A comparison of stochastic and deterministic model of insulin secretion from islets of Langerhans

A thesis submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

Thesis Supervisor: Dr. Pranay Goel

by
Ankit Dwivedi
April, 2012

Indian Institute of Science Education and Research Pune
Sai Trinity Building, Pashan, Pune India 411021

Certificate

This is to certify that this thesis entitled "A comparison of stochastic and deterministic model of insulin secretion from islets of Langerhans" submitted towards the partial fulfillment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research Pune, represents the work carried out by Ankit Dwivedi at Indian Institute of Science Education and Research Pune, during the academic year 2011-2012 under the supervision of Dr. Pranay Goel.

Ankit Dwivedi

Thesis committee:
Dr. Pranay Goel
Dr. Chetan Gadgil

Professor A. Raghuram
Coordinator of Mathematics

Acknowledgments

This project would not have been possible without the support and guidance of several people.

First, I would sincerely like to express my gratefulness to Dr. Pranay Goel for introducing me to the interesting field of Mathematical Biology. He has encouraged and guided me in every aspect of the project and thesis work. Being my guide and mentor for the last two years he has patiently taught me many techniques and aspects of mathematical modelling and different ways to think about and tackle a problem. Also, without his encouragement and guidance I would not have been able to participate in poster presentations at various conferences on Mathematical Biology.

Next, I would like to thank Dr. Chetan Gadgil for guiding me through many aspects of the project. His guidance and valuable suggestions has helped me a lot in learning and understanding various aspects of stochastic processes. I am also grateful to him for being a member of my thesis advisory committee.

I would sincerely like to thank Dr. Arthur Sherman for sharing relevant information about the model with us, which helped us in understanding the model in a better way and producing the deterministic solution.

I am also very thankful to my batch mate Mr. Shadab Alam for helping me through my programming work in MATLAB, and Ms. Ankita Jha a 4th year student for helping me through and making me understand the biological portion of the project.

I am also thankful to Dr. Rama Mishra, who has been my faculty advisor and guided me for the last 5 years. In the last, I would like to thank my family members and friends for showing belief in me and supporting me in every way.

- Ankit Dwivedi

Abstract

A comparison of stochastic and deterministic model of insulin secretion from islets of Langerhans

by Ankit Dwivedi

The main aim of the project is to present a stochastic version of the model of Insulin secretion in islets of Langerhans in pancreatic β-cells by Pederson et al.(2009) and account for integral copy numbers of the granules instead of concentrations. The reactions involved in the system corresponding to the granule pools are modelled as a set of coupled ordinary differential equations. We have implemented a hybrid Gillespie stochastic simulation algorithm to produce a stochastic version of this model.

In the beginning we implemented the usual Gillespie SSA in order to carry out the stochastic simulations and got discrepancies in comparison to the deterministic solution. As the model of Insulin granule pools contains time-dependent rates we later used a hybrid Gillespie SSA to include time-dependent propensities. The difference in the usual and the hybrid Gillespie algorithm is the step to calculate time of occurrence of the next reaction. Then using the hybrid Gillespie SSA, the average pool sizes were calculated and were compared to the deterministic solution which showed discrepancy in some pools.

To check the working and correctness of the algorithm, the algorithm was implemented on related examples and different cases. Euler's method was used to solve the differential equations involved. For small pool sizes for the IRP chain of the model the deterministic solution were also verified against the solutions using the Master equation. As the discrepancies were more significant in the IRP chain of the model as compared to other pools, cases with different $f_I(C_{md})$ functions , number of runs and different Euler time step were tested on the IRP chain.

We show the analytical solution for the open and closed systems. Also, we show the mean and variance over stochastic runs for fast and slow depolarisation protocols described by Pederson et al.(2009) matching up with the deterministic solution for the complete model. The calcium compartment functions used are close fits of the Arthur Sherman's description of the calcium compartment equations. For all the pools, stabilized variance is plotted against mean and deterministic solution choosing random and discrete initial conditions for each run.

Contents

Chapter 1

Introduction

The main reference for the project work was the paper by Morton Gram Pederson and Arthur Sherman "Newcomer insulin secretory granules as a highly calcium-sensitive pool" [1]. They present a model of insulin secretory cells in islets of Langerhans which include insulin granule pools. These pools correspond to the different states of the granules moving from production in the Golgi body towards the membrane and the L-type calcium channels.

The process of insulin secretion takes place as the plasma glucose concentration is increased which causes the ADP/ATP ratio to increase. Due to the increase in ATP/ADP the ATP-gated potassium channel closes causing depolarisation of the cell membrane as the potassium ions (positive charge) accumulate inside the cell [21]. This in turn opens the voltage gated calcium channels causing calcium influx. This increase in the calcium concentration leads to release of insulin containing granules from the cell. [17]

The process of secretion of insulin occurs in 2 phases. Experimentally, 2 different mechanisms are suggested for the 2 phases of insulin secretion. One which shows that the first secretion is large with the already docked granules contributing and in the second phase secretion is flat and rises with a very slow rate with the newcomer granules contributing [21], [13]. Another research shows that two different releasing pools with different calcium sensitivities exist [23], [19]. Pederson et al. propose that including a pool with a high calcium sensitivity away from the L-type calcium channels leads to the newcomer granules participating in the second phase of insulin secretion [20], [22].

The insulin granule pools model presented by Pederson et al. [1] is modified

from the model by Chen et al. [14]. The model includes 9 vesicle pools representing granules in different states as shown in Fig. 1.1. Inside the cell, from the reserve pool (RP) which is considered to be infinity, granules move towards the membrane. Passing through the actin network from the almost docked pool (AP) granules tether weakly to the membrane with a very high affinity for cytosolic calcium and are said to be in the highly calcium sensitive pool (HCSP). Maturing further the granules undergo docking and priming, and are said to be in the Docked pool (DP) and Primed pool (PP) respectively. Moving further the primed granules move close to the L-type calcium channel and are in the Immediately releasable pool (IRP). From the IRP the granules fuse with low affinity for microdomain calcium. FHP and FIP are the fusion pools of HCSP and IRP respectively. After fusing, the fusion pore expands and the granules are in the releasing pools. RHP and RIP are the releasing pools of HCSP and IRP respectively. The docked pool, primed pool and the immediately releasable pool are identified as readily releasable pool (RRP). [1]

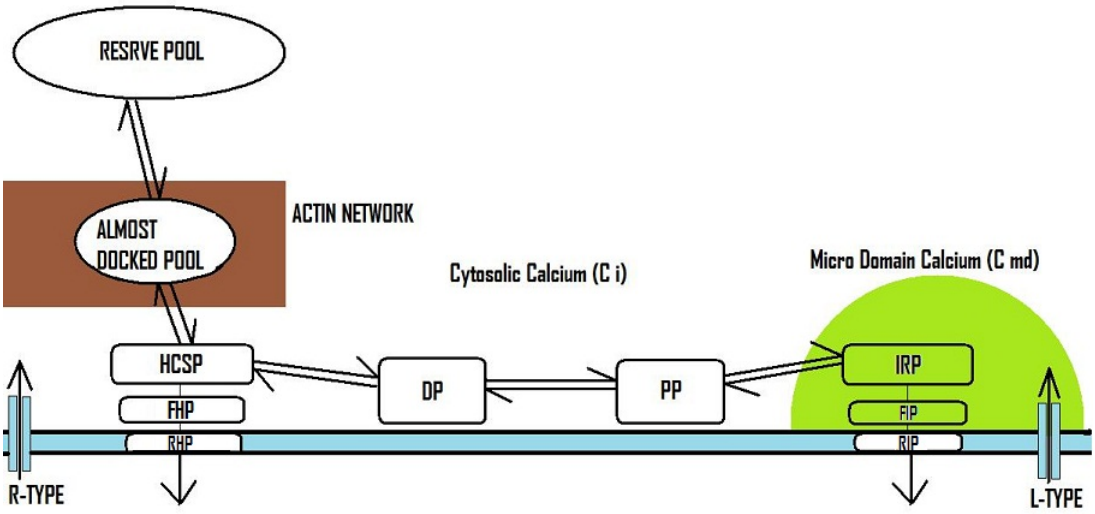


Figure 1.1: Schematic overview of the insulin granule pools model by Pederson et al. Granules from RP goes to HCSP through AP. Maturing further they are in DP and then PP. The primed granules move close to the L-type calcium channels and are said to be in IRP. FHP and FIP are the fusion pools of HCSP and IRP respectively. RHP and RIP are the releasing pools of HCSP and IRP respectively.

