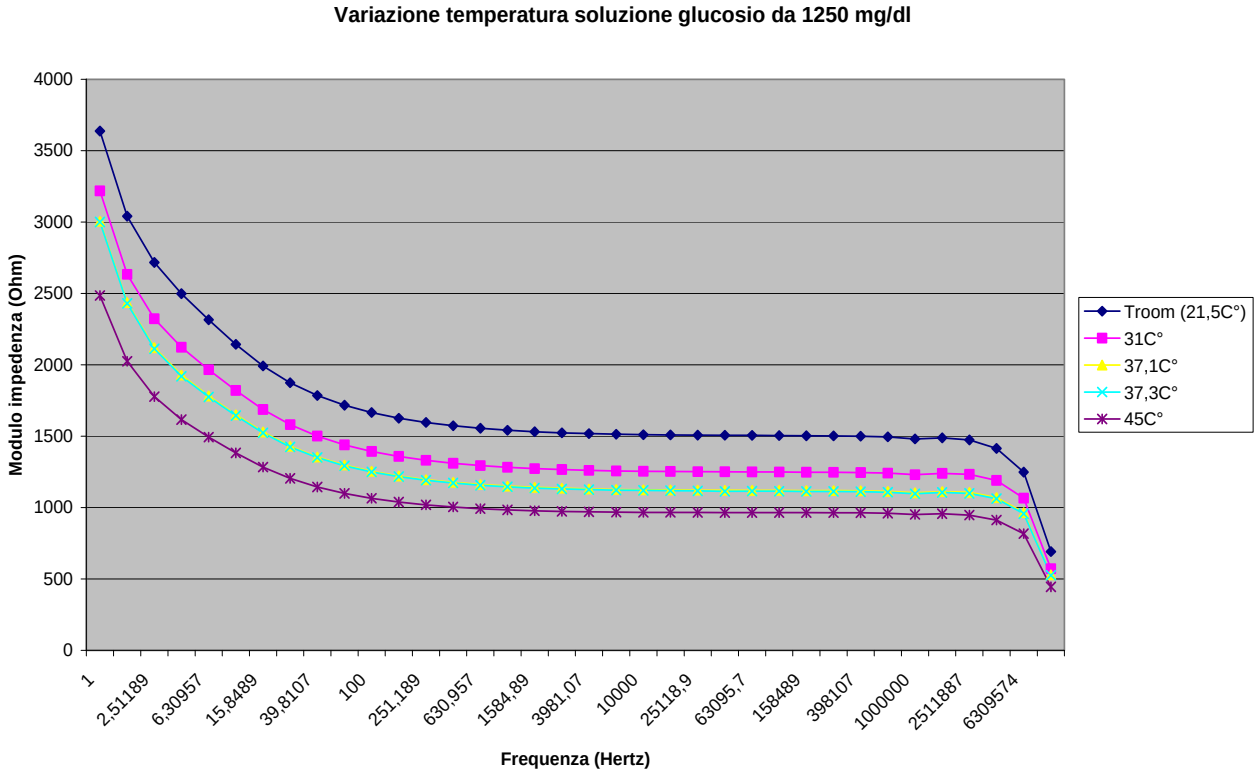

Dal grafico emerge che l'impedenza di una soluzione di glucosio aumenta all'aumentare della concentrazione del glucosio stesso: nonostante il glucosio sia una molecola elettricamente neutra, tali risultati dimostrano che il glucosio ha un effetto resistivo.

A partire, però, dalla soluzione da 625 mg/dl, dimezzando la concentrazione, si hanno curve fra loro sovrapposte a causa di un ipotetico errore sperimentale: si è, pertanto, proceduto a una ripetizione dell'esperimento per le concentrazioni prese in causa, con i seguenti risultati:



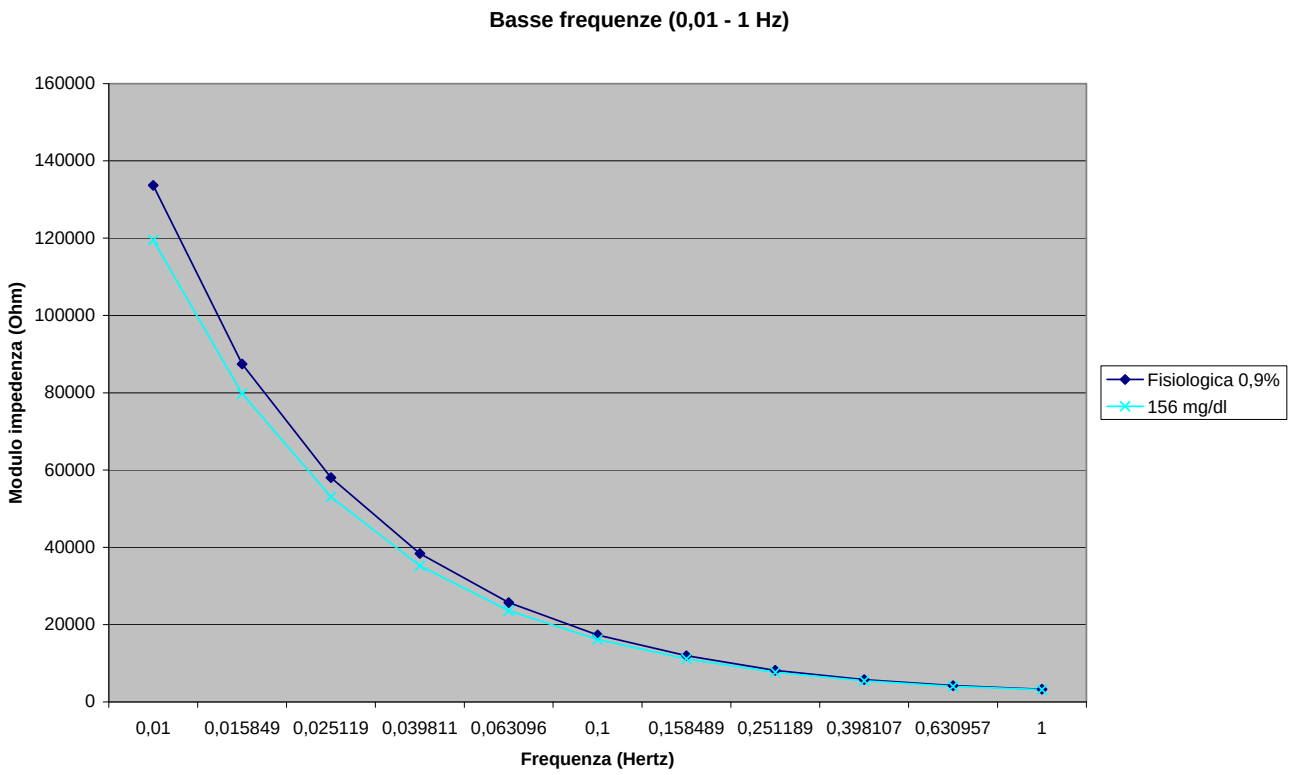

Dalla seconda prova, si nota come vi sia una quasi totale coincidenza fra le curve, probabilmente non dovuta a un errore sperimentale, bensì a variazioni di impedenza più contenute (e quindi poco apprezzabili) alle basse concentrazioni: il tipo di configurazione sperimentale utilizzata pare non adatta (probabilmente per l'eccessiva semplicità) a discriminare tra diverse concentrazioni di glucosio a valori relativamente bassi.

*Variazione temperatura*

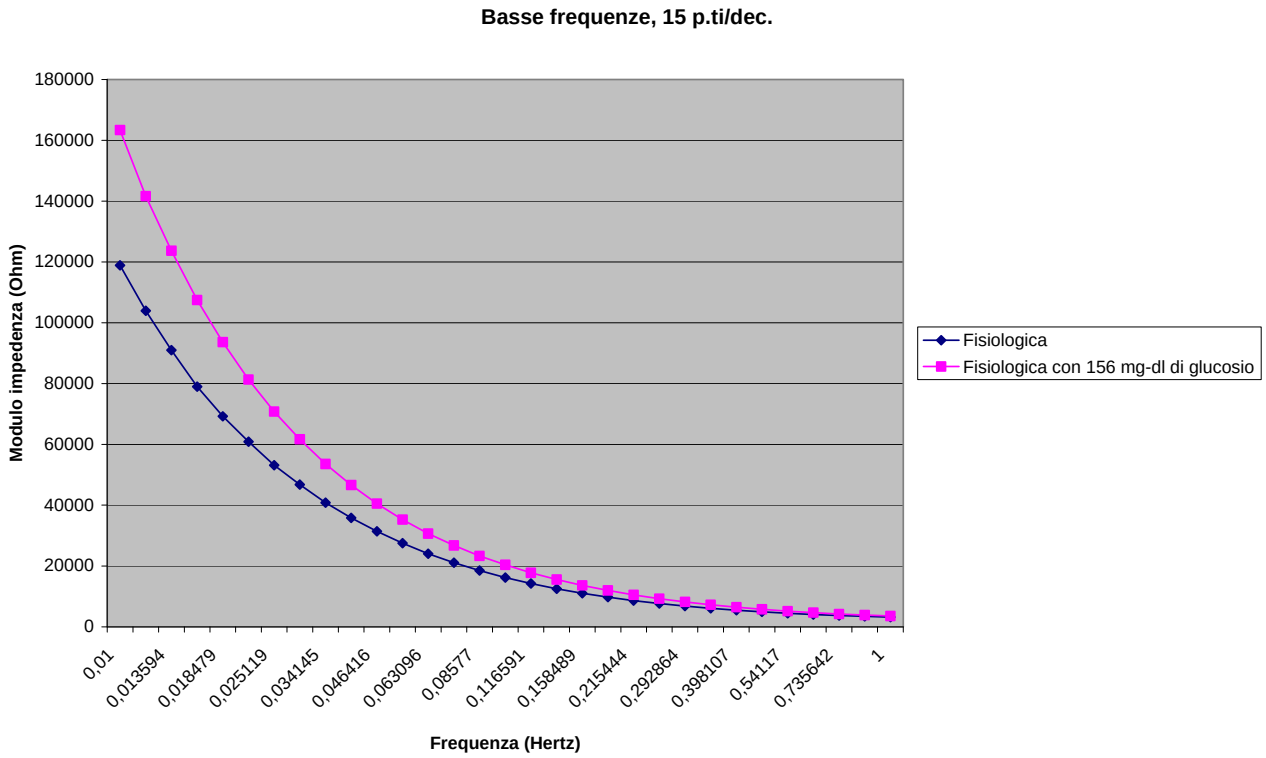



Aumentando la temperatura della soluzione in esame, si nota una diminuzione dell'impedenza, ovvero un aumento della conducibilità, dovuto a un maggiore stato di agitazione termica della molecole che di conseguenza hanno la possibilità di muoversi più liberamente una volta sollecitate da un campo elettrico esterno.
Ciò in pieno accordo con le informazioni riportate in letteratura in merito agli effetti della temperatura sull'impedenza di una soluzione.

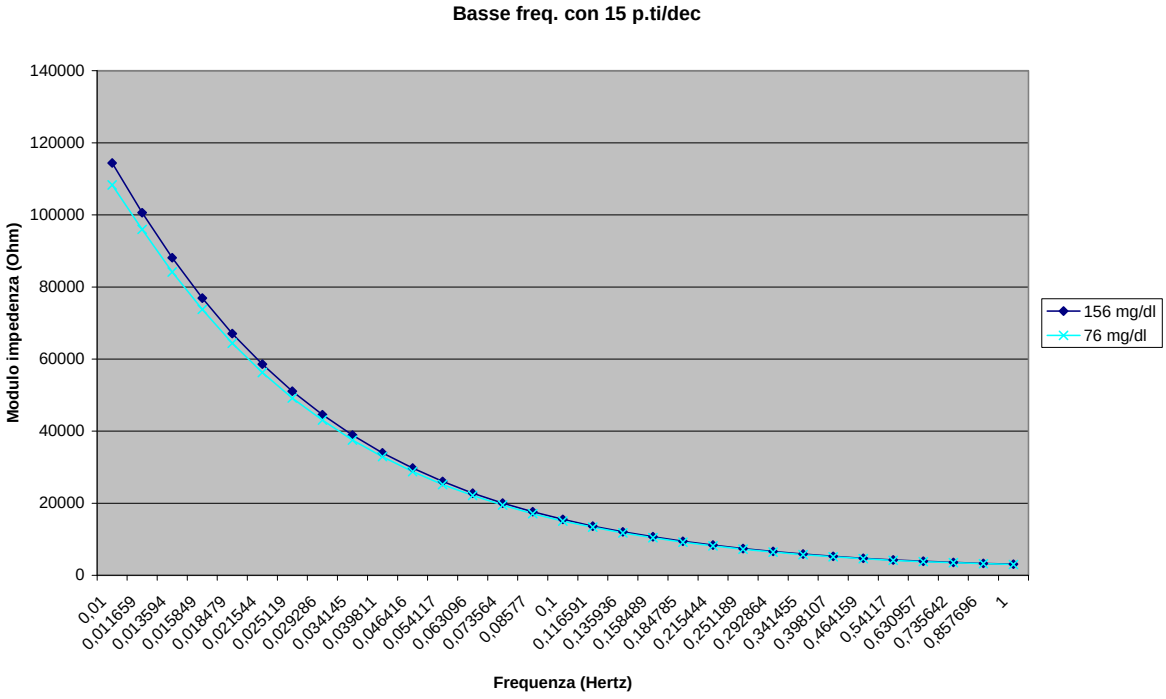
*Basse frequenze*




Eseguendo l'analisi alle basse frequenze sui campioni di fisiologica pura e di fisiologica con 156 mg/dl di glucosio, sono stati ottenuti dei risultati in controtendenza rispetto alle attese, in cui la conducibilità della soluzione contenente glucosio è risultata maggiore dell'altra: questo errore potrebbe essere dovuto al manifestarsi di fenomeni di polarizzazione degli elettrodi insorti a causa delle basse frequenze di analisi. Anche in questo caso si è proceduto di conseguenza a una seconda prova, portando però a 15 i punti di misura per decade:

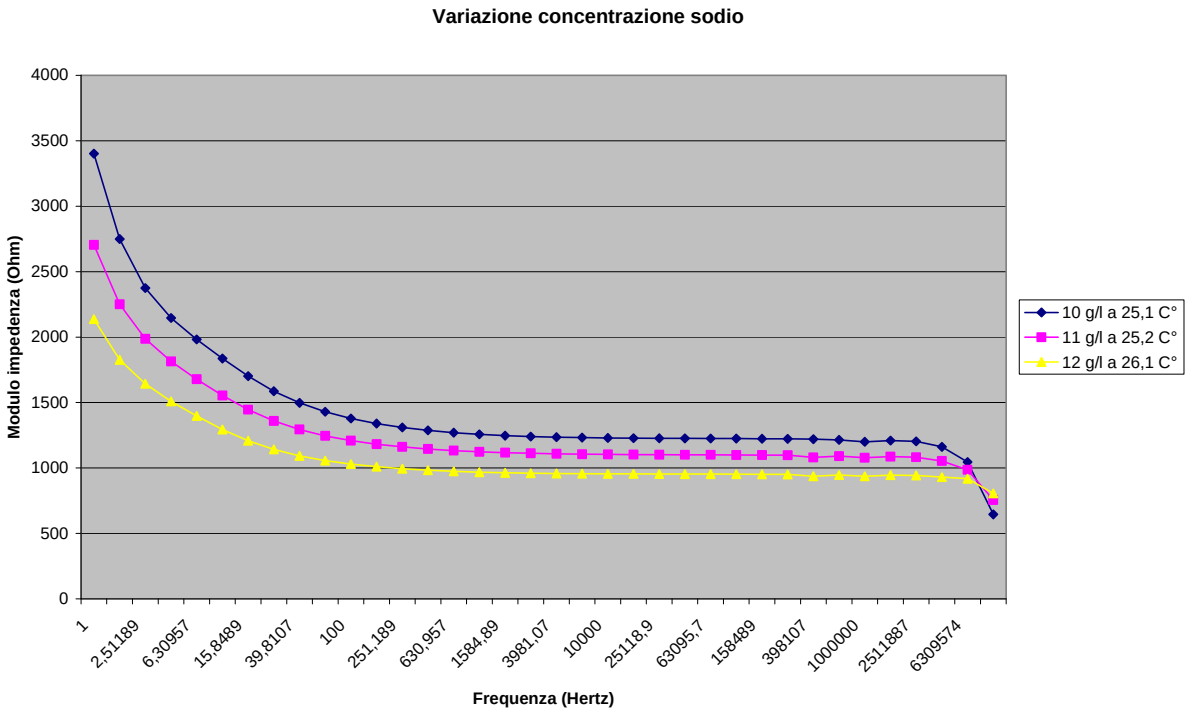



I risultati del secondo test rispecchiano le attese, secondo cui l'impedenza della soluzione fisiologica pura sarebbe dovuta essere minore della soluzione contenente glucosio.



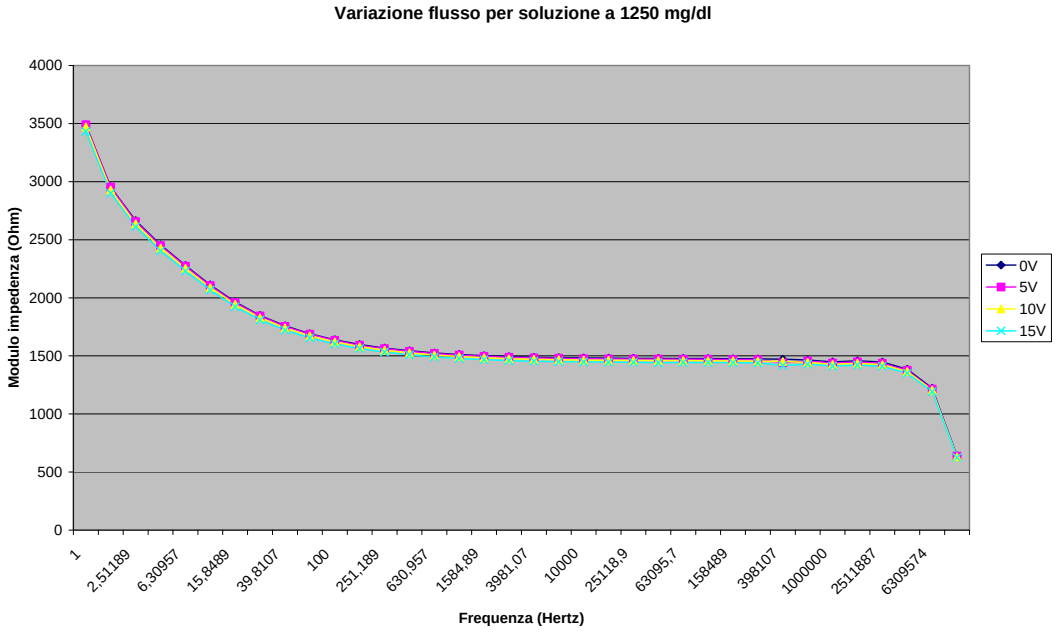

Anche le misure effettuate sulle soluzioni rispettivamente da 156 e 76 mg/dl rispecchiano il trend previsto, nonostante i limiti e le difficoltà dell'analisi a basse frequenze.

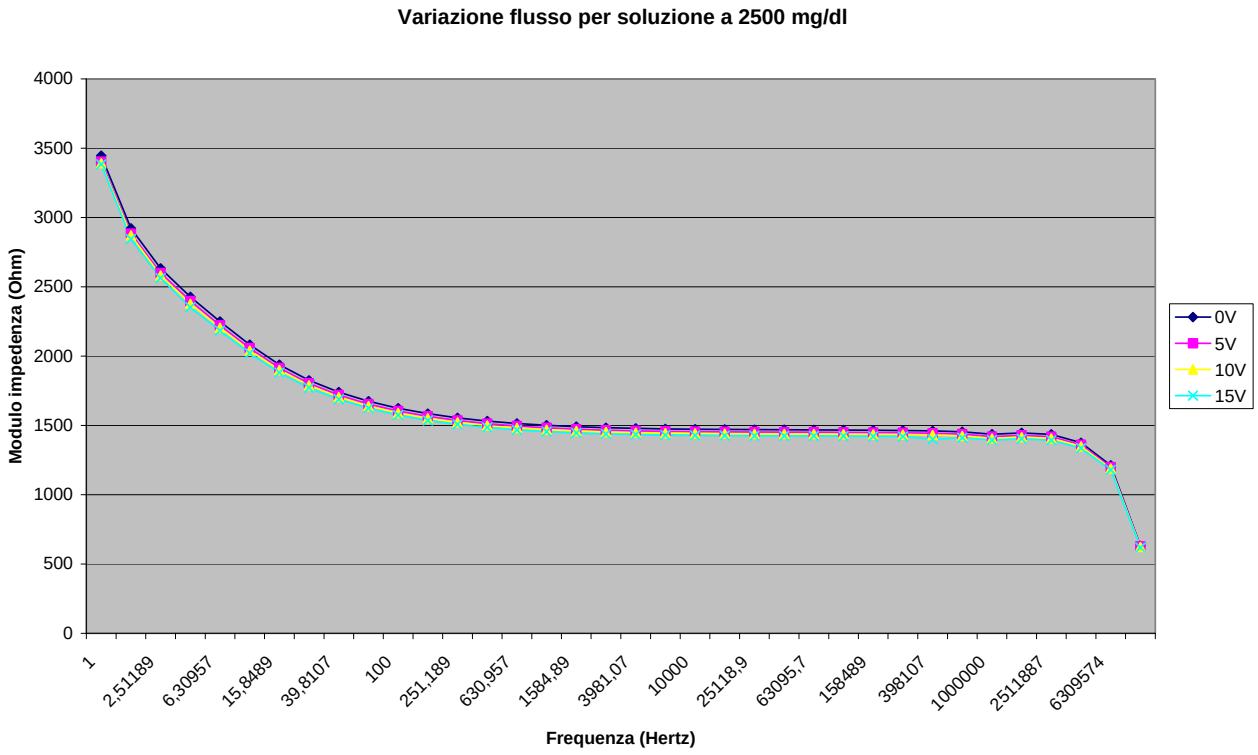
*Variazione concentrazione sodio cloruro*

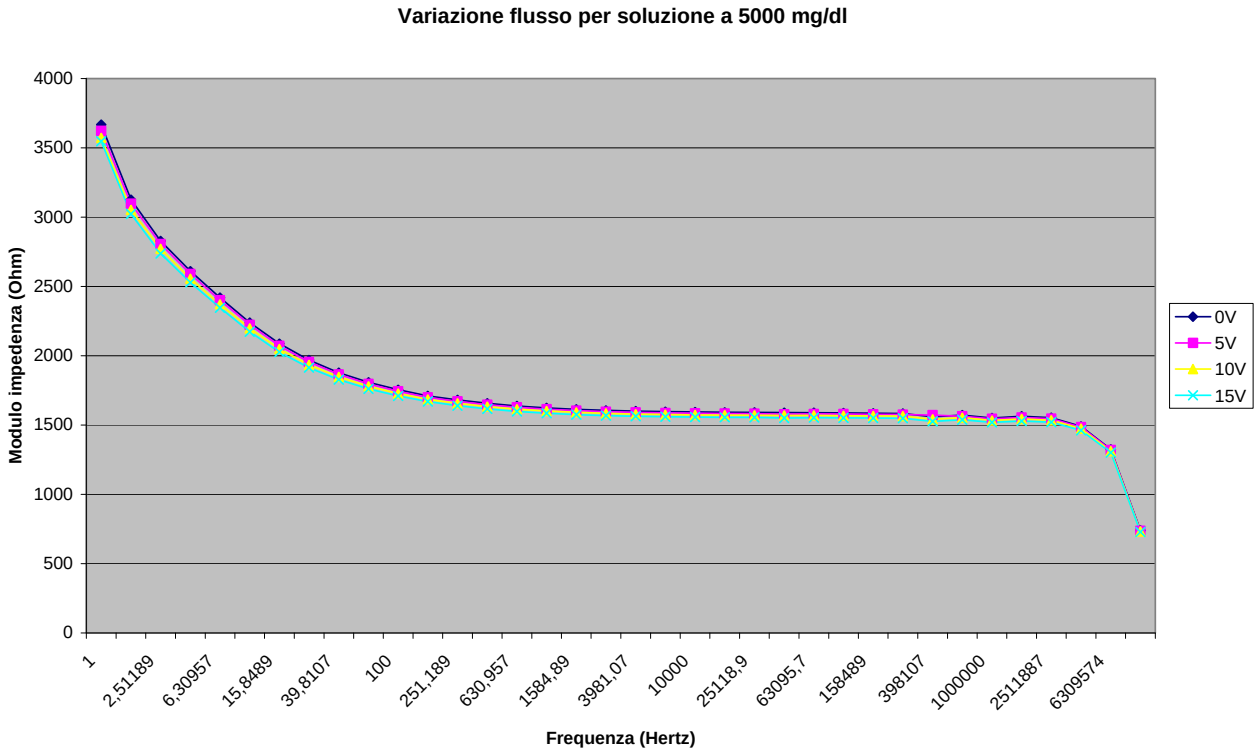



All'aumentare della concentrazione di sodio cloruro, dal grafico si evince che cresce conseguentemente la conducibilità elettrica della soluzione, trattandosi di un sale che si dissocia in ioni (ione sodio e ione cloro), i quali quindi risentono del campo elettrico esterno applicato.

*Variazione del flusso*






Variando il flusso della soluzione esaminata all'interno del circuito idraulico, non si notano cambiamenti rilevanti nell'andamento del modulo dell'impedenza, indipendentemente dal grado di concentrazione di glucosio: tuttavia, esaminando con attenzione piccole porzioni di spettro, si possono denotare dei cambiamenti, per quanto contenuti: in particolare, l'impedenza

tende leggermente a calare al crescere della velocità di flusso: probabilmente, un aumento della velocità di scorrimento delle molecole può facilitare il flusso di corrente, con conseguente aumento di conducibilità. Poiché sono state ottenute le medesime tendenze per tutte le soluzioni prese in esame, le prove non sono state ripetute.