The microdomain calcium compartment receive calcium ions from the L-type Voltage gated calcium channels and cytosolic calcium compartment receives calcium ions from other types of calcium channels. There is also a diffusion process between the

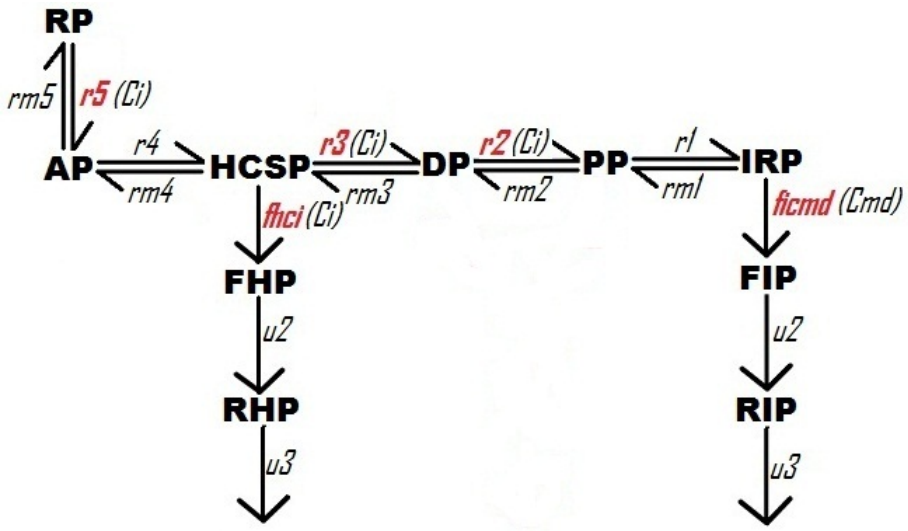


Figure 1.2: Reaction overview of the Arthur Sherman's description of the insulin ganule pools model. The rates in red color are the function of calcium and hence time. r_5, r_3, r_2 and $f_h(C_i)$ are the function of Cytosolic calcium C_i and the $f_I(C_{md})$ is the function of micro domain calcium C_{md}.

microdomain and cytosolic calcium compartments. The calcium equations defined by Pederson et al. are adapted from the sample experiments carried out by Yang and Gillis [19], [23], in which the membrane was depolarized to +20 mv 3 times for 10 ms followed by the photo elevation of C_i to 1.8 μM to release the HCSP. Parameters were changed and chosen to get pools sizes defined by Rorsman et al. [1], [15]. The oscillations approximated were square pulses of membrane depolarisation from -70 mv to + 20 mv. Fast protocol refers to the period of oscillation as 1 minute and the slow protocol refers to the period of oscillations as 6 minutes. For the equations of the calcium compartments see Appendix: Supporting information. Also, the concentration of microdomain calcium and cytosolic calcium for the fast and slow protocol are plotted versus time and shown in Section. 5.2.

We follow Arthur Sherman's description of the insulin granule pools model in which the combined fusion pools and releasing pools of DP and PP are also described. This chain is not considered in the simulations as it contribute very less to the total secretion and capacitance. The reaction overview of the model is shown in Fig. 1.2. For the equations involved, initial conditions and the involved parameters see Appendix : Supporting information.

For all the pools and the calcium compartments in the model, the rate of change of

the concentration are described by first-order ordinary differential equations (ODE's). The model of insulin secretion in islets of Langerhans is deterministic and the concentrations are continuous as the rate law is described by the ordinary differential equations. The species with large number of particles or concentration can be considered to be continuous variables. But, the particles or the granules (in our case) involved can not be fraction. These are discrete quantities and for the small pool sizes these can be stochastic processes described by probability functions. The stochastic model represents a biological system more accurately when the system can have only discrete values. [9]

To produce the stochastic solution of the existing deterministic solution we used a hybrid Gillespie stochastic simulation algorithm which is modification of the classical Gillespie SSA to include time-dependent propensities (Described in Chapter 2). The usual Gillespie algorithm is highly used to stochastically simulate chemical and biochemical models accurately. Basically, in this SSA we choose two random numbers from the uniform distribution, one for calculating the time of occurrence of the next reaction and other for selecting the reaction taking place at that time. The pseudo code for the usual and hybrid Gillespie SSA are explained in Chapter 2. [2]

We also verified the working of the hybrid Gillespie SSA and correctness of the code on different examples and with different cases of membrane depolarisation. We also verify the deterministic solution with the Master equation. Later in chapter 4 we compute analytical solution for the mean and variance in open and closed system. The mean and variance are shown matching up with the deterministic solution for both the protocols (fast and slow) and integral copy numbers for the granules can be predicted.

Chapter 2

Algorithm

For developing a stochastic version of the model of insulin secretion from the islets of Langerhans [1], we have used a modified Gillespie stochastic simulation algorithm to include the time-dependent propensities as the model contains time-dependent rates. This algorithm simulates the time evolution of the coupled equations stochasticaly which is in a way solving the Master equation [3]. In this chapter we describe and explain the working of the usual and hybrid Gillespie SSA. The difference between usual and hybrid Gillespie SSA is the step to calculate the time of occurrence of the next reaction (t_{next}) as t_{next} depends on the total propensity.

2.1 Gillespie algorithm

Gillespie stochastic simulation algorithm simulates chemical and biochemical reactions and considers the time evolution as a random process governed by the Master equation [2]. Unlike the in the deterministic solution, inherent fluctuations and relations are accounted using this SSA. Mathematically, it is very similar to kinetic Monte Carlo method. In Gillespie SSA, each reaction is defined by a probability $a_i\,dt$. Then, we choose 2 random numbers from the uniform distribution for calculating the time of occurence of the next reaction i.e t_{next} and choosing the reaction occuring at t_{next}.

2.1.1 Pseudo code for Gillespie SSA

The pseudo code for the Gillespie SSA for $i = 1, 2, ...M$ reactions is as follows [2]:

- Calculate the propensity a_i for each reaction, which is the product of two parts:

(the reaction rate k_i for the reaction i)$\times$(the number of particles of the reactant)

- Let $a0$ be sum of all the $a_i's$,

$$a0 = \sum a_i$$

- To find the time after t at which the next reaction will take place (let it be denoted with "τ"), draw a random number u_1 from a uniform distribution function and calculate the t_{next} as shown in Eq. 2.7.

- Now choose at random the reaction occurring at time $t + \tau$ by getting another random number u_2 from a uniform distribution. If the number falls between 0 and $a_1/a0$, first reaction is chosen ; if the number is between $a_1/a0$ and $(a_1 + a_2)/a0$, second reaction is chosen and so on. The numbers of molecules involved will change due to occurrence of the chosen reaction at time $t + \tau$.

- Therefore the number of particles of the reactant involved in the choosen reaction will change. Update the values of the a_i with the new distributions of the particles at time $t + \tau$.

- The process can be reiterated for as long as one wants to evolve the system.

2.1.2 Expression for t_{next}

The reactants involved in the model are exponentially distributed, so the random numbers have to be choosen from the exponential distribution instead of uniform distribution. We use Inverse Transform method for choosing continuous random numbers.

$$\text{Uniform} \longrightarrow \text{Exponential}$$

For transforming the uniform distribution to some other distribution the process is as follows,
Let X be a random variable with F(x) as cumulative distribution function (CDF) where $x \in \mathbb{R}$
Then CDF in terms of probability density functions is defined as

$$CDF(x) = \int_{-\infty}^{x} f(t)dt \tag{2.1}$$

Also,

$$F(x) = P(X \leq x) \tag{2.2}$$

Since the CDF of the probability function is non-decreasing, a quantile function for $y \in [0, 1]$ may be defined as

$$F^{-1}(y) = inf x : F(x) \geq y \tag{2.3}$$

Now, if U is uniformly distributed in (0,1), then

$$X = F^{-1}(U) \tag{2.4}$$

can be shown to have a CDF = F(x)

Proof:

$$\begin{aligned} P(X \leq x) &= P(F^{-1}(U) \leq x) \\ &= P(U \leq F(x)) \\ &= F(x) \end{aligned}$$

So, Given a density function convert it to $CDF = F(x)$.

Next, Set $F(x) = U$ solving x in terms of U.

Now,

For Exponential Distribution, density will be $\lambda e^{-\lambda x}$, where λ is the total propensity of the system.

Integrating the density function, we get

$$\begin{aligned} \int f(x)dx &= \int_0^x \lambda e^{-\lambda x} dx \\ &= \lambda \int_0^x e^{-\lambda x} dx \\ &= e^{-\lambda x}|_0^x \\ &= 1 - e^{-\lambda x} \end{aligned} \tag{2.5}$$

Equationg the CDF of the exponential distribution to uniform distribution implies

$$\begin{aligned} 1 - e^{-\lambda x} &= U \\ e^{-\lambda x} &= 1 - U \\ -\lambda x &= ln(1 - U) \\ x &= -1/\lambda ln(1 - U) \end{aligned} \tag{2.6}$$

Therefore, the expression for t_{next} (x) that we get in terms of U is

$$x = 1/\lambda ln(1/U) \tag{2.7}$$

where, λ is the total propensity of the system and U is the uniform random number.

2.2 Gillespie algorithm with time-dependent rates

As mentioned earlier that the Insulin granule pools model include time dependent rates, a modified Gillespie SSA is used to include the time-dependent propensities. Now, for the time dependent rates λ the total propensity will be a function of time t,

$$P(k) = 1 - e^{-\int_t^{t+\tau} \lambda(t)dt} \tag{2.8}$$

Equating the CDF to the uniform distribution implies

$$1 - U \sim U = e^{-\int_t^{t+\tau} \lambda(t)dt} \tag{2.9}$$

Now, equating a dummy variable to the total propensity, we get

$$\dot{X}_{dummy} = \lambda(t) \tag{2.10}$$

$$\int_t^{t+\tau} X = -lnU \tag{2.11}$$

The value of t we get after checking this event is the time of occurence of the next reaction (t_{next}).

Therefore, the pseudo code for the hybrid Gillespie algorithm with time-dependent propensity $\lambda(t)$ is as follows:

- Calculate $\lambda(t) = a0(t) = \sum a_i(t)$ *(Number of molecules involved)*
- $\dot{X}_{dummy} = a0(t)$; $X(0) = 0$
- Generate $Y \in U[0, 1]$
- Integrate X from t to $t + \tau$ and stop when $\int_t^{t+\tau} X = -lnY$

The value of t we get is the time at which the next reaction is occurring i.e t_{next}. After getting the value of t_{next}, follow the usual Gillespie algorithm.