## II.II.V. CONSIDERAZIONI

Il metodo della misura del modulo dell'impedenza sembrerebbe rappresentare idealmente un approccio promettente per sviluppare una metodica di analisi della glicemia in modo non invasivo, valida come possibile alternativa ai metodi basati sulla reazione enzimatica della glucosio – ossidasi. Rimane comunque il problema dei confondenti, quali ad esempio il sodio, che pare rappresentare un importante elemento di disturbo per una misura di glicemia tramite impedenza nei liquidi fisiologici.

Nei paragrafi seguenti si illustra il comportamento di resine a scambio ionico, che potrebbero essere una valida strategia per eliminare gli effetti confondenti del sodio. Va osservato che l'uso di tali resine determina un allontanamento dalla possibile applicazione di tipo non invasivo (non è possibile pensare di iniettare una resina – ammesso che svolga il compito desiderato – per poter poi misurare la glicemia in modo non invasivo!). Tuttavia, vi sono due considerazioni da presentare:

1. In questa fase si sta facendo pura ricerca di base (avendo quindi in mente una possibile applicazione clinica futura, ma per il momento anche prescindendo da questa);
2. In ogni caso, anche l'uso di resine potrebbe rivelarsi utile per una applicazione clinica relativa alla misura della glicemia: non certo per lo sviluppo di un metodo non invasivo (escluso per l'ovvia ragione spiegata sopra), ma per lo meno al fine di migliorare l'attuale tecnica di misura basata sulla glucosio ossidasi, utilizzata nei glucometri per uso domiciliare tradizionali: questi ultimi risultano in generale poco precisi ed affidabili (ed oltretutto ogni singola misura, effettuata con lo "stick", risulta abbastanza costosa): per questo potrebbero essere comunque di interesse tecniche di misura alternative, che migliorino la situazione dal punto di vista della accuratezza, o dei costi, o di entrambi gli aspetti. Una possibile tecnica alternativa potrebbe essere appunto la misura di impedenza della goccia di sangue del paziente, previo eventuale

# II.III. MISURE D'IMPEDENZA DI SOLUZIONI FILTRATE MEDIANTE RESINE A SCAMBIO IONICO

## II.III.I. DESCRIZIONE ATTIVITÀ

Sono stati studiati i risultati prodotti dal filtraggio di soluzioni fisiologiche mediante due tipi di resine a scambio ionico: gli aspetti indagati sono stati l'effettiva eliminazione del maggior agente confondente, il sodio, e la durata d'azione di tali resine.

## II.III.II. PRINCIPIO DI FUNZIONAMENTO DELLE RESINE A SCAMBIO IONICO

In generale, le resine a scambio ionico sostituiscono un particolare ione con uno di un altro tipo: durante il passaggio della soluzione attraverso la resina, questa trattiene lo ione bersaglio rilasciandone un altro.
Per le nostre misurazioni sono state utilizzate due diverse resine, rispettivamente con le seguenti funzioni:

- Catturare ioni $Na^+$ rilasciando $H^+$, con il conseguente abbassamento del pH;
- Normalizzare l'$H^+$ con $OH^-$, annullando la carica della soluzione e alzando il pH.

## II.III.III. STRUMENTAZIONE E MATERIALE

La strumentazione per gli esperimenti mediante resine a scambio ionico sono i medesimi delle precedenti esperienze, fatta eccezione per l'utilizzo di:

- Una colonna per l'eluizione;
- Cartine per la misurazione del pH;
- Un circuito non chiuso costituito dalla sola sonda a due elettrodi (ancora collegata all'analizzatore di impedenza), riempita di soluzione da analizzare da un capo mentre l'altro è stato sigillato: è stata adottata tale

Per quanto riguarda le sostanze si sono utilizzati 2 tipi di resine a scambio ionico:

- DOWEX G-26, H Form (SIGMA-ALDRIC)
- MTO-Dowex M43 Anion Exchange Resin (SUPELCO Analytical)

## II.III.IV. PROCEDURA

Anche durante queste prove ci sono state procedure comuni a tutte le esperienze:

- Le misure sono state eseguite nel range di frequenze 10 MHz – 1KHz, con 5 punti di misura per decade, in scala logaritmica;
- La colonna è stata riempita di volta in volta con 1 g della resina in esame e attraverso questa è stato fatto fluire mediante infusione costante (manuale, agendo su un percussore) un campione di 20 ml di soluzione: questo è stato poi sottoposto a misura di pH e iniettato nella sonda per la misura di impedenza;
- Le misurazioni successive sono state effettuate con quantità costante di resina ma con soluzione fresca, fino all'esaurirsi dell'azione scambiatrice della resina.

*Resina a eliminazione $Na^+$*

Eseguita una preliminare misura su un campione di fisiologica non filtrato dalla resina per avere un risultato di riferimento, sono stati compiuti poi i passaggi dei vari campioni attraverso la resina e le rispettive analisi.

*Resina a rilascio $OH^-$*

L'eluito prodotto passando di volta in volta 20 ml di soluzione fisiologica attraverso 1 g nuovo della prima resina, è stato fatto filtrare dalla seconda resina, utilizzando, di quest'ultima, sempre lo stesso campione. Anche in tal caso, è stata preliminarmente compiuta una misurazione del bianco, costituito dall'eluito non filtrato dalla seconda resina.

*Resina a eliminazione $Na^+$ con fisiologica caricata*

È stato ripetuto l'esperimento della resina a eliminazione di sodio, utilizzando però una soluzione di sodio cloruro all'1% anziché al 0,9%.

## II.III.V. RISULTATI

*Resina a eliminazione $Na^+$*

Il passaggio attraverso questa resina fa aumentare la conducibilità (pare cioè che a fronte di ciascun ione sodio eliminato vengano rilasciati diversi ioni idrogeno).

Comunque, si può notare come l'impedenza aumenti all'aumentare dei passaggi della soluzione sul medesimo campione di resina: ciò è atteso, causa l'esaurimento dell'azione della resina. Infatti, dopo un filtraggio all'incirca di 180 ml di soluzione, non si hanno più effetti apprezzabili da parte della resina.

*Resina a rilascio $OH^-$*

Come atteso, l'impedenza del filtrato cresce molto (vengono eliminati gli ioni idrogeno), ma verso la fine della batteria di esperimenti ci si avvicina notevolmente all'impedenza della soluzione di partenza (il filtrato della prima resina): anche in questo caso, come atteso, la resina tende ad esaurire il proprio potere di scambio.

## II.III.VI. CONSIDERAZIONI

Per questioni di confidenzialità, essendo lo studio delle resine un progetto ancora in corso presso l'ISIB-CNR, non è stato possibile includere figure in merito agli esperimenti preliminari indicati.

# *III. SIMULAZIONE DI MODELLI DI INTERESSE FISIOLOGICO*

# III.I. INTRODUZIONE

I modelli matematici sono largamente utilizzati per lo studio dei processi coinvolti nel metabolismo umano, per esempio nell'ambito della farmacocinetica, nello studio della cinetica dei traccianti nell'organismo e nella descrizione quantitativa della regolazione omeostatica del glucosio. Uno degli strumenti più utilizzati per la simulazione di tali modelli è l'ambiente di calcolo della MathWorks, Matlab. Grazie a questo software, i modelli biologici sono rappresentabili in diverse forme, tipicamente:

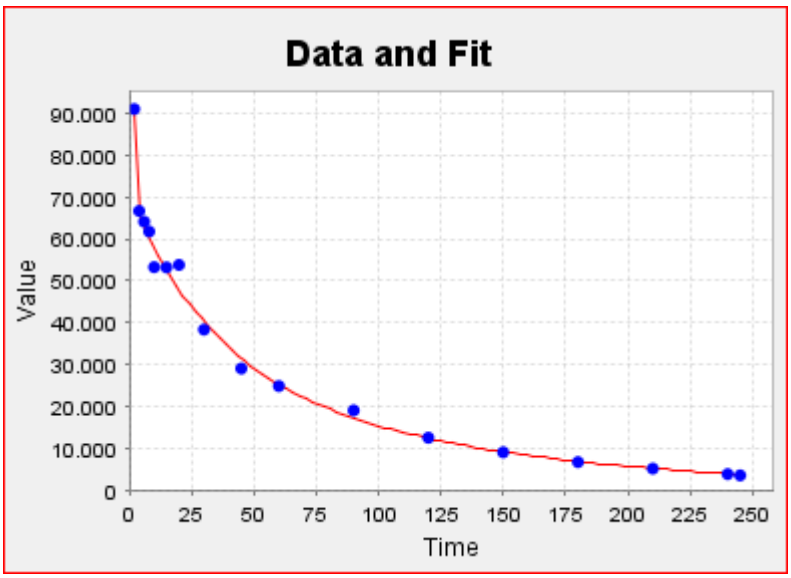
- In forma esplicita, per esempio mediante una funzione multiesponenziale;
- Con la "ODE Suite", un insieme di funzioni che permettono la soluzione di equazioni differenziali ordinarie utilizzando i metodi di base di Matlab;
- Utilizzando l'interfaccia grafica del simulatore annesso a Matlab, Simulink.

# III.II. CINETICA DI UN TRACCIANTE

Un tracciante è un atomo od una molecola che lascia una traccia misurabile, permettendo di ottenere informazioni sulla distribuzione nello spazio e sulla cinetica temporale di una sostanza (sostanza tracciata) con la quale il tracciante si mescola.
I traccianti si usano nello studio del metabolismo per quantificare la cinetica delle sostanze endogene, ad esempio del glucosio [1]. Un tipico esperimento prevede la somministrazione di un tracciante tramite un bolo (breve iniezione, idealizzabile secondo la teoria dei segnali con una delta di Dirac) in condizioni di equilibrio metabolico. In seguito al bolo, la concentrazione del tracciante decade nel corso del tempo secondo una legge che può essere approssimata tramite una funzione multiesponenziale. In altre parole, la cinetica di un tracciante può essere rappresentata con un sistema lineare invariante con risposta impulsiva multiesponenziale.
Nella prima parte di questo studio è stato considerato un esempio di cinetica di un tracciante derivato da dati reali. Una curva di scomparsa di un tracciante è stata analizzata con una funzione multiesponenziale e i parametri della funzione (ampiezze e esponenti) stimati dai dati (usando un semplice strumento di analisi ai minimi quadrati disponibile in rete [2]). La funzione multiesponenziale così determinata (rappresentabile semplicemente in Matlab) è stata usata come riferimento per le successive più articolate simulazioni basate su Matlab e Simulink.

La figura seguente illustra i dati originali e la corrispondente interpolazione multiesponenziale:

| Tempo (min) | Concentrazione |
|---|---|
| dose | 17650653 |
| 2 | 90946,62 |
| 4 | 66619,98 |
| 6 | 64089,28 |
| 8 | 61747,19 |
| 10 | 53223,1 |
| 15 | 53207,88 |
| 20 | 53759,92 |
| 30 | 38383,14 |
| 45 | 29115,16 |
| 60 | 24918,81 |
| 90 | 19074,96 |
| 120 | 12597,29 |
| 150 | 9115,577 |
| 180 | 6791,349 |
| 210 | 5209,287 |
| 240 | 3914,595 |
| 245 | 3601,508 |




Risultato del Multiexponential Data Fitting

La funzione multiesponenziale interpolante, normalizzata alla dose, è:

$0{,}021e^{-1{,}4t} + 0{,}002e^{-0{,}03t} + 0{,}002e^{-0{,}009t}$

## III.II.I. IL MODELLO LINEARE ESPRESSO IN FORMA DI STATO E IL PROBLEMA DEGLI INGRESSI

La funzione multiesponenziale del problema illustrato si può esprimere, indipendentemente dal metodo di simulazione utilizzato, sottoforma di un modello lineare in forma di stato, descritto dalle equazioni differenziali:

$$\begin{aligned} dx/dt &= Ax + Bu, \\ y &= Cx; \\ x(0) &= x_0 . \end{aligned} \qquad (1)$$

In tale modello, u indica un eventuale ingresso (nelle simulazioni qui descritte che prevedono un input esso è costituito da un impulso rettangolare), x lo stato del sistema, y i valori in uscita, mentre le tre matrici A, B, C vengono rappresentate in forma di Jordan a partire dalle ampiezze e dagli esponenti della funzione multiesponenziale. Da notare che, rispetto alla teoria generale, non è

presente il termine Du nell'espressione dell'output, dal momento che per motivi fisici non c'è un legame diretto ingresso - uscita.

In Simulink, rappresentare il modello (1) è molto semplice, grazie all'apposito blocco "State – Space", che sarà descritto in seguito. Questo blocco prevede esplicitamente un ingresso, al quale si può collegare una sorgente di segnale quale, ad esempio, un generatore di impulsi.
In Matlab non esistono funzionalità specifiche che consentano di definire in modo esplicito un segnale di ingresso. Le equazioni differenziali sono, infatti, descritte tramite una funzione m, che deve anche contenere una descrizione degli ingressi. Ciò crea delle difficoltà aggiuntive rispetto a Simulink; alcune soluzioni sono presentate e discusse in seguito.

## III.II.II. METODI DI SIMULAZIONE

La simulazione dell'andamento della concentrazione del tracciante somministrato tramite bolo può essere affrontata secondo due approcci, a seconda che si rappresenti il bolo come uno stato iniziale non nullo ($x_0$=Bk, dove k è la dose del bolo) o come un ingresso rettangolare di breve durata e di area uguale alla dose del bolo (e quindi di ampiezza k/T, dove T è la durata).

- Nel primo caso si considera l'evoluzione libera, in altre parole la risposta del sistema se questo non viene sollecitato da alcun ingresso ma evolve solamente a partire dal suo stato iniziale.
- Nel secondo caso si considera risposta forzata, ossia l'uscita del modello con stato iniziale nullo e avente come segnale d'ingresso un impulso rettangolare.

Entrambi questi metodi possono essere realizzati tanto in Simulink quanto in Matlab con la suite ODE.

### *Ode Suite*

La libreria di Matlab mette a disposizione la suite ODE [3] per la soluzione di numerosi tipi di equazioni differenziali. Tale suite è composta da un insieme di risolutori numerici (ad ed. Runge-Kutta, Gear), che richiedono una definizione standardizzata delle equazioni differenziali da risolvere.
La sintassi per l'utilizzo di questi risolutori è la seguente:
`[T,X] = solver(odefun,tspan,x0,options),`
dove:

- T è un vettore colonna degli istanti di tempo in cui il solutore calcola la soluzione;

- X è una matrice in cui ogni riga contiene il vettore di stato relativo all'istante restituito nella corrispondente riga di T;
- Solver è il particolare risolutore usato (nelle mie simulazioni ho usato esclusivamente la funzione "ode45");
- Odefun è un "function handle", un riferimento a una funzione esterna (un .m file) che restituisce in output il lato destro dell'equazione differenziale, espressa secondo la struttura dx/dt = … (vedi in seguito);
- Tspan è un vettore che specifica l'intervallo di integrazione [t0 tf], il cui primo elemento è quello a cui il risolutore impone le eventuali condizioni iniziali contenute in y0;
- Options è una variabile struttura di parametri opzionali che modificano le opzioni standard di integrazione.

È importante notare come all'interno di questa struttura non siano esplicitamente previsti segnali di ingresso. Nasce, infatti, proprio da qui la necessità degli espedienti sopra menzionati per poter gestire l'analisi in Matlab di un sistema di cui si considera la sola risposta forzata.

*Simulink*

Simulink è un ambiente di simulazione a blocchi compreso in Matlab che rende decisamente più agevole l'approccio al problema del tracciante grazie a blocchi appositamente dedicati. In particolare, il blocco "State – Space" permette di simulare un sistema lineare espresso in forma di stato fornendo come parametri, mediante l'opportuna finestra di configurazione, le quattro matrici e gli eventuali stati iniziali. A questo blocco è possibile collegare un'eventuale sorgente di segnale in input, come il blocco "Step" o il "Pulse Generator", oltre a uno "Scope" in output che permette di ottenere un grafico dell'uscita y del sistema.

## III.II.III. DESCRIZIONE METODI

*Risposta Libera In ODE Suite*

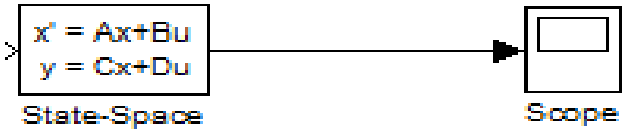
Per risolvere il sistema (1) con la ODE suite ho realizzato la funzione m `solve_stateeq` riportata sotto. Questa funzione contiene la funzione dipendente `stateeq` che definisce il sistema (1) nel caso di assenza di ingresso (u=0). Si noti come la funzione rappresenti il membro di destra dell'equazione differenziale (1) (xdot corrisponde a dx/dt). La funzione `stateeq` è passata come function handle (`@stateeq`) al solutore ode45. Poiché una funzione

dipendente eredita le variabili dalla funzione che la contiene, la matrice A nella funzione `stateeq` è quella che è passata come argomento alla funzione `solve_stateeq`. Si noti anche come la descrizione del modello (1) è completata dalla definizione dello stato iniziale (2ª riga di `solve_stateeq`) e dell'uscita (4ª riga di `solve_stateeq`).
Per la simulazione numerica, le matrici A e C in forma di Jordan sono state definite usando i parametri di riferimento derivati dalla funzione multiesponenziale.

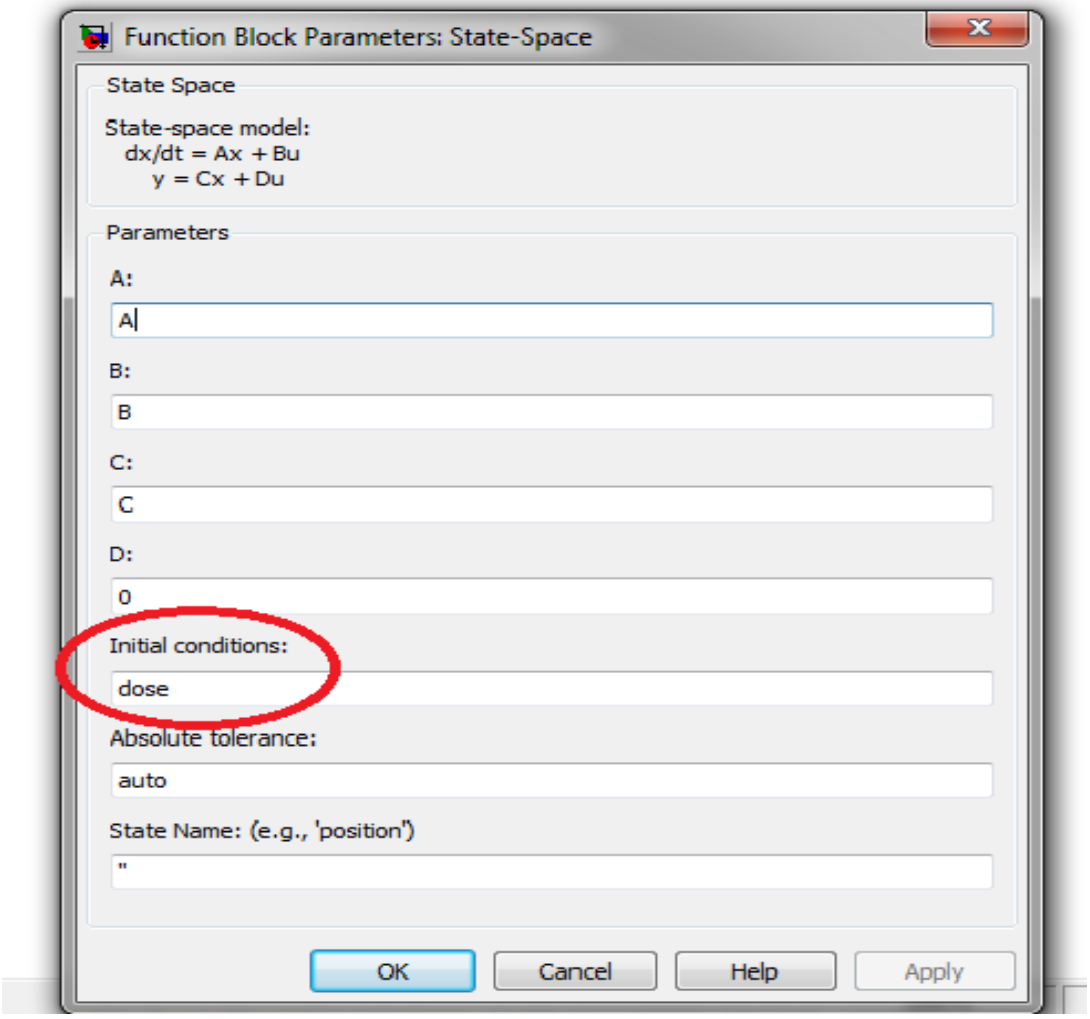
```
function [t,y] = solve_stateeq(A,C,tspan,instate)
x0 = instate*ones(size(A,1),1);
[t,x] = ode45(@stateeq,tspan,x0);
y = C * x';

    function xdot = stateeq(t,x)
    xdot = A*x;
    end

end .
```

*Risposta Libera In Simulink*

Il modello Simulink che realizza l'evoluzione del sistema a partire dal solo stato iniziale è illustrato nella figura successiva. La sua struttura è molto semplice, in quanto Simulink dispone di un blocco specifico per la descrizione di sistemi in forma di stato.




In questo modello Simulink, si imposta il campo "Initial conditions", tra i parametri del blocco State – Space, a "dose", variabile del workspace di Matlab che contiene il valore della quantità di tracciante iniettata (il bolo), lasciando la porta di ingresso priva di collegamenti: questa operazione è possibile poiché Simulink ricerca automaticamente all'interno del workspace di Matlab eventuali parametri espressi mediante il nome di una variabile.




*Evoluzione Forzata In Matlab*

Nel considerare l'evoluzione forzata del sistema in Matlab ci si scontra con l'assenza di strumenti per l'utilizzo esplicito di ingressi con la suite ODE. Il problema è stato affrontato in due modi:

Nel primo metodo, l'impulso rettangolare è stato definito mediante struttura if/else. Questa struttura è una copia dell'analoga descrizione matematica, ma, come si vedrà, non è adeguata per la simulazione numerica. La funzione `stateeq` che rappresenta il sistema (1) è stata modificata inserendo il termine di ingresso Bu e definendo u con il costrutto if/else. Si noti che in questo caso lo stato iniziale è zero.
La soluzione di seguito illustrata presenta la particolarità del passaggio dei parametri in cascata. La variabile ***dose***, argomento della funzione `rispImpRett` è vista dalla funzione `stateeq` in quanto la funzione `stateeq` è contenuta in `rispImpRett` (si notino i costrutti `function/end` inclusi uno

nell'altro). In questo modo variabile ***dose*** giunge fino alla funzione interna `stateeq` dove viene utilizzata per la creazione dell'impulso rettangolare.

```
function [t,y] = rispImpRett(A,C,T,tspan,dose)
B = ones(size(A,1),1);
x0 = zeros(size(B));
[t,x] = ode45(@stateeq,tspan,x0);
y = C * x';

    function xdot = stateeq(t,x)
        if t>=0 && t<=T
            u = dose/T;
        else
            u = 0;
        end;
        xdot = A*x + B*u;
    end

end .
```

Con il secondo metodo, la simulazione è divisa in due intervalli: da 0 a T e da T all'instante finale (`tend`). In entrambi gli intervalli l'ingresso u è costante (dose/T nel primo intervallo e 0 nel secondo). Si noti come questi valori sono definiti prima della chiamata a `ode45` e sono visibili alla funzione `stateeq` in quanto funzione dipendente (come per A e C).
Da notare come lo stato iniziale della seconda chiamata di `ode45` sia uguale allo stato finale della prima chiamata (`x1(end,:)'`) e come, infine, si uniscano i vettori dei tempi e degli stati ottenuti.

```
function [t,y] = rispImpRett2(A,C,T,tend,in,dose)
B = ones(size(A,1),1);
x0 = zeros(size(B));
u = dose/T;
[t1,x1] = ode45(@stateeq,[0 T],x0);
u = 0;
[t2,x2] = ode45(@stateeq,[T tend],x1(end,:)');
t = [t1;t2(2:end)];
x = [x1;x2(2:end,:)];
y = C * x';

    function xdot = stateeq(t,x)
        xdot = A*x + B*u;
    end

end .
```

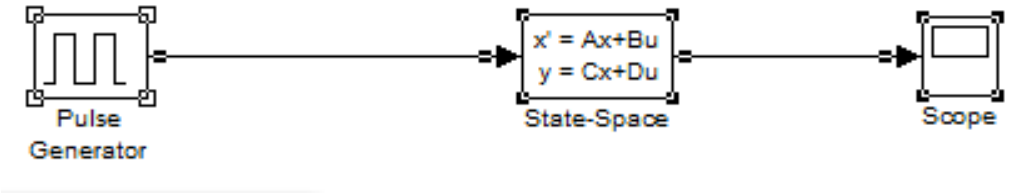
Si noti come la funzione interna, “stateeq”, abbia accesso alla variabile u, contenente il valore dell’ingresso per il meccanismo del passaggio dei parametri in cascata descritto sopra.