In further chapters we show some cases to verify the working and correctness of the algorithm. Also, stochastic simulations for the complete model using the hybrid Gillespie SSA are shown.

Chapter 3

Verification of hybrid Gillespie SSA

In this chapter we show different examples and cases tested to verify the working and correctness of the hybrid Gillespie algorithm. We have tested the working of the hybrid Gillespie algorithm on a simple model, where A degrades to, and another where A acts a catalyst in the production of B (Section. 3.1) showing that the results using usual Gillespie algorithm and results using hybrid Gillespie algorithm agree well. For some pools with small number of particles the deterministic solution is compared against the Master equation. Later in this chapter we simulate different cases of the IRP chain of the Insulin granule compartment model, as when the differential equations were solved using a ODE solver *ode*23, the IRP chain of the model showed discrepancy compared to the rest of the model. To test this we have used Euler's method to solve the differential equations. IRP chain of the model is simulated using different $f_I(C_{md})$ functions as it is the time-dependent rate and also with different number of runs and different Euler time steps.

3.1 Catalysis model

In the example, where A degrades with a rate k_1 and catalyses the formation of B, i.e, B is produced with the rate $k_2\,[A]$ (time-dependent rate), it has been shown that the results using usual Gillespie SSA and hybrid Gillespie SSA agree well.

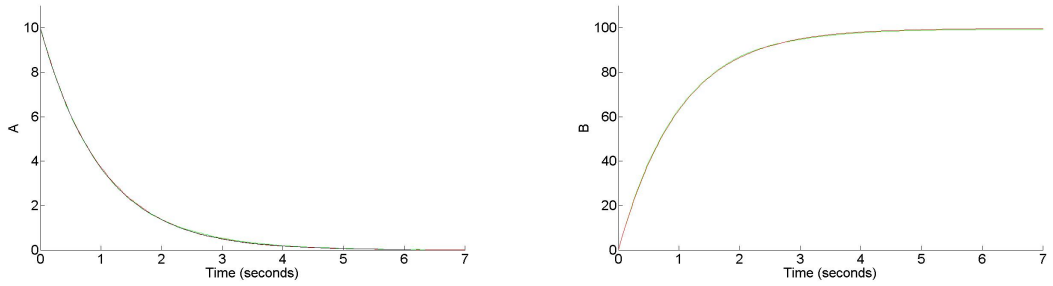
$$A \xrightarrow{k_1} \phi$$

$$\phi \xrightarrow{k_2[A]} B$$

where, the initial concentrations and constant rates are
A(0) = 10 , B(0) = 0, $k_1 = 1\ s^{-1}$, $k_2 = 10\ s^{-1}$
Average over 1000 simulations for 7 seconds using usual Gillespie algorithm (green) and hybrid Gillespie algorithm (red) is calculated.
Euler method is used for solving the differential equations involved with a Euler time step = $1e^{-4}$.
The analytical solution for A implies

$$A(t) = A(0)e^{-|k_1\, t|} \tag{3.1}$$

For both A and B, average over 1000 runs using usual Gillespie algorithm (green) matches closely with the average over 1000 runs using hybrid Gillespie algorithm (red). For A both the averages match with the deterministic solution. Deterministic solution for B is not calculated. (Fig. 3.1)



(a) A (b) B

Figure 3.1: Comparison of the average solution using usual Gillespie algorithm (green) and hybrid Gillespie algorithm (red). (a) A, both the solutions agree very well with the deterministic solution deterministic solution, (b) B, both solutions overlap.

3.2 Verification with the Master Equation

In this section we have compared the deterministic solution to the Master equation of the IRP chain of the model with 3 particles with a time dependent rate ($f_I(C_{md})$). It is a closed system i.e no secretion from RIP.

$$IRP \xrightarrow{f_I(C_{md})} FIP \xrightarrow{u_2} RIP$$

where, $f_I(C_{md})$ is the function with 3 peaks at t = 0.1s, 0.2s and 0.3s for 0.01s as shown in Fig. 5.2(a) and $u_2 = 3$.
Number of possible states of the system = 10, i.e

IRP	FIP	RIP
3	0	0
0	3	0
0	0	3
2	1	0
2	0	1
0	2	1
1	2	0
1	0	2
0	1	2
1	1	1

Let, P_{300} be the probability of being in state IRP = 3 , FIP = 0 and RIP = 0. Similarly P_{030} , P_{003} , P_{210} , P_{201} , P_{021} , P_{120} , P_{102} , P_{012} and P_{111} be the probability of being in corresponding states Fig. 3.2. Then,
where,

$P_{300}(0) = 1$, $P_{210}(0) = 0$, $P_{030}(0) = 0$, $P_{201}(0) = 0$, $P_{021}(0) = 0$, $P_{120}(0) = 0$, $P_{003}(0) = 0$, $P_{102}(0) = 0$, $P_{012}(0) = 0$, $P_{111}(0) = 0$
Therefore, probability equations corresponding to each state, will be

$$
\begin{aligned}
P'_{300} &= -3\, f_I(C_{md})\, P_{200} \\
P'_{210} &= 3\, f_I(C_{md})\, P_{300} - 2\, f_I(C_{md})\, P_{210} - u2\, P_{110} \\
P'_{120} &= 2\, f_I(C_{md})\, P_{210} - f_I(C_{md})\, P_{120} - 2\, u2\, P_{120} \\
P'_{201} &= u2\, P_{210} - 2\, f_I(C_{md})\, P_{201} \\
P'_{030} &= f_I(C_{md})\, P_{120} - 3\, u_2\, P_{030} \\
P'_{111} &= 2\, u_2\, P_{120} - f_I(C_{md})\, P_{111} - u2\, P_{111} + 2\, f_I(C_{md})\, P_{201} \\
P'_{021} &= 3\, u_2\, P_{030} - 2\, u_2\, P_{021} + f_I(C_{md})\, P_{111} \\
P'_{102} &= u_2\, P_{111} - f_I(C_{md})\, P_{102} \\
P'_{012} &= 2\, u_2\, P_{021} - u_2\, P_{012} + f_I(C_{md})\, P_{102} \\
P'_{003} &= u_2\, P_{012}
\end{aligned}
$$

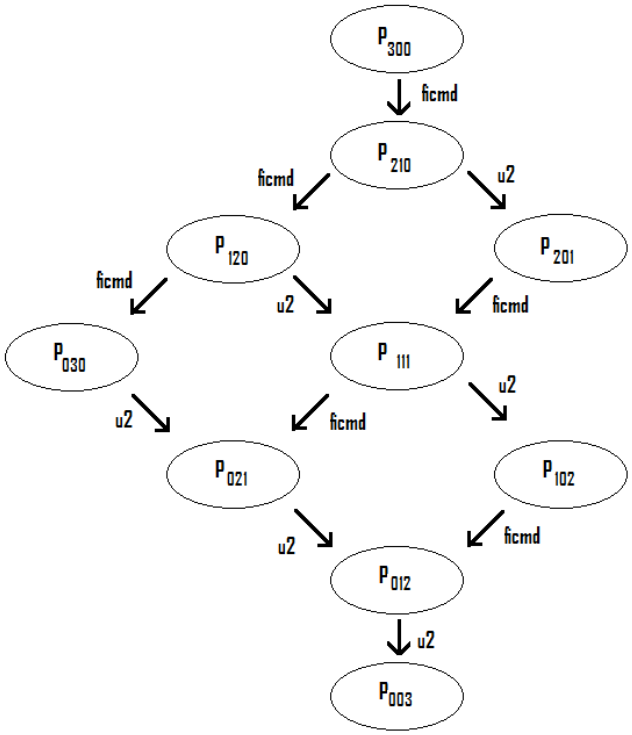


Figure 3.2: Representation of probability of being in each state.

Then the number of particles in IRP , FIP and RIP are

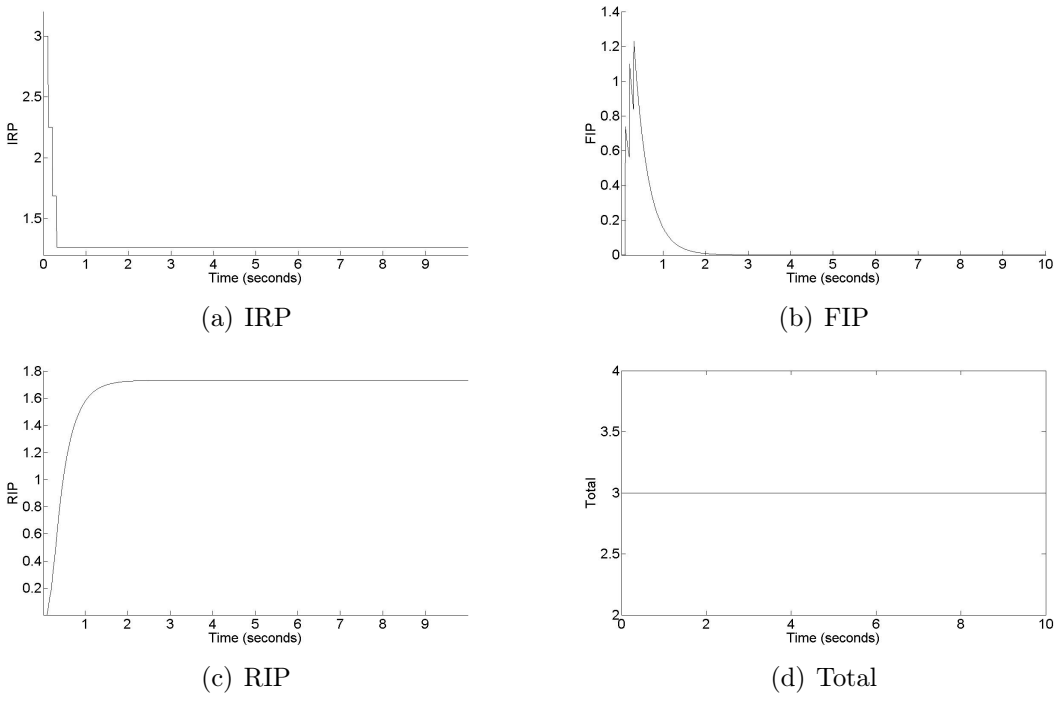
$$N_{IRP} = 3\,P_{300} + 2\,P_{210} + 2\,P_{201} + 1\,P_{120} + 1\,P_{111} + 1\,P_{102} \tag{3.2}$$
$$N_{FIP} = 3\,P_{030} + 2\,P_{120} + 2\,P_{021} + 1\,P_{210} + 1\,P_{012} + 1\,P_{111} \tag{3.3}$$
$$N_{RIP} = 3\,P_{003} + 2\,P_{012} + 2\,P_{102} + 1\,P_{021} + 1\,P_{201} + 1\,P_{111} \tag{3.4}$$

We know that,

$$[IRP]' = -f_I(C_{md})\,[IRP] \tag{3.5}$$
$$[FIP]' = f_I(C_{md})\,[IRP] - u_2\,[FIP] \tag{3.6}$$
$$[RIP]' = u_2\,[FIP] \tag{3.7}$$

Now, the solution of the Master equation can be shown equal to the analytical solution

$$
\begin{aligned}
N'_{IRP} &= -f_I(C_{md})\,(3\,P_{300} + 2\,P_{210} + 2\,P_{201} + 1\,P_{120} + 1\,P_{111} + 1\,P_{102}) \\
&= -f_I(C_{md})\,IRP \\
N'_{FIP} &= f_I(C_{md})\,(3\,P_{300} + 2\,P_{210} + 2\,P_{201} + 1\,P_{120} + 1\,P_{111} + 1\,P_{102}) \\
&\quad -u_2\,(3\,P_{030} + 2\,P_{120} + 2\,P_{021} + 1\,P_{210} + 1\,P_{012} + 1\,P_{111}) \\
&= f_I(C_{md})\,IRP - u_2\,FIP \\
N'_{RIP} &= u_2\,(3\,P_{030} + 2\,P_{120} + 2\,P_{021} + 1\,P_{210} + 1\,P_{012} + 1\,P_{111}) \\
&= u_2\,FIP
\end{aligned}
$$



(a) IRP (b) FIP

(c) RIP (d) Total

Figure 3.3: The deterministic solution (black) exactly overlaps the solution using master equation (red) for (a) IRP, (b) FIP, (c) RIP. Also, (d) Total = IRP + FIP + RIP, total number of molecules at each point of time is constant i.e 3.

Hence, the deterministic solution matches up with the solution using the Master equation.

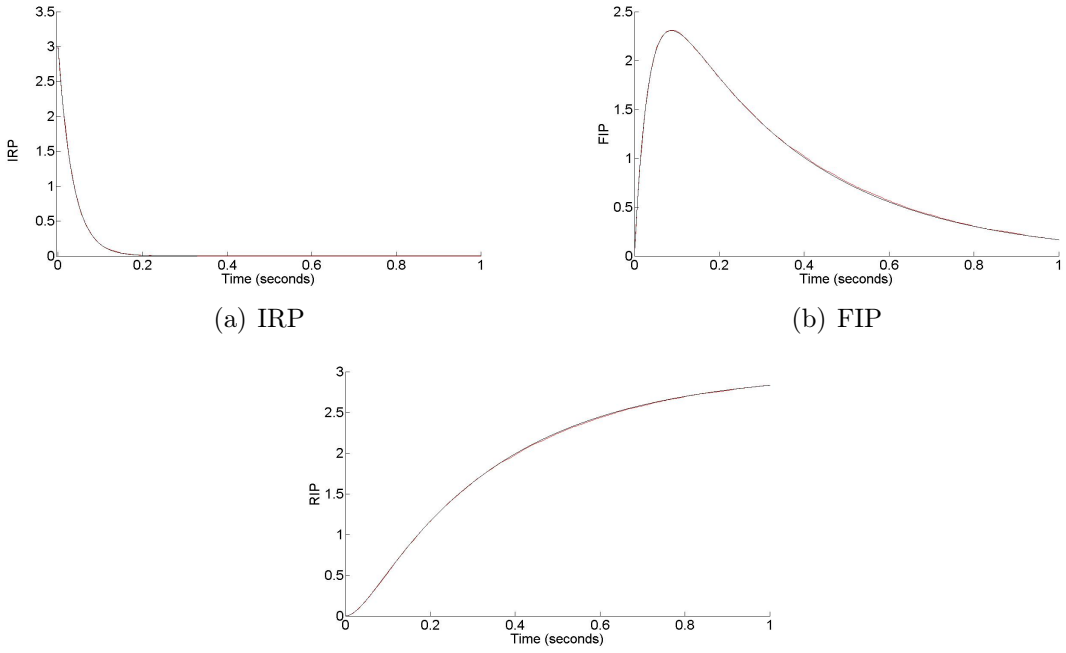
3.3 IRP chain of the Insulin granule compartments model

3.3.1 Constant $f_I(C_{md})$

In this section we have simulated the IRP chain of the vesicle compartment model considering no secretion from RIP (i.e considering it to be a close system). IRP goes to FIP with a constant rate $f_I(C_{md})$ and FIP goes to RIP with a constant rate u_2. Solution using usual Gillespie algorithm can not be calculated as the initial propensity for this chain of the model is 0 and we get $tnext = \infty$.
$f_I(C_{md})$ and u_2 are kept constant.
Average over 5000 simulations for 10 seconds using hybrid Gillespie algorithm is calculated. (red)
Euler method is used for solving the differential equations involved with a Euler time step (d) = e^{-4}.

$$IRP \xrightarrow{f_I(C_{md})} FIP \xrightarrow{u_2} RIP$$

where, the initial pool size and rates are
IRP(0) = 3 , FIP(0) = 0 , RIP(0) = 0, $u_2 = 3$, $f_I(C_{md}) = 28.72$
We see that for all the three IRP, FIP and RIP the mean over 5000 runs using hybrid Gillespie algorithm (red) and the Deterministic solution (black) match up very closely as shown in Fig. 3.4.



(a) IRP

(b) FIP

(c) RIP

Figure 3.4: Comparison of the average solution using hybrid Gillespie algorithm (red) and the deterministic solution (black). For the small pool size (a) IRP (till 1 sec), (b) FIP (till 1 sec) and (c) RIP (till 1 sec) the mean over 5000 stochastic solution agrees well with the deterministic solution.

3.3.2 $f_I(C_{md})$ as a step function

Here we have simulated the same model as above with a different $f_I(C_{md})$ function. $f_I(C_{md})$ is a step function as shown in Fig. 3.5(a). Average over 1000 simulations for 5 seconds using hybrid Gillespie algorithm is calculated.

Euler method is used for solving the differential equations involved with a Euler time step (d) $= e^{-4}$.

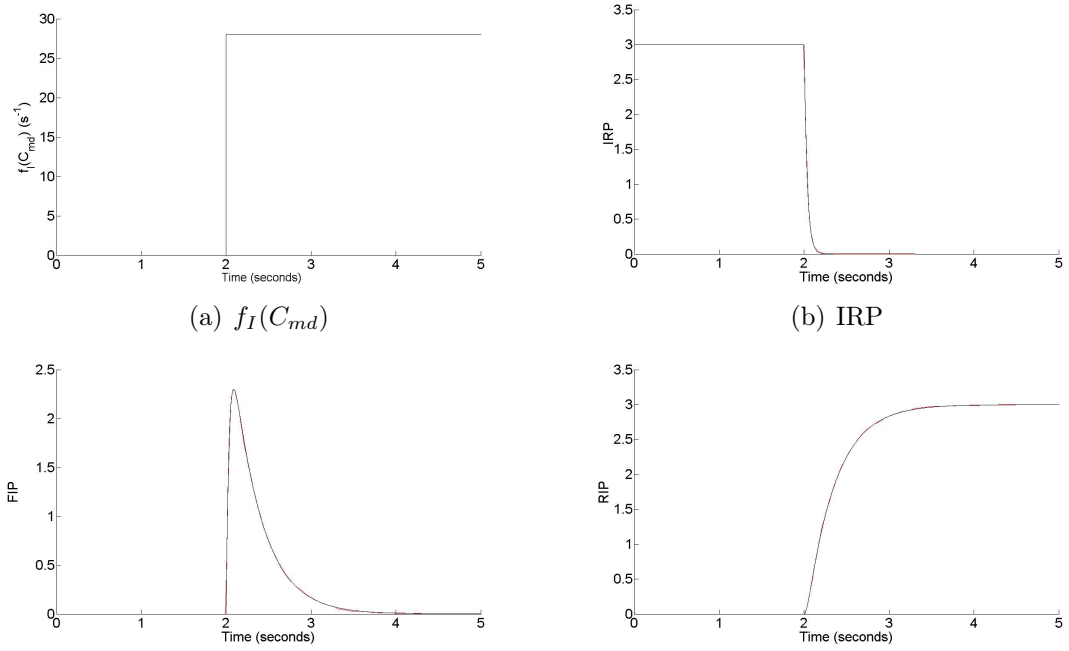
$$IRP \xrightarrow{f_I(C_{md})} FIP \xrightarrow{u_2} RIP$$

where, the initial pool size and rates are

IRP(0) = 3 , FIP(0) = 0 , RIP(0) = 0, $u_2 = 3$ and

$$f_I(C_{md}) = \begin{cases} 28.72 & \text{for } t \geq 2secs \\ 0 & \text{for } t < 2secs \end{cases}$$

We see that for all the three IRP, FIP and RIP the mean over 1000 runs using hybrid Gillespie alorithm (red) and the Deterministic solution (black) match up very closely as shown in Fig. 3.5.