Una generalizzazione di un ingresso impulsivo rettangolare, come quello fin qui considerato, è basato sull’uso di una *lookup table*. I valori di un ingresso di questo tipo si determinano come interpolazione (per es. costante o lineare a tratti) da una coppia tempo-valore. Matlab dispone di una funzione interpolatrice, interp1, che è adatta a questo scopo. La struttura della funzione di simulazione è in questo caso:

```
function [t,y] = eqdiff(A,C,tspan,tref,uval)
B = ones(size(A,1),1);
x0 = zeros(size(B));
[t,x] = ode15s(@stateeq,tspan,x0);
y = C * x';

    function xdot = stateeq(t,x)
    xdot = A*x + B*interp1(tref,uval,t,'linear',0);
    end

end .
```

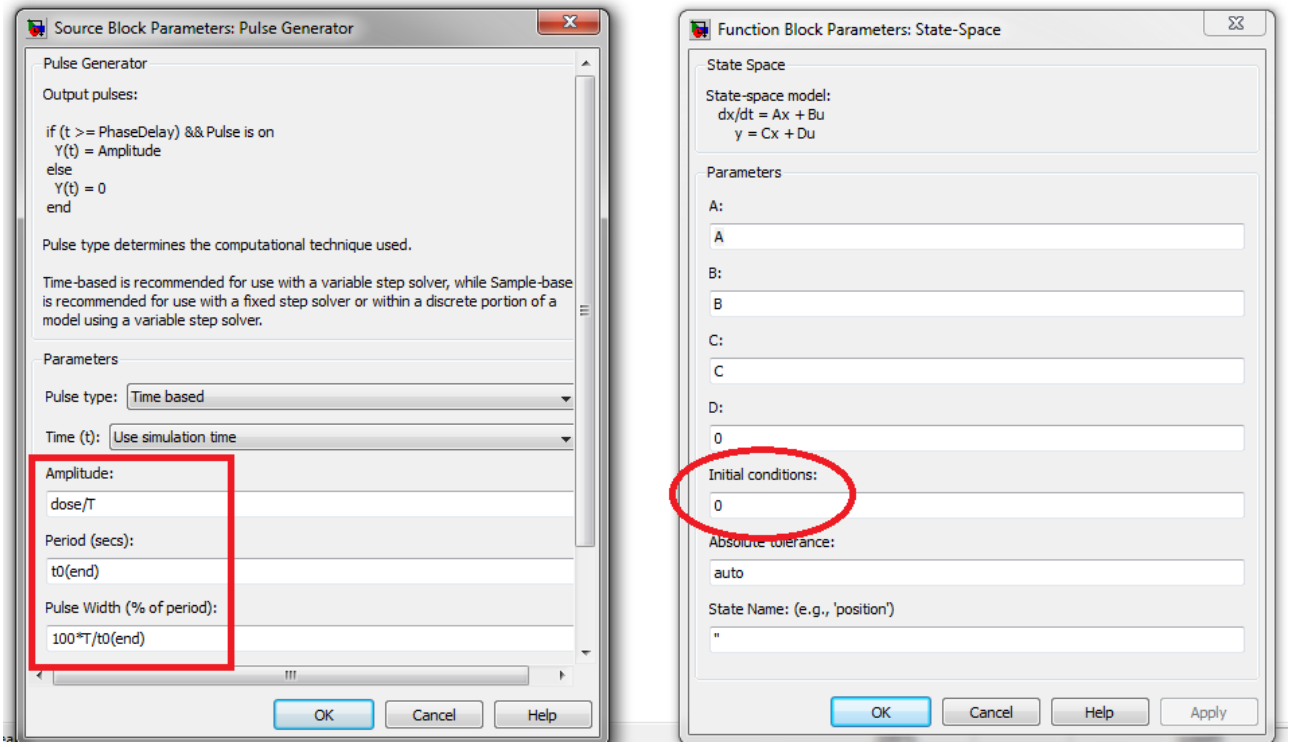
*Evoluzione Forzata In Simulink*

Il modello Simulink per la simulazione dell’iniezione del tracciante come un impulso rettangolare è visualizzato nella figura seguente:




Per la simulazione dell’evoluzione forzata in Simulink il campo Initial conditions del blocco State – Space viene posto a zero e la sua porta di input viene collegata ad un Pulse Generator. Il Pulse Generator è concepito per produrre un impulso periodico; per la simulazione dell’impulso rettangolare è quindi necessario definire una lunghezza del periodo non inferiore all’intervallo di simulazione. Il periodo è stato posto uguale all’intervallo di simulazione, che va da 0 a t0(end) (vedi figura successiva). L’ampiezza dell’impulso è stata impostata al valore del rapporto tra la dose e la durata dell’iniezione (dose/T);

la durata deve essere espressa in percentuale del periodo ed è quindi stata impostata a 100*T/t0(end).



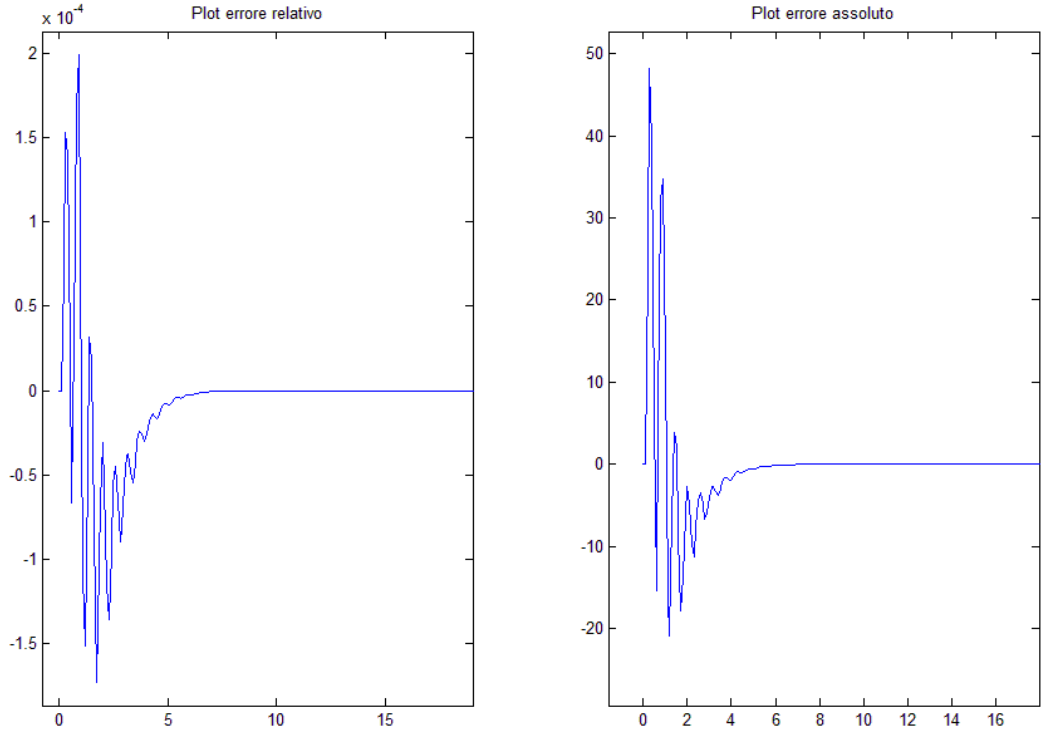

## III.II.IV. CONFRONTI

Una volta realizzati questi diversi metodi di simulazione, si è proceduto ad effettuare vari confronti tra i differenti risultati ottenuti.

*Multiesponenziale ed Evoluzione Libera In Matlab*

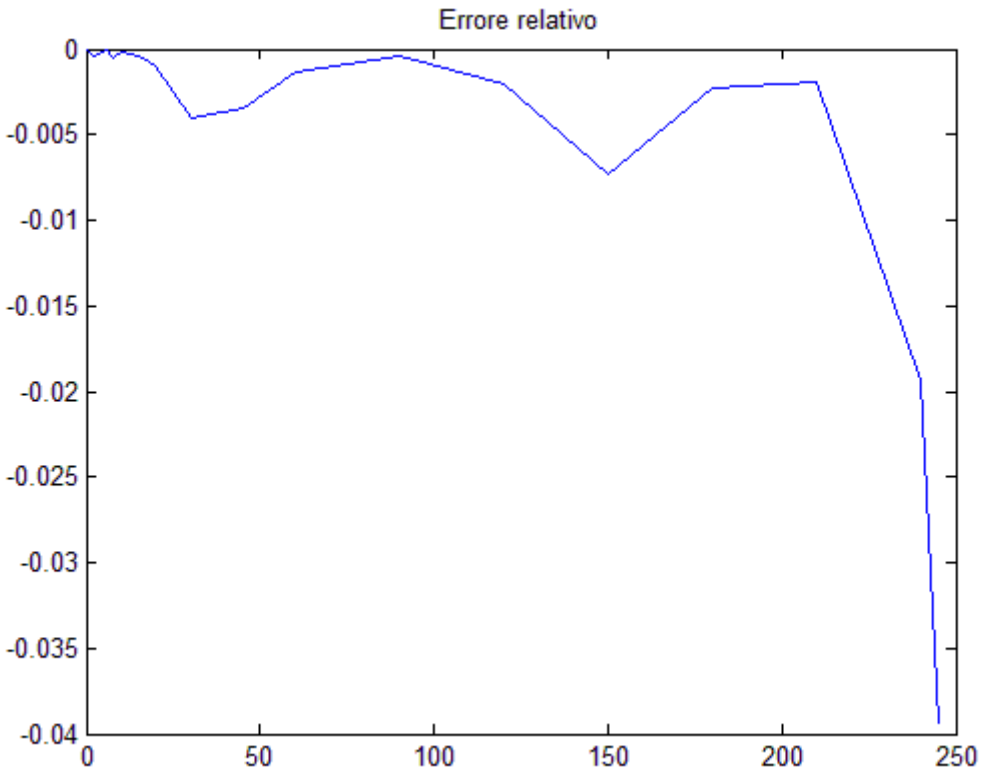
In una prima fase, sono state confrontate le simulazioni mediante funzione multiesponenziale e sistema lineare in forma di stato simulato con la ODE suite. Pur essendo matematicamente equivalenti, queste rappresentazioni non danno gli stessi risultati, com'è atteso dalla diversa precisione numerica. Gli errori sono però relativamente piccoli: mentre l'errore assoluto raggiunge un massimo dell'ordine delle decine, il massimo dell'errore relativo calcolato rispetto alla somma di esponenziali è di circa $2 \times 10^{-4}$. L'errore numerico di calcolo della funzione multiesponenziale si può ritenere non lontano dalla precisione macchina (variabile `eps` di Matlab, dell'ordine di $10^{-16}$ nell'ambiente di calcolo adottato). Per la ODE Suite invece, l'errore numerico è considerevolmente più grande; i parametri tipici di tolleranza del simulatore adottato prevedono un errore relativo di circa $1 \times 10^{-3}$.
Da notare, inoltre, come errore relativo ed assoluto abbiano un andamento simile, con rapide oscillazioni in prossimità dei primi secondi di infusione, per

poi attestarsi in prossimità dello zero (~ -1,5 x $10^{-10}$) attorno ai dieci secondi fino alla fine della simulazione.



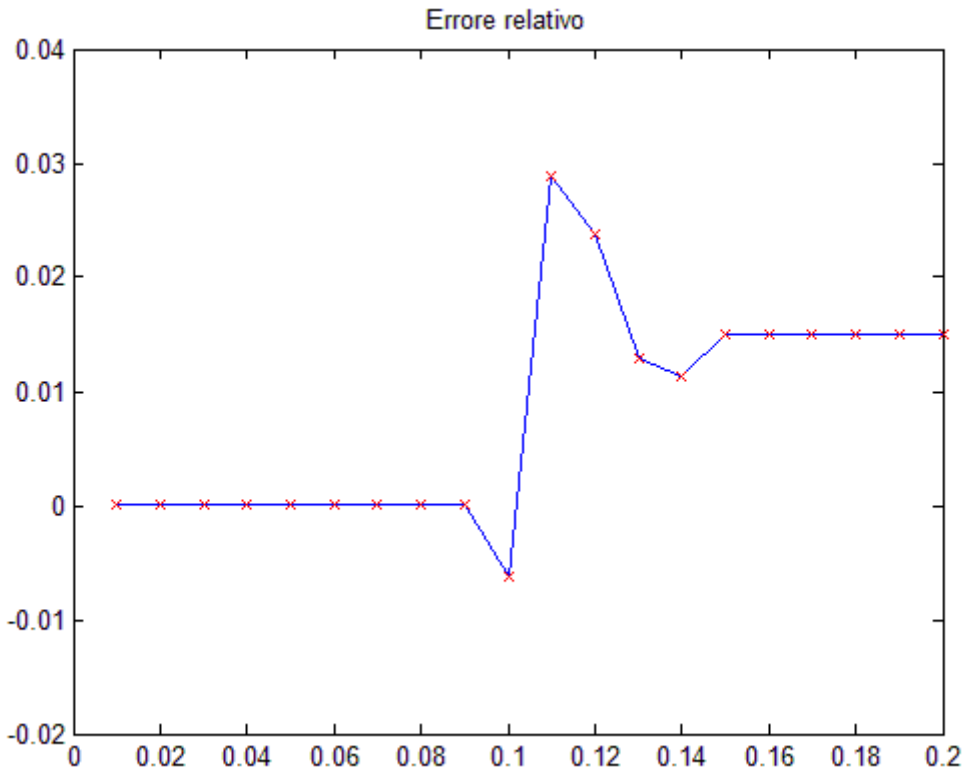

*Multiesponenziale Vs Risposta Libera In Simulink*

Confrontando la funzione multiesponenziale con la risposta libera in Simulink, si ottiene un errore relativo, calcolato in riferimento alla funzione analitica, che oscilla per quasi tutta la durata della simulazione fino a raggiungere il suo valore massimo in modulo nella fase finale, pari al 3,5%: questo può essere considerato l'errore di integrazione dovuto alle tolleranze impostate di default per i risolutori si Simulink (1 x $10^{-6}$ di tolleranza assoluta).