(a) $f_I(C_{md})$ (b) IRP

(c) FIP (d) RIP

Figure 3.5: (a) $f_I(C_{md})$, at t = 2 value of $f_I(C_{md})$ is raised to 28.72 (step function). Comparison of the average solution using hybrid Gillespie algorithm (red) and the deterministic solution (black) for (b) IRP , (c) FIP and (d) RIP. The mean over 1000 stochastic simulations agrees well with the deterministic solution for IRP, FIP and RIP.

3.3.3 $f_I(C_{md})$ as a function of square pulses

Here we have simulated the same model as above with a different $f_I(C_{md})$ function and average is calculated over increased number of runs. $f_I(C_{md})$ is a function with 3 square pulses at 0.1, 0.2, 0.3 for 0.01 seconds as shown in Fig. 3.6(a). Average over 50,000 simulations for 10 seconds using hybrid Gillespie algorithm has been calculated. (red)

Euler method is used for solving the differential equations involved with a Euler time step (d) = e^{-4}.

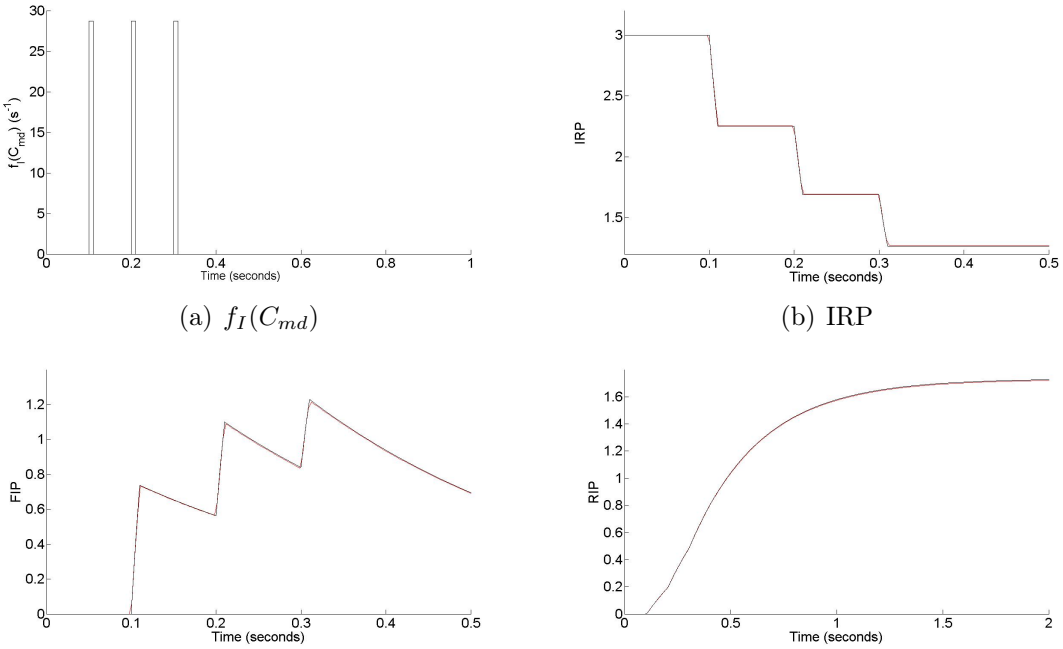
$$IRP \xrightarrow{f_I(C_{md})} FIP \xrightarrow{u_2} RIP$$

where, the initial pool size and rates are

IRP(0) = 3 , FIP(0) = 0 , RIP(0) = 0, $u_2 = 3$ and

$$f_I(C_{md}) = \begin{cases} 28.72 & \text{for } t = 0.1, 0.2, 0.3 secs\, for\, 0.01 secs \\ 0 & \text{for } elsewhere \end{cases}$$

Here also we see that for all the three pools IRP, FIP and RIP, mean over 50,000 simulations using hybrid Gillespie algorithm shows some discrepancy at the peaks in comparison to the deterministic solution as shown in Fig. 3.6. Increasing the number of runs the mean over stochastic simulation is more smoothed.



(a) $f_I(C_{md})$ (b) IRP

(c) FIP (d) RIP

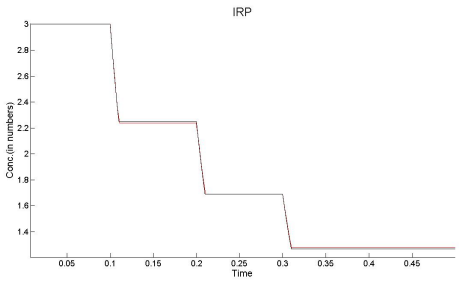
Figure 3.6: (a) $f_I(C_{md})$ (till 1 sec), at t = 0.1, 0.2, 0.3 value raised to 28.72 for 0.01 seconds. Comparison of the average solution using hybrid Gillespie algorithm (red) and the deterministic solution (black) for (b) IRP (till 0.5 secs) , (c) FIP (till 0.5 seconds) and (d) RIP (till 2 secs). The mean over 50,000 stochastic simulations agrees well with the deterministic solution with small discrepencies at the peaks for IRP and FIP.

3.3.4 Smaller Euler time step

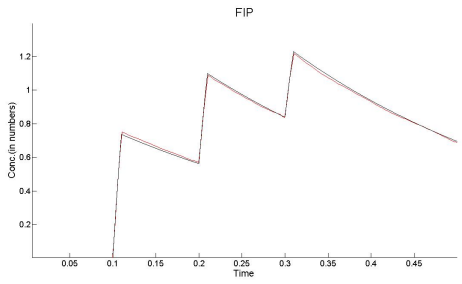
Here we have simulated the same model as above with a much smaller Euler time step. $f_I(C_{md})$ is a function with 3 square pulses at 0.1, 0.2, 0.3 for 0.01 seconds as shown in Fig. 3.6(a). Average over 50,000 simulations for 10 seconds using hybrid Gillespie algorithm has been calculated.
Euler method is used for solving the differential equations involved with a much smaller Euler time step (d) = e^{-5}.

$$IRP \xrightarrow{f_I(C_{md})} FIP \xrightarrow{u_2} RIP$$

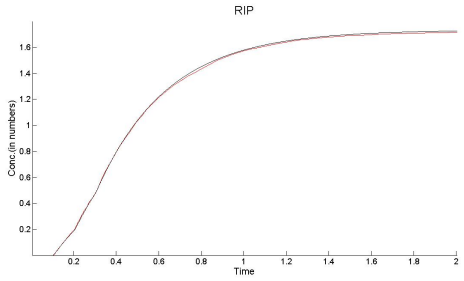
where, the initial pool size and rate u_2 and $f_I(C_{md})$ are as mentioned above in Section. 3.3.3
Here we can see that for all the three pools IRP, FIP and RIP, average over 50,000 simulations using hybrid Gillespie algorithm shows some discrepancy at the peaks in comparison to the deterministic solution as shown in Fig. 3.7.



(a) IRP



(b) FIP



(c) RIP

Figure 3.7: Comparison of the average solution using hybrid Gillespie algorithm (red) and the deterministic solution (black). (a) IRP (till 0.5 seconds) , (b) FIP (till 0.5 seconds) and (c) RIP (till 2 seconds).

Chapter 4

Mean and Variance in closed and open systems

In this chapter we show the mean and variance for a closed system i.e number of particles in our case granules are conserved, and open system i.e number of particles are not conserved. For the closed system, AP is considered to be degrading with some rate $r_1 = 1$ with a low initial particle number i.e AP(0) = 5. It has been shown in Fig. 4.1 that for a closed system, mean and variance match up and goes to 0 in the course of time. For the case of open system, AP is considered to be coupled to a infinite pool RP which increases AP with a rate $r_1 = 3$ and AP decreases to ϕ with a rate $r_2 = 1$. It has been shown in Fig. 4.2 that for open system, mean and variance settle near the value r_1/r_2 as t$\rightarrow \infty$.

4.1 Closed system

Consider the case where AP is degrading to ϕ with a constant rate r_1 i.e the system is closed

$$AP \xrightarrow{r_1} \phi$$

There can be 5 possible states for AP i.e states with 5 , 4 , 3 , 2 , 1 and 0 particles in AP. Let P_5 , P_4 , P_3 , P_2 , P_1 and P_0 be the probability of each state respectively.

Then, the differential equations corresponding to each state can be written as

$$\begin{aligned}
\frac{dP_5}{dt} &= -5\,r_1\,P_5 \\
\frac{dP_4}{dt} &= 5\,r_1\,P_5 - 4\,r_1\,P_4 \\
\frac{dP_3}{dt} &= 4\,r_1\,P_4 - 3\,r_1\,P_3 \\
\frac{dP_2}{dt} &= 3\,r_1\,P_3 - 2\,r_1\,P_2 \\
\frac{dP_1}{dt} &= 2\,r_1\,P_2 - r_1\,P_1 \\
\frac{dP_0}{dt} &= r_1\,P_1
\end{aligned}$$

Now, the expectation value of AP is

$$< n > = \sum_{n=0}^{5} n\,P_n \tag{4.1}$$

$$\frac{d < n >}{dt} = \sum_{n=0}^{5} n\,\frac{dP_n}{dt}$$

$$\frac{d < n >}{dt} = -r_1\,< n > \tag{4.2}$$

Similarly,

$$\begin{aligned}
\frac{d < n^2 >}{dt} &= \sum_{n=0}^{5} n^2\,\frac{dP_n}{dt} \\
\frac{d < n^2 >}{dt} &= -r_1\,(45\,P_5 + 28\,P_4 + 15\,P_3 + 6\,P_2 + P_1)
\end{aligned} \tag{4.3}$$

Variance can be shown to be

$$Var[\mathrm{AP}] = \mathrm{AP}(0)\;\mathrm{e}^{-\mathrm{r_1 t}}\left(1 - \mathrm{e}^{-\mathrm{r_1 t}}\right) \tag{4.4}$$

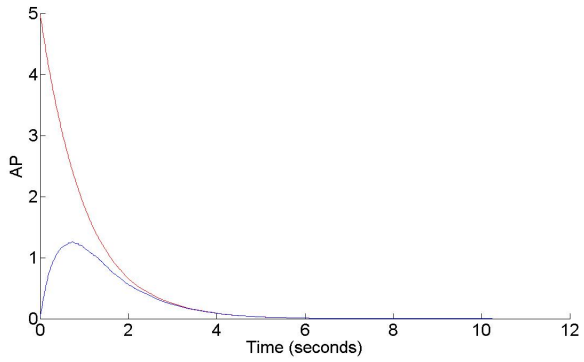


Figure 4.1: Mean (red) and Variance (blue) has been calculated over 5000 stochastic runs using Gillespie SSA and plotted against time (seconds). In the closed system if the variance is started from 0 at t = 0 it settles to 0 matching up with the mean for t$\rightarrow \infty$.

4.2 Open system

Now, for the open system, AP is considered to be coupled to a infinite size pool RP making the system open i.e granule number is not conserved.

$$RP \xrightarrow{r_1} AP \xrightarrow{r_2} \phi$$

In the open system case, there can be n possible states for AP as the granule number is also increasing. So, derivative of the n^{th} state can be written as

$$\frac{dP_n}{dt} = (n+1)\, r_2\, P_{n+1} - n\, r_2\, P_n + r_1\, P_{n-1} - r_1\, P_n \tag{4.5}$$

Therefore, the derivative of the expectation value will be

$$\begin{aligned} \frac{d < n >}{dt} &= \sum_{n=0}^{\infty} \frac{d\, n\, P_n}{dt} \\ \sum_{n=0}^{\infty} \frac{d\, n\, P_n}{dt} &= r_2[\sum_{n=0}^{\infty} n(n+1)\, P_{n+1} - \sum_{n=0}^{\infty} n^2\, P_n] \\ &\quad + r_1[\sum_{n=1}^{\infty} n\, P_{n-1} - \sum_{n=0}^{\infty} n\, P_n] \end{aligned} \tag{4.6}$$

Let the 4 terms in Eq. (4.6) be Term 1, Term 2, Term 3 and Term 4 respectively, i.e

$$\sum_{n=0}^{\infty} \frac{d\, n\, P_n}{dt} = r_2[Term1 - Term2] + r_1[Term3 - Term4] \tag{4.7}$$

Solving each term separately in Eq. (4.6) implies
Term 1 = $\sum_{n=0}^{\infty} n(n+1)\, P_{n+1}$

$$\begin{aligned} \sum_{n=0}^{\infty} n(n+1)\, P_{n+1} &= \sum_{n=0}^{\infty} (n+1)^2\, P_{n+1} - \sum_{n=0}^{\infty} (n+1)\, P_{n+1} \\ \sum_{n=0}^{\infty} n(n+1)\, P_{n+1} &= < n^2 > - < n > \end{aligned} \tag{4.8}$$

Term 2 = $\sum_{n=0}^{\infty} n^2\, P_n$

$$\sum_{n=0}^{\infty} n^2\, P_n = < n^2 > \tag{4.9}$$

Term 3 = $\sum_{n=1}^{\infty} n\, P_{n-1}$

$$\begin{aligned} \sum_{n=1}^{\infty} n\, P_{n-1} &= \sum_{n=1}^{\infty} (n-1)\, P_{n-1} + \sum_{n=1}^{\infty} P_{n-1} \\ \sum_{n=1}^{\infty} n\, P_{n-1} &= < n > +1 \end{aligned} \tag{4.10}$$

Term 4 = $\sum_{n=0}^{\infty} n\, P_n$

$$\sum_{n=0}^{\infty} n\, P_n = < n > \tag{4.11}$$

Now, putting the values of Term 1, Term 2, Term 3 and Term 4 in Eq. (4.7), we get

$$\begin{aligned} \frac{d < n >}{dt} &= r_2[< n^2 > - < n > - < n^2 >] + r_1[< n > +1 - < n >] \\ \frac{d < n >}{dt} &= r_1 - r_2 < n > \end{aligned} \tag{4.12}$$

Similarly solve for $\frac{d<n^2>}{dt}$

$$\frac{d<n^2>}{dt} = \sum_{n=0}^{\infty} n^2 \frac{dP_n}{dt} \tag{4.13}$$

$$\begin{aligned}\frac{d<n^2>}{dt} &= r_2[\sum_{n=0}^{\infty} n^2(n+1)\,P_{n+1} - \sum_{n=0}^{\infty} n^3\,P_n] \\ &\quad + r_1[\sum_{n=0}^{\infty} n^2\,P_{n-1} - \sum_{n=0}^{\infty} n^2\,P_n]\end{aligned} \tag{4.14}$$

Let the 4 terms in Eq. (4.14) be Term 1, Term 2, Term 3 and Term 4 respectively, i.e Term 1 $= \sum_{n=0}^{\infty} n^2(n+1)\,P_{n+1}$

$$\begin{aligned}\sum_{n=0}^{\infty} n^2(n+1)\,P_{n+1} &= \sum_{n=0}^{\infty}(n^2-1+1)(n+1)\,P_{n+1} \\ &= \sum_{n=0}^{\infty}\{[(n-1)(n+1)+1](n+1)\,P_{n+1}\} \\ &= \sum_{n=0}^{\infty}\{(n-1)(n+1)^2\,P_{n+1} + (n+1)\,P_{n+1}\} \\ &= \sum_{n=0}^{\infty}\{n(n+1)^2\,P_{n+1} - (n+1)^2\,P_{n+1} + (n+1)\,P_{n+1}\} \\ &= \sum_{n=0}^{\infty}\{(n+1-1)(n+1)^2\,P_{n+1} - (n+1)^2\,P_{n+1} + (n+1)\,P_{n+1}\} \\ &= \sum_{n=0}^{\infty}\{(n+1)^3\,P_{n+1} - (n+1)^2\,P_{n+1} - (n+1)^2\,P_{n+1} \\ &\quad + (n+1)\,P_{n+1}\} \\ &= \sum_{n=0}^{\infty}\{(n+1)^3\,P_{n+1} - 2\,(n+1)^2\,P_{n+1} \\ &\quad + (n+1)\,P_{n+1}\}\end{aligned}$$

$$\sum_{n=0}^{\infty} n^2(n+1)\,P_{n+1} = <n^3> - 2\,<n^2> + <n> \tag{4.15}$$

Term 2 = $\sum_{n=0}^{\infty} n^3 P_n$

$$\sum_{n=0}^{\infty} n^3 P_n = < n^3 > \tag{4.16}$$

Term 3 = $\sum_{n=0}^{\infty} n^2 P_{n-1}$

$$\begin{aligned} \sum_{n=0}^{\infty} n^2 P_{n-1} &= \sum_{n=0}^{\infty} \{[(n-1)^2 + (2n-1)] P_{n-1}\} \\ &= \sum_{n=0}^{\infty} \{(n-1)^2 P_{n-1} + 2 P_{n-1}(n-1) + P_{n-1}\} \\ \sum_{n=0}^{\infty} n^2 P_{n-1} &= < n^2 > +2 < n > +1 \end{aligned} \tag{4.17}$$

Term 4 = $\sum_{n=0}^{\infty} n^2 P_n$

$$\sum_{n=0}^{\infty} n^2 P_n = < n^2 > \tag{4.18}$$

Now, putting the values of Term 1, Term 2, Term 3 and Term 4 in Eq. (4.14), we get

$$\begin{aligned} \frac{d < n^2 >}{dt} &= r_2[< n^3 > -2 < n^2 > + < n > - < n^3 >] \\ &\quad + r_1[< n^2 > +2 < n > +1 - < n^2 >] \\ \frac{d < n^2 >}{dt} &= -r_2 (2 < n^2 > + < n >) + r_1 (2 < n > +1) \end{aligned} \tag{4.19}$$

Then, variance can be shown to satisfy

$$Var[\mathrm{AP}] = \mathrm{r}_1/\mathrm{r}_2 \tag{4.20}$$

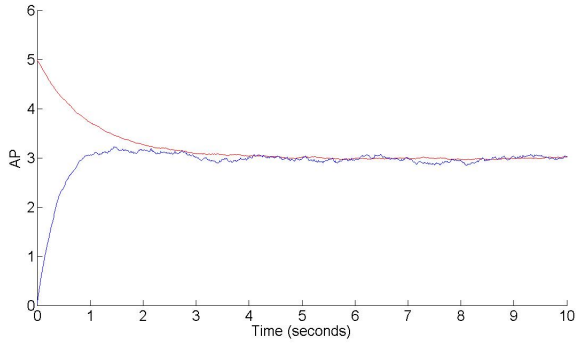


Figure 4.2: Mean (red) and Variance (blue) has been calculated over 5000 stochastic runs using Gillespie SSA and plotted against time (seconds). In the open system, mean is started from 5 and if the variance is started from 0 it settles into a non-zero equilibrium given by r_1/r_2 as the mean.