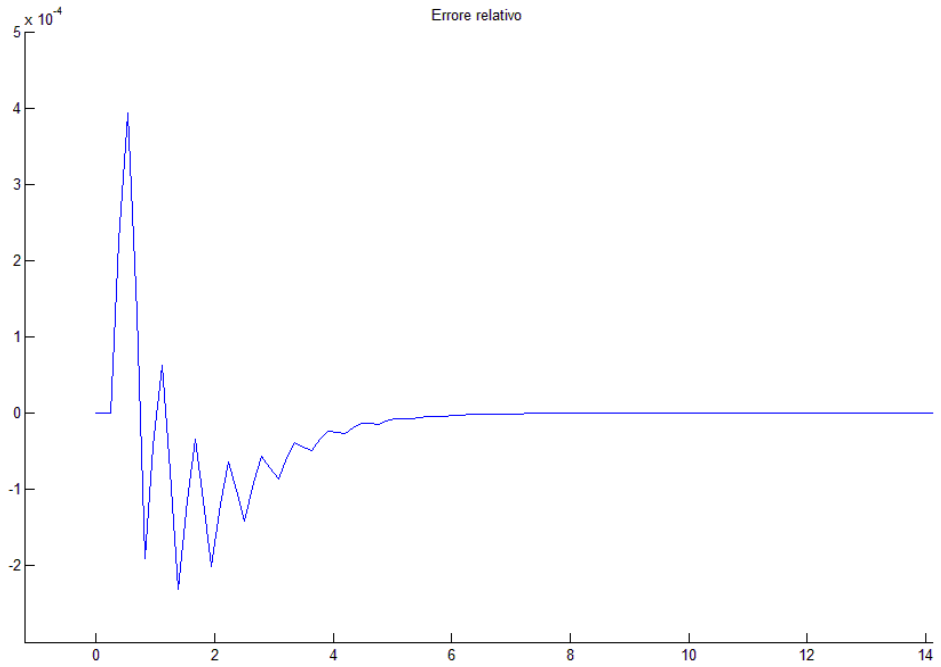

*Risposta Forzata: Simulink Vs If/Else In Matlab*

Sono stati successivamente analizzati i risultati ottenuti dalla sollecitazione del sistema con un impulso rispettivamente in Simulink e in Matlab, utilizzando nel caso di quest'ultimo la versione che realizza l'impulso mediante il costrutto if/else: si ottiene un errore relativo rispetto al modello a blocchi quasi del 3%, in seguito alla sollecitazione da parte dell'impulso (che agisce tra 0 e 0,1). Fonte di questo errore notevole è l'imprecisione del costrutto if/else, dal momento che il risolutore ODE, procedendo per passi discreti, non può accorgersi del rapido salto dell'ingresso alla fine del bolo, ma procede assumendo una sua variazione più continua. Esauritosi l'ingresso impulsivo, l'errore relativo si attesta attorno ad un valore ancora relativamente alto (0,015 o 1,5%). In altre parole, l'effetto dell'imprecisione del costrutto if/else si propaga per tutta la simulazione.




*Risposta Forzata: Simulink Vs Simulazione "Spezzata" In Matlab*

Dal confronto tra la risposta forzata in Simulink e la versione Matlab basata divisione dell'intervallo di simulazione in due periodi, emerge un errore relativo calcolato in relazione al risultato di Simulink molto più contenuto rispetto all'analisi precedente, dell'ordine del $10^{-4}$. Pertanto, è evidente come la metodologia migliore per studiare l'analisi forzata in Matlab sia il troncamento della simulazione in prossimità del termine dell'impulso rettangolare.

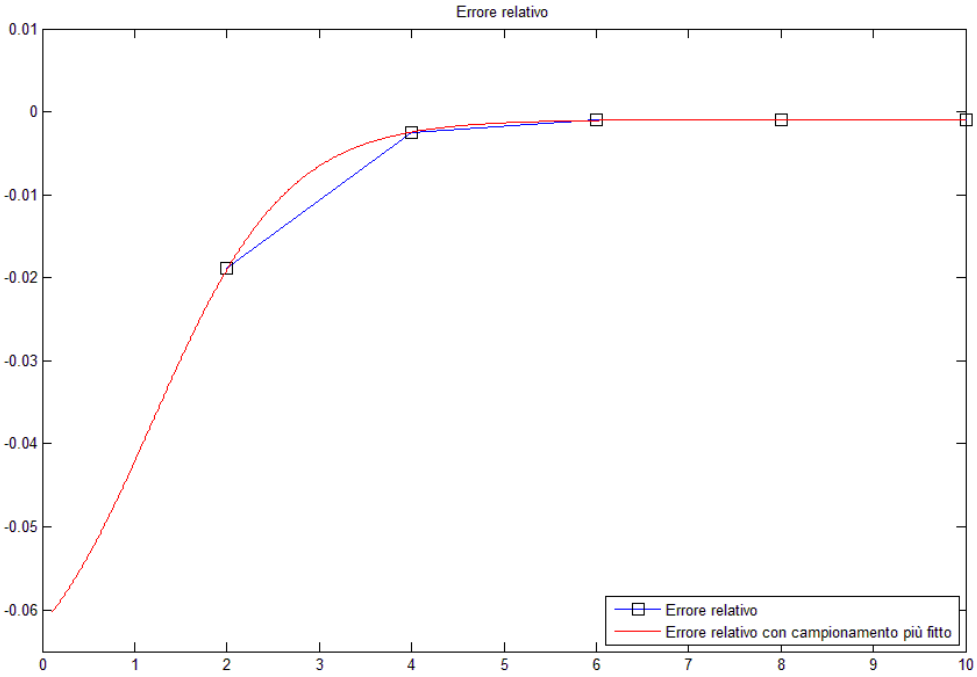



Per semplicità, è stato omesso il confronto tra la risposta forzata in Simulink e la risposta analitica ottenibile dalla funzione multiesponenziale.

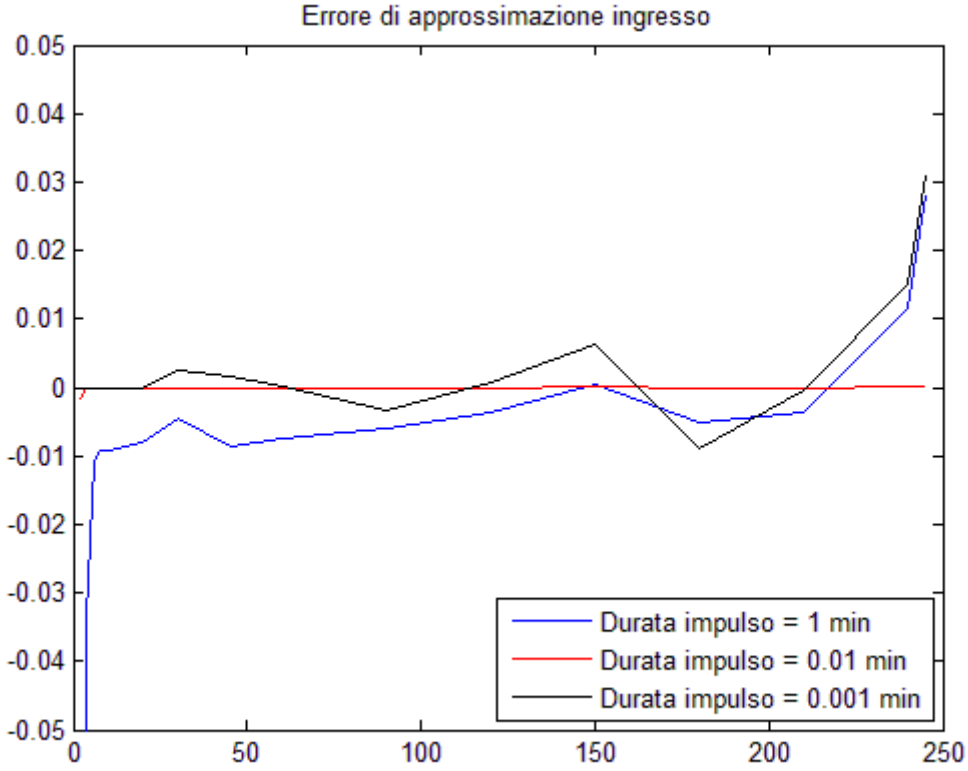
*Evoluzione Libera Vs Risposta Forzata In Simulink*

Il confronto tra risposta libera e forzata chiarisce l'entità della differenza dovuta alla rappresentazione del bolo di tracciante con i due modelli: stato iniziale non nullo ed evoluzione libera o impulso rettangolare ed evoluzione forzata. Questi due modelli sono concettualmente diversi e quindi le differenze non sono da attribuirsi alla precisione numerica. In particolare, il modello ad evoluzione libera ha una risposta monotona decrescente, mentre in quello ad evoluzione forzata ha una risposta crescente per tutta la durata dell'impulso rettangolare (T, espresso in minuti) e successivamente decrescente. Conseguentemente, ha senso confrontare le differenze solo per istanti di tempo maggiori di T.
I confronti sono stati realizzati per semplicità solo in Simulink. Poiché la durata dell'impulso influenza la differenza (minore è T più simile è l'impulso ad una delta di Dirac), le simulazioni sono state ripetute per vari valori di T.

Confrontando i due modelli per un bolo di 0,1 minuti, si ha un errore relativo il cui andamento è notevolmente influenzati dagli istanti di campionamento: per un campionamento più fitto, si ha una curva più regolare e uniforme, che parte immediatamente dopo il termine dell'impulso, a differenza del grafico relativo a un campionamento più rado, in cui viene eseguita un'interpolazione lineare tra i vari punti.




Dal grafico dell'errore relativo in funzione della durata dell'impulso, emerge come la simulazione per il bolo da 0,01 min sia la più vicina al modello con stato iniziale non nullo, mentre per gli altri due valori, rispettivamente 1 e 0,001 min, si hanno delle curve irregolari ma simili, che arrivano fino a un errore relativo del 3%.



# III.III. SIMULAZIONE DI MODELLI CODIFICATI IN SBML

Successivamente all'introduzione dei metodi di base per la simulazione di modelli in Matlab e Simulink è stato affrontato un tema più complesso: l'utilizzo con questi strumenti per la simulazione di modelli provenienti da una libreria standardizzata e codificati nel linguaggio di mark-up SBML (Systems Biology Markup Language).

## III.III.I. LIBRERIA DI MODELLI SBML E CONVERSIONE IN MATLAB E SIMULINK

La libreria "BioModels Database" dell'European Bioinformatics Institute (EBI) raccoglie modelli matematici di interesse biologico, nell'ambito della disciplina della Systems Biology. I modelli sono redatti in un linguaggio di mark-up SBML, basato su XML, che permette la descrizione qualitativa e quantitativa di reazioni biochimiche (espresse mediante equazioni differenziali secondo la teoria dei compatimenti). SBML è stato creato con il proposito di consentire l'utilizzo dei modelli con svariati software senza dover riscrivere i codici e di permetterne la pubblicazione e la condivisione. L'SBML è stato pensato e realizzato come "un formato comune intermedio – una lingua franca – che consente la comunicazione degli aspetti più essenziali dei modelli" [4].
Per poter utilizzare questo formato con Matlab, è necessario installare l'"SBML Toolbox", disponibile sul sito sbml.org. Il toolbox è basato sulla libSBML, una libreria gratuita per gestire il formato SBML e tradurlo in codice appropriato alle applicazioni in uso, e fornisce un set di funzioni di base che permettono l'utilizzo di modelli SBML in Matlab. Questo toolbox è uno strumento gratuito che realizza le funzioni fondamentali del toolbox proprietario di Matlab "SimBiology" [5].
Il passo fondamentale per la simulazione di un modello SBML in Matlab è la sua conversione in una struttura Matlab chiamata Matlab_SBML, che contiene gli stessi campi presenti all'interno del file SBML. Questa struttura si ottiene con la funzione "TranslateSBML" del toolbox.
Altre funzioni del toolbox, fondamentali per la simulazione, sono: "DispalyODEFunction", che restituisce un grafico del risultato dei risolutori ODE applicati alle equazioni del modello, e "WriteODEFunction", che permette di generare una funzione m contenente le equazioni differenziali del modello a partire dalla struttura in cui esso è stato salvato durante

l'importazione. Un esempio di funzione m generata da "WriteODEFunction" è il seguente:

```
function xdot = Westermark2003_Pancreatic_GlycOsc_basic(time,
x_values)
% function Westermark2003_Pancreatic_GlycOsc_basic takes
%
% either    1) no arguments
%               and returns a vector of the initial species concentrations
%
% or        2) time - the elapsed time since the beginning of the reactions
%              x_values    - vector of the current concentrations of the species
%               and returns a vector of the rate of change of concentration of
each of the species
[…]
The species in this model are related to the output vectors with the following
indices
%   Index   Species name
%     1       GLC
%     2       G6P_F6P
%     3       F6P
%     4       FBP
%     5       G3P
[…]
% species concentration rate equations
xdot(1) = 0;

xdot(2) = ( +
(comp*Vgk*power(GLC/Sgk,hGK)/(1+power(GLC/Sgk,hGK))) -
(comp*Vpfk*power(F6P/Spfk,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))/(power(F6P/Spfk,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))+(1+power(FBP/Xpfk,hx))/(1+powe
r(alpha,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))*power(FBP/Xpfk,hx)))))/comp;

xdot(3) = xdot(2)*KeqGPI/(1+KeqGPI);

xdot(4) = ( + (comp*Vpfk*power(F6P/Spfk,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))/(power(F6P/Spfk,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))+(1+power(FBP/Xpfk,hx))/(1+powe
r(alpha,hpfk-(hpfk-
hact)*(FBP/Sfba/(1+FBP/Sfba)))*power(FBP/Xpfk,hx)))) -
(comp*Vfba*(FBP/Sfba)/(FBP/Sfba+1)))/comp;

xdot(5) = 0;
[…]
```

La funzione generata da "WriteODEFunction" risulta di particolare importanza dal momento che costituisce la funzione a cui applicare gli ODE solvers in fase di simulazione; inoltre, invocandola priva di argomenti, esse restituisce le

concentrazioni iniziali di tutte le specie coinvolte nelle reazione (ossia lo stato iniziale):

```
function xdot = Westermark2003_Pancreatic_GlycOsc_basic(time,
x_values)
[…]
if (nargin == 0)

    % initial time
    time = 0;

    % initial concentrations
    GLC = 10;
    G6P_F6P = 3.71728;
    F6P = 0.857834;
    FBP = 0.00063612;
    G3P = 0;
[…] .
```

Il parametro time, pur essendo esplicitamente tra gli argomenti della funzione, non viene in realtà mai utilizzato nel corpo delle equazioni differenziali di questo specifico modello.