We have also calculated mean and variance making the full vesicle pool model a closed system i.e decoupling RP from AP and compared it with the open system i.e RP coupled to AP. For all the pools except AP, deterministic solution for open and close system match up exactly. For AP the deterministic solution for closed system is slightly below the deterministic solution for open system. (Fig. 4.3)

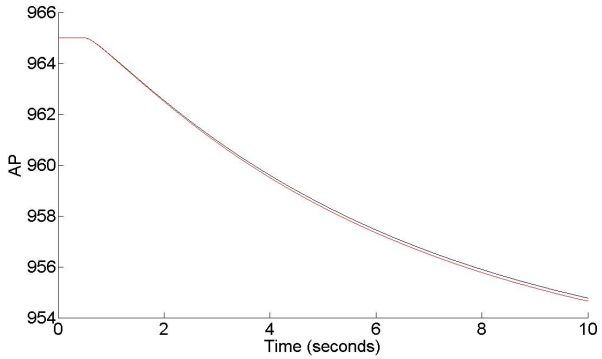


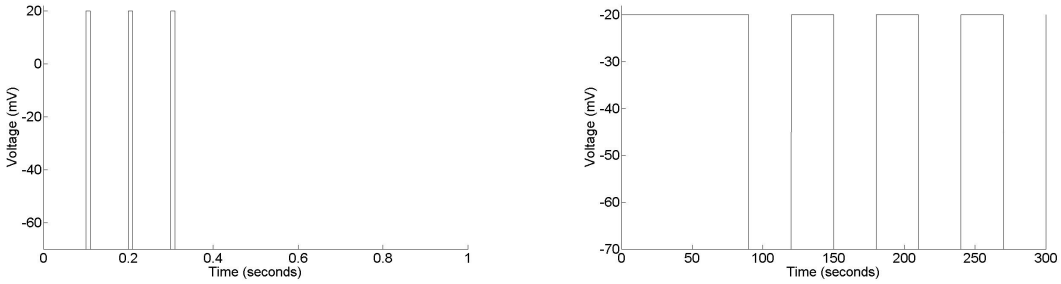
Figure 4.3: Deterministic solution when the system is open (black) and deterministic solution when the system is closed (blue).

Chapter 5

Results

5.1 Pulse protocols

Pederson et al. (2009) have described two different depolarisation patterns for the Insulin granule pool model. A fast depolarisation pattern consists of 3 peaks at very short intervals for very small depolarisation as shown in Fig. 5.1(a). A slow depolarisation pattern consist of long depolarisation for long intervals as shown in Fig. 5.1(b).



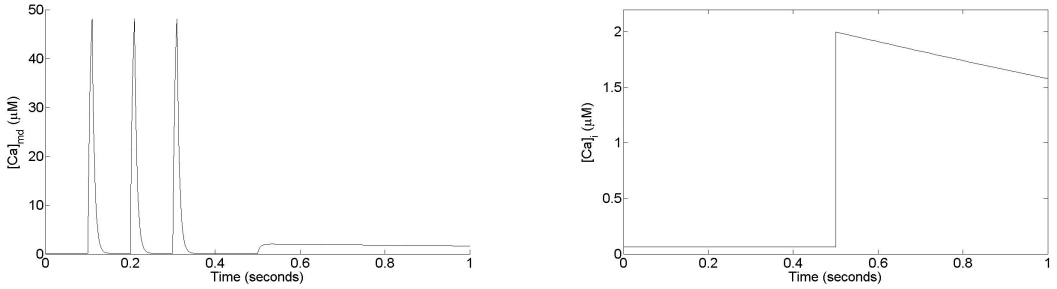
(a) Fast depolarisation

(b) Slow depolarisation

Figure 5.1: (a) Fast depolarisation pattern (shown for 1 sec), voltage goes from -70 mV to 20 mV at t = 0.1, 0.2 and 0.3 seconds for 0.01 seconds. (b) Slow depolarisation pattern (shown for 300 seconds), voltage goes from -70 mV to -20 mV at t = 0, 120, 180, 240 and so on till 3000 seconds and voltage goes from -20 mV to -70 mV at t = 90, 150, 210 and so on till 3000 seconds.

5.2 Calcium fits

In this section we show the microdomain calcium and cytosolic calcium corresponding to the fast and slow depolarisation protocols mentioned in Section 5.1. For simplicity we have used close fits of the microdomain and cytosolic calcium compartments defined by Arthur Sherman for both fast and slow protocols.



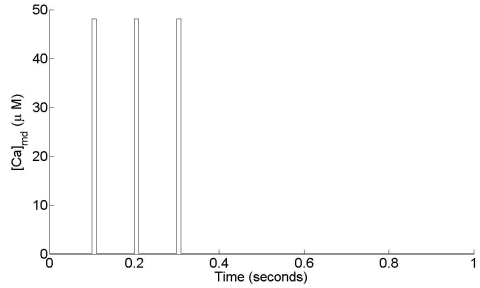
(a) Microdomain calcium (b) Cytosolic calcium

Figure 5.2: (a) C_{md} (shown for 1 sec) rises upto 48.13 at t = 0.1, 0.2 and 0.3 seconds for 0.01 seconds. (b) C_i (shown for 1 sec) is raised to $2\mu M$ at t = 0.5 seconds to depict the flash release. [1]

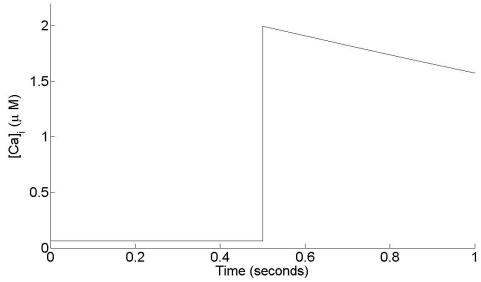
But, for simplicity we take the square pulse approximation of the calcium equations for fast protocol and it is defined as

$$\begin{aligned} C_{md} &= 0.1132 + 48.0168((heav(t-0.1))(heav(0.1+0.01-t))) \\ &\quad +(heav(t-0.2))(heav(0.2+0.01-t)) \\ &\quad +(heav(t-0.3))(heav(0.3+0.01-t)) \end{aligned} \tag{5.1}$$

$$C_i = \begin{cases} 0.06419 & \text{for } t < 0.5 \\ \Big\{1.299 * exp(-((t-(-0.6304))/1.2)^2) & \\ +(3.285e+008) * exp(-((t-(-1036))/220.8)^2) & \\ +1.375 * exp(-((t-0.4701)/2.028)^2)\Big\} & \text{for } t \geq 0.5 \end{cases} \tag{5.2}$$



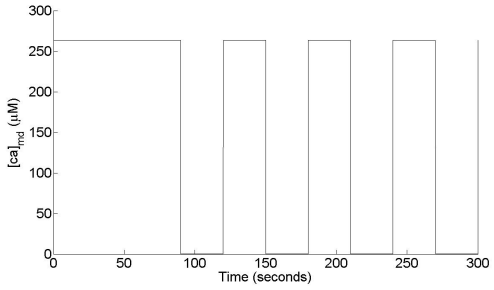
(a) Microdomain calcium



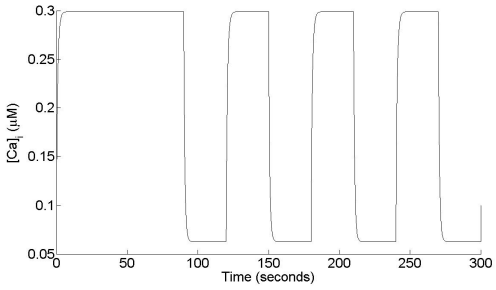
(b) Cytosolic calcium

Figure 5.3: (a) C_{md} (shown for 1 sec) is a function with 3 square pulses that rises upto 48.13 at t = 0.1, 0.2 and 0.3 seconds for 0.01 seconds. (b) C_i (shown for 1 sec) is raised to $2\mu M$ at t = 0.5 seconds.

For the slow protocol, depolarisation takes place for longer time with longer time intervals.



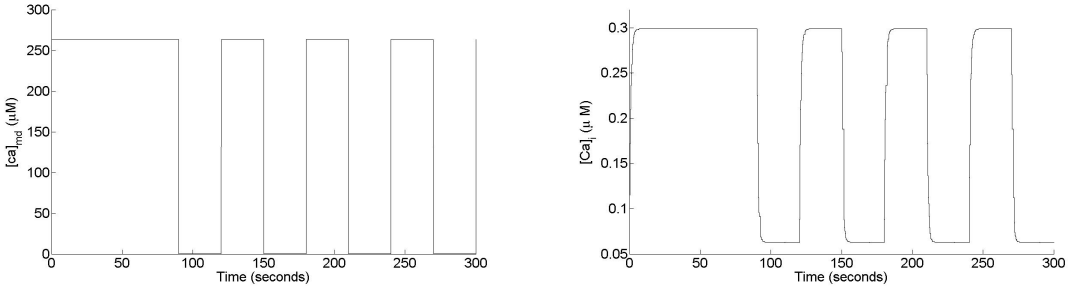
(a) Microdomain calcium



(b) Cytosolic calcium

Figure 5.4: (a) C_{md} (shown for 300 seconds), (b) C_i (shown for 300 seconds).

We have taken the fits of the calcium equations responding to slow protocol using the fitting tool 'cftool' in MATLAB. The fits for the slow protocol are shown in Fig. 5.5



(a) Microdomain calcium (b) Cytosolic calcium

Figure 5.5: (a) C_{md} (shown for 300 seconds), (b) C_i (shown for 300 seconds).

5.3 Deterministic solution

In this section we compare the deterministic solution corresponding to the model described by Pederson et al. (2009) and the deterministic solution from the close fits of the actual calcium compartment equations. It can be seen that both the deterministic solution differs as the calcium compartment equations used are different, and also the initial conditions for the pools are different. We use discrete initial conditions close to the steady state values of each pool. The comparison of both the deterministic results for each pool are shown below.

In the figures below we can see that there is a difference between both the solutions. The difference is due to the different initial conditions. For the large pools AP and DP, both the solutions follow the same trend whereas for PP the deterministic solution using the close fits is a little different from the deterministic solution by Pederson et al. For the small pools i.e IRP chain and the HCSP chain of the model, both the solutions follow the same trend with a little differnce because of the different initial conditions.

5.4 Response to fast protocol

In this section we show the variance and mean over 5000 stochastic runs compared to the deterministic solution for each pool. The microdomain calcium and cytosolic calcium functions used are the close fits of Pederson et al. (2009) model shown in Section. 5.2. We also show the stochastic mean and variance calculated with random initial conditions for each pool as to run the simulation after the variance has stabilized.

5.4.1 No variance in the initial conditions

In this section we have simulated the complete model of insulin vesicle pools with C_{md} and C_i as close fits (Fig. 5.3) of the calcium compartment equations in Arthur Sherman's representation of the Insulin granule pool model. The mean and variance over 5000 stochastic runs for 10 seconds have been calculated and compared with the deterministic solution. We have simulated the model in 2 ways, in the first method we keep the initial conditions for each pool same for each run, which shows $var(t = 0) = 0$. In the second method we choose random initial conditions from the normal distribution whose mean is size of the pool. This shows variance after it has stabilized. For all the pools mean over stochastic simulations is shown matching up with the deterministic solution. Euler's method is used to solve the differential equations involved with a Euler time step of e_{-4}. The results for the fast protocol response are shown in Fig. (5.8) and Fig. (5.9).

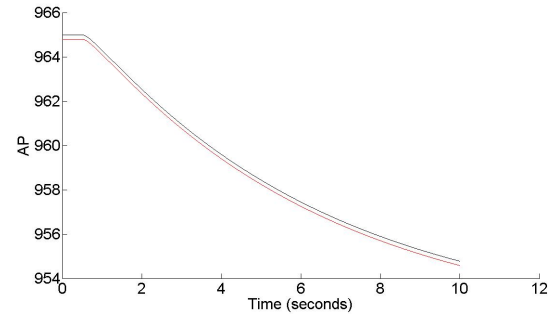
We also show the results for the case when the model is simulated using Arthur Sherman's description of C_{md} and C_i as shown in Fig. 5.2. For all the pools results are same as the results computed with close fits of the calcium equations except for the IRP chain of the model. For all the three IRP, FIP and RIP the mean over 5000 stochastic runs does not match up with the deterministic solution and the variance is also comparatively low.(shown in Fig. 5.10)

5.4.2 Initial conditions corresponding to asymptotic steady state

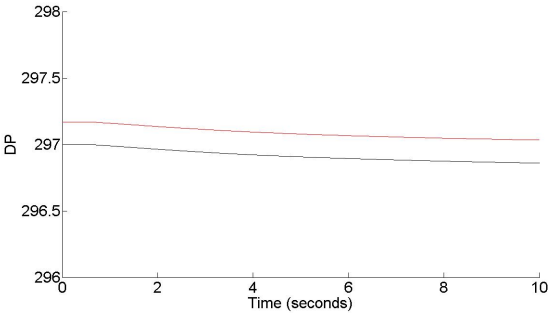
Now, when calculating the mean and variance with C_{md} and C_i as mentioned in Fig. (5.3) over 5000 stochastic runs with random initial conditions for each run, the mean, variance and the deterministic solution are shown matching up. In this case instead of variance starting from 0, it starts form the steady state value of variance. The mean is slightly below the deterministic solution as we have used the floor values of the random initial conditions from normal distribution to choose the discrete values. The results for the fast protocol response with random initial conditions for each pool for each run are shown in Fig. (5.11) and Fig. (5.12).

5.5 Response to slow protocol

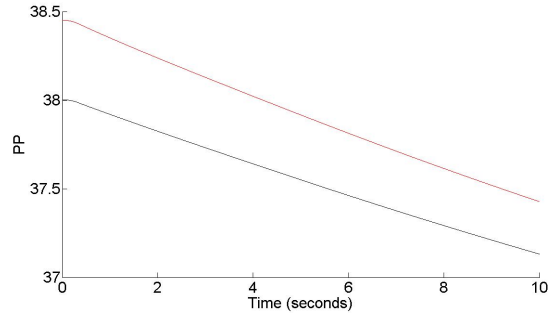
In this section we show the mean and variance over 500 stochastic runs for the slow depolarisation protocol. C_{md} and C_i are taken to be very close fits (Fig. 5.5) of the calcium compartment equations described by Pederson et al. (2009). Euler's method is used to solve the differential equations involved with a Euler time step of e^{-4}. The mean and the deterministic solution match up very closely for all the pools, but the variance is slightly away for large pools i.e AP, DP and PP. For the HCSP chain of the model the variance follows the trend as of the mean, but is not very smooth. Unlike the other pools of the model, in the IRP chain of the model the mean, variance and the deterministic solution match up closely. The results for the slow protocol response are shown in Fig. (5.13), Fig. (5.14) and Fig. (5.15).



(a) Almost Docked Pool

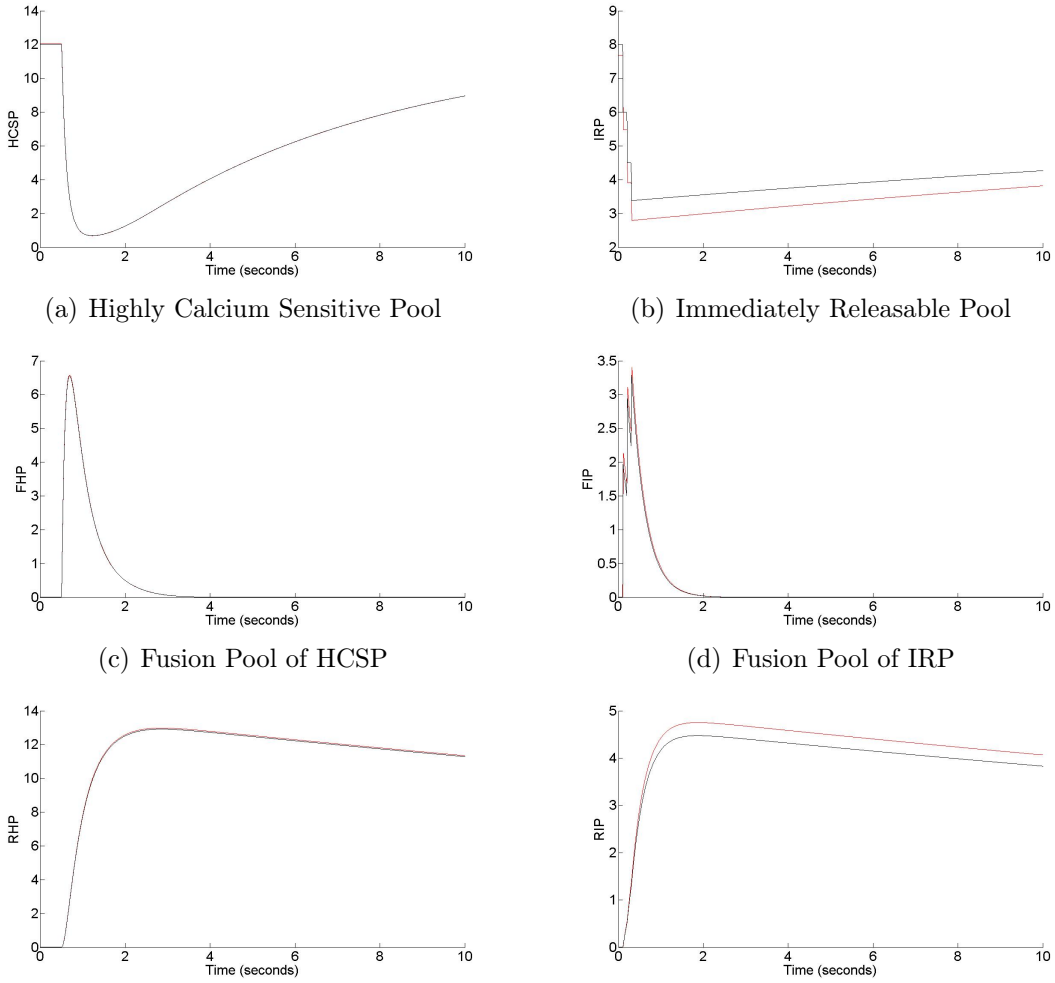


(b) Docked Pool



(c) Primed Pool

Figure 5.6: Results for the fast protocol with no variance in the initial conditions. For the large pools AP