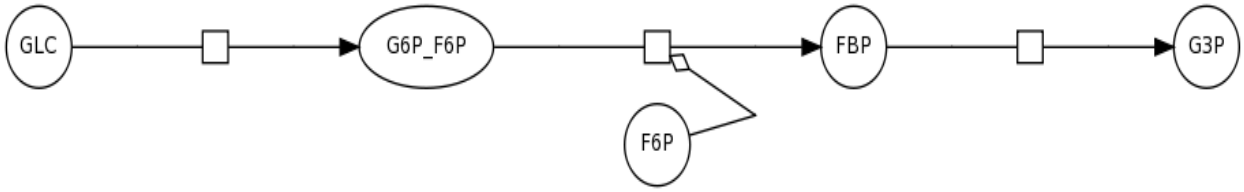
Va precisato che, per utilizzare “WriteODEFunction”, è necessario applicare preliminarmente alla struttura contenente il modello un’ulteriore funzione (“adjustVaryingParameters”), fornita separatamente come complemento al toolbox, al fine di aggirare un problema di progettazione del toolbox.

Nell’ambito di questo studio si è mostrato come utilizzare l’SBML Toolbox per due scopi: 1) modificare le equazioni originali del modello in modo da introdurre un ingresso, non presente nel modello di libreria; 2) realizzare il modello in Simulink con un costrutto veloce e relativamente sicuro. Il materiale presentato nell’introduzione è alla base di queste realizzazioni.

### III.III.II. MODELLO SBML DI WESTERMARK

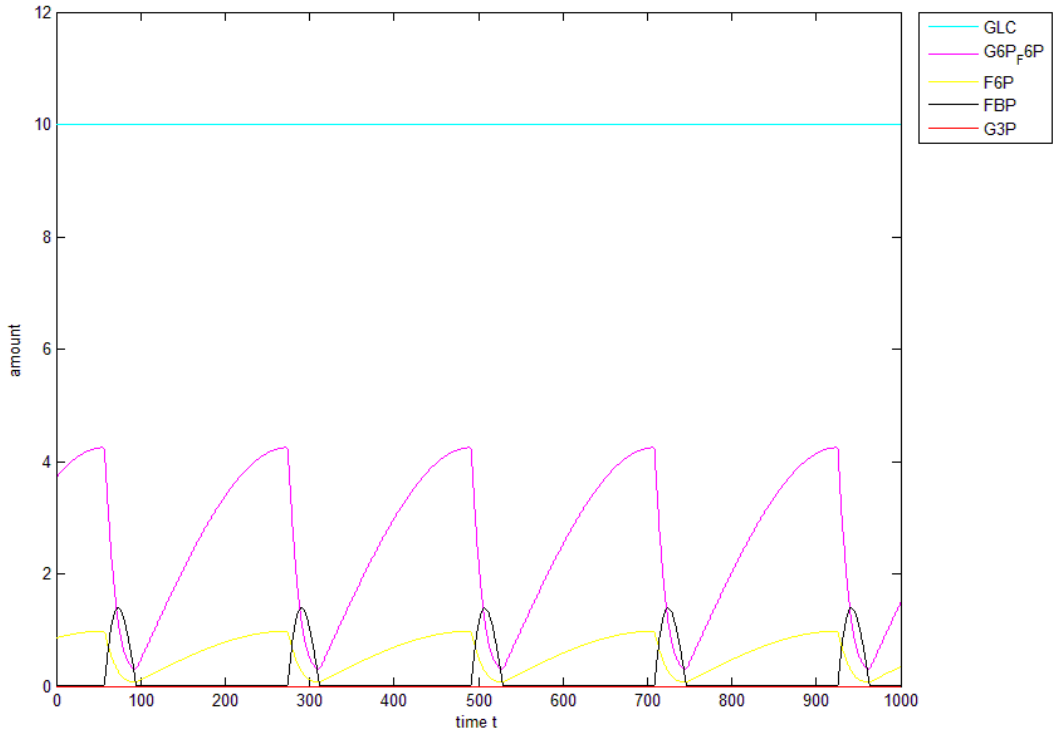
Il modello preso in considerazione nelle mie analisi e simulazioni è stato il modello di Westermark [6,7] che descrive una delle fasi iniziali della glicolisi all’interno delle beta-cellule del pancreas, durante la quale avvengono una serie di reazioni enzimatiche a catena che coinvolgono vari enzimi, dalla glucochinasi alla GADP (gliceraldeide-3-fosfato idrogenasi): il glucosio (GLC) si trasforma in glucosio 6-fosfato - fruttosio 6-fosfato (G6P_F6P), che a sua volta, mediante il modificatore fruttosio 6-fosfato (F6P), diventa fruttosio 1,6-

bifosfato (FBP) dal quale si ottiene, infine, gliceraldeide 3-fosfato (G3P), secondo lo schema seguente:




*Simulazione Standard*

La simulazione standardard di un modello SBML con l'SBML Toolbox si ottiene con la funzione "DispalyODEFunction". L'andamento nel tempo della concentrazione di tutte le specie coinvolte nelle reazioni è rappresentato nella figura seguente:

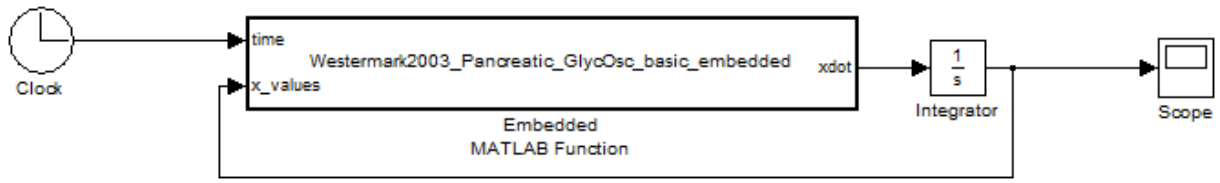



*Sbml In Simulink*

Il metodo usato per realizzare in Simulink un modello SBML senza doverlo ricodificare è basato sulla classica rappresentazione delle equazioni differenziali in forma di stato (estensione del modello (1)):

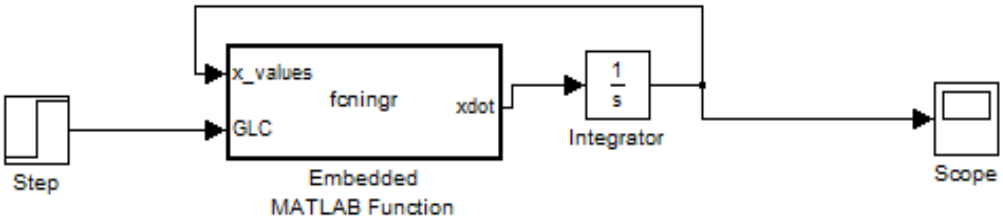
$dx/dt = f(x,t); x(0) = x_0.$ (2)

Poiché l'SBML Toolbox fornisce una descrizione di f(x,t) come funzione m, è sufficiente rappresentare il modello (2) con il classico sistema in retroazione usato in Simulink per descrivere le equazioni differenziali:

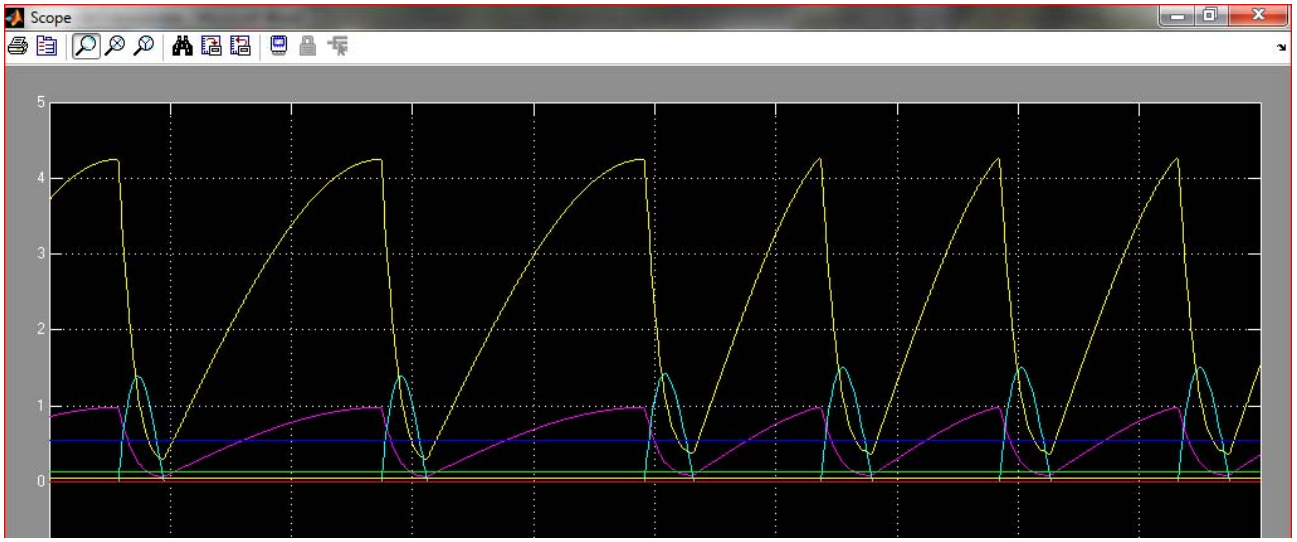



Si noti che in questo schema la f(x,t) è realizzata con un blocco "Embedded Matlab Function" che permette di includere una funzione Matlab all'interno di un modello Simulink. Il codice di questa funzione si ottiene con semplici adattamenti della funzione m prodotta da "WriteODEFunction".
Nello schema, l'integratore è vettorizzato, cioè integra simultaneamente tutti gli elementi del vettore di stato x. Le condizioni iniziali sono le concentrazioni delle specie chimiche prima della simulazione, ottenute invocando senza argomenti la funzione delle equazioni del modello, come già detto precedentemente: il campo "Initial conditions" dell'integratore viene quindi riempito con il nome di tale funzione, privo di parametri.
Il blocco "Clock" è un elemento fittizio per fornire alla funzione integrata il parametro time che è uno dei suoi argomenti, in quanto la funzione è definita secondo il modello (2). In realtà il tempo non è utilizzato nel corpo della funzione e potrebbe essere eliminato (il tempo è il primo argomento di ingresso della funzione).
Si noti che una struttura di questo tipo permette di creare un modello Simulink dalla funzione m derivata direttamente dalla descrizione SBML (con "WriteODEFunction") quasi con un semplice "copia e incolla". Pertanto, l'integrazione della libreria SBML con Simulink non è particolarmente più complicata rispetto a quella con la ODE suite di Matlab.

L'introduzione in un ingresso (un gradino di glucosio al tempo 500, con concentrazione che passa dal valore originale 10 a 20 mmol/L) richiede invece il riadattamento della funzione del blocco Embedded. In particolare, l'argomento rappresentate il tempo è stato sostituito con un argomento che definisce l'ingresso. Ciò porta allo schema seguente, dove il blocco Step rappresenta il glucosio, portato da 10 a 20 mmol/L al tempo 500:




Il risultato della simulazione è illustrato nella figura seguente. Da notare come il periodo delle curve delle concentrazioni diminuisca a partire dall'istante di innalzamento della concentrazione di glucosio (a metà della figura), e come le curve di glucosio 6-fosfato - fruttosio 6-fosfato (linea gialla) diventino più appuntite.




*Sbml In Matlab*

Le medesime simulazioni si possono effettuare in Matlab, sfruttando il file con le equazioni differenziali del modello Westermark: per simulare il modello senza ingressi, è sufficiente invocare il solutore ODE per le equazioni delle reazioni per ottenere il grafico delle concentrazioni:

```
function [t,x] = WestermarkODE(tend)
ImpostazioneTolleranzeWestermark;
 [t,x] = ode45(@Westermark2003_Pancreatic_GlycOsc_basic, [0, tend], InitConds, options);
plot(t,x);
end .
```

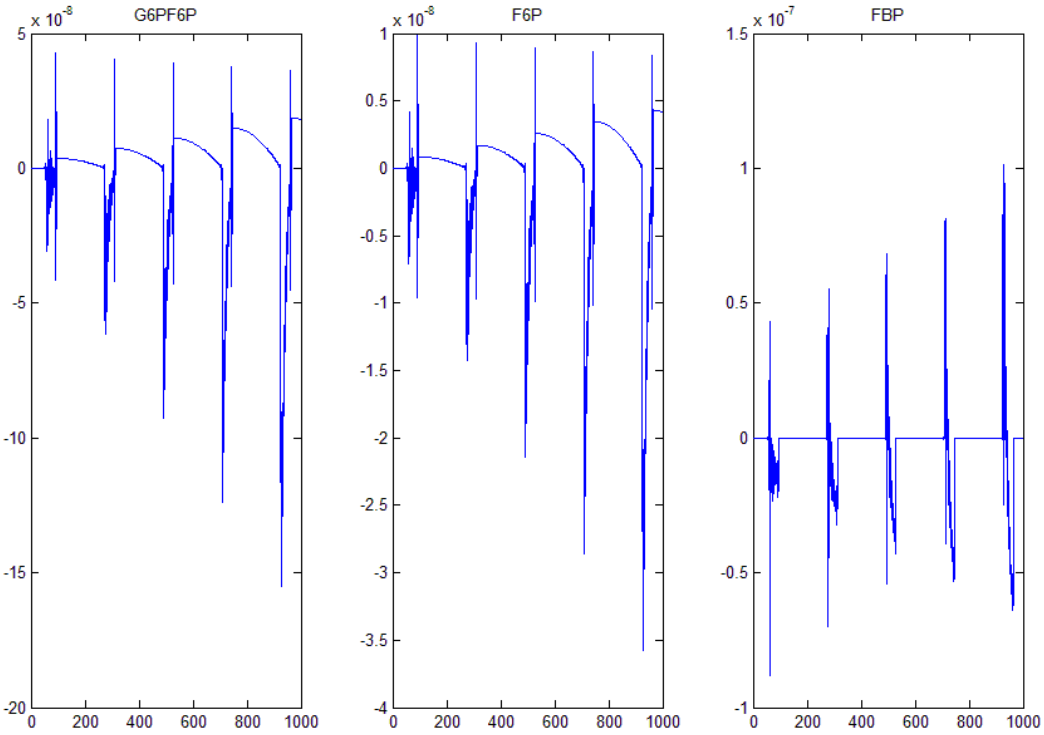
“ImpostazioniTolleranzeWestermark” è uno script creato per velocizzare le operazioni di importazione del modello in Matlab e, soprattutto, di definizione delle tolleranze dei risolutori ODE.

Nella simulazione con un gradino di glucosio in ingresso, si parte dalle concentrazioni iniziali delle specie e dal desiderato valore di glucosio fino all’inizio del gradino, momento da cui si fa ripartire il sistema fornendo come condizioni iniziali gli ultimi valori assunti dalle specie e la nuova concentrazione di glucosio:

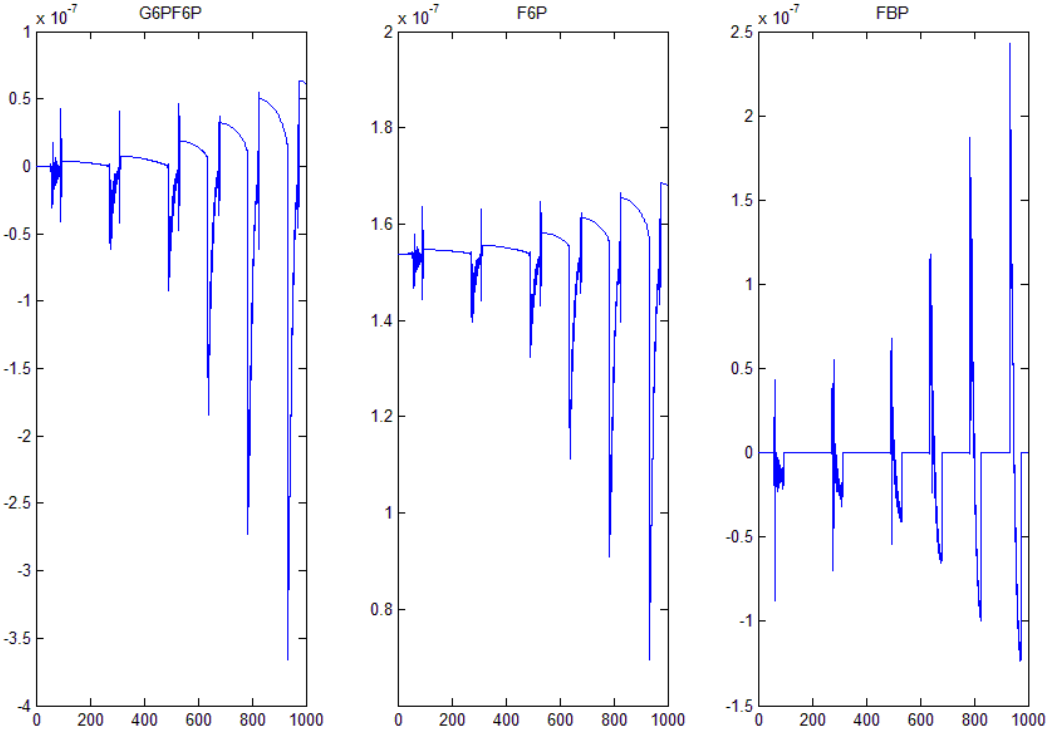
```
function [t,x] = WestermarkODEStep(t_step, GLC2)
ImpostazioneTolleranzeWestermark;
ConcIniz = Westermark2003_Pancreatic_GlycOsc_basic;
[t1,x1] = ode45(@Westermark2003_Pancreatic_GlycOsc_basic, [0, t_step], ConcIniz, options);
x1(end,1) = GLC2;
[t2,x2] = ode45(@Westermark2003_Pancreatic_GlycOsc_basic, [t_step, 1000], x1(end,:), options);
t = [t1;t2(2:end)];
x = [x1;x2(2:end,:)];
plot(t,x);
end .
```

## III.III.III. CONFRONTI

Analizzando in parallelo i risultati ottenuti dai sistemi privi di ingresso, implementati rispettivamente in Matlab e Simulink, si scorgono differenze assolute molto contenute, dell’ordine del $10^{-7}$ mmol*$l^{-1}$, a fronte di concentrazioni variabili tra 1 e 4 mmol*$l^{-1}$, a dimostrazione della similarità tra le due rappresentazioni:

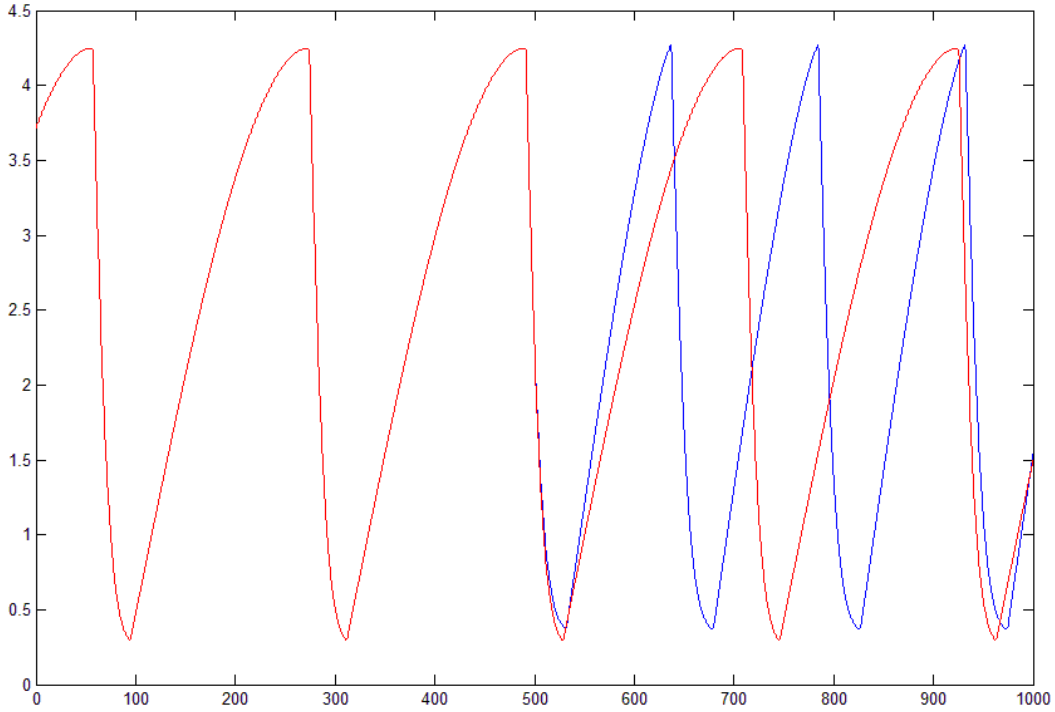



Analoghe considerazioni si possono fare relativamente ai modelli sollecitati da un andamento a gradino della concentrazione di glucosio:

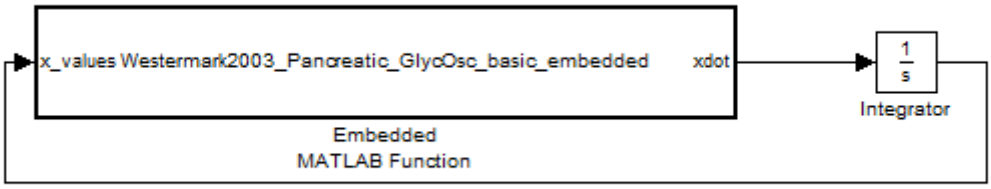



La figura seguente mostra l'andamento della concentrazione del glucosio 6-fosfato - fruttosio 6-fosfato senza e con raddoppio della concentrazione di glucosio al tempo t = 500 (in rosso la versione originale, in blu la versione con

gradino di glucosio). Questa simulazione illustra come il modello di Westermark predice l'effetto di un incremento di glucosio sulle oscillazioni metaboliche: l'innalzamento glicemico si traduce in un aumento della frequenza di oscillazione senza variazioni di ampiezza.




Infine, è stata creata una versione ottimizzata dal punto di vista prestazionale del modello Simulink della risposta libera, eliminando vari possibili fattori di ritardo dell'esecuzione:




A tal scopo, la funzione Embedded è stata rivisitata eliminando il parametro time (come già detto, non necessario) così da poter rimuovere dal modello anche il blocco Clock. È stato quindi eseguito un raffronto di prestazioni mediante il comando "tic – toc" tra il modello Simulink e il risolutore "ode45" di Matlab: il primo si è dimostrato più veloce del fratello in Matlab, probabilmente a causa della sua specifica natura di simulatore, con un tempo di esecuzione pari a 2.5 secondi contro i 3.6 della funzione della ODE suite.

# *IV. CONCLUSIONE*

In riferimento alla parte più strumentale della mia esperienza presso l'IS.I.B., in base ai risultati ottenuti si può affermare come il metodo di misura della glicemia basato sull'impedenza possa rappresentare una valida alternativa (a domicilio o in laboratorio) all'attuale procedura basata sulla reazione della glucosio – ossidasi, una volta eliminato il confondente sodio grazie alle resine a scambio ionico.
Riguardo gli studi modellistici, il percorso seguito ha avuto come inizio lo studio dei costrutti essenziali di Matlab e Simulink necessari per la simulazione, e si è concluso con il loro utilizzo in un contesto pratico e di interesse fisiologico, con le simulazioni di un modello delle oscillazioni glicolitiche derivato da una libreria standardizzata basata sul linguaggio SBML.
Uno sviluppo interessante è stato l'utilizzo del simulatore Simulink con i modelli SBML, che è stato possibile con limitate modifiche delle equazioni differenziali del modello e usando un costrutto valido per tutti i modelli. Ciò ha anche permesso di verificare le migliori prestazioni di Simulink rispetto ai risolutori delle ODE suite.

# *V. BIBLIOGRAFIA*

1. Cobelli C, Carson E: *Introduction to Modeling in Physiology and Medicine.* Academic Press, 2008
2. http://webmet.pd.cnr.it/expfit/
3. Shampine LF, Gladwell I, Thompson S: *Solving ODEs with MATLAB.* Cambridge University Press, 2003
4. http://sbml.org/Basic_Introduction_to_SBML
5. http://www.mathworks.com/products/simbiology/
6. "Westermark2003_Pancreatic_GlycOsc_basic", BIOMD0000000225 – http://www.ebi.ac.uk/biomodels-main/BIOMD0000000225
7. Westermark PO, Lansner A: A model of phosphofructokinase and glycolytic oscillations in the pancreatic beta-cell. *Biophys J.* 2003 Jul;85(1):126-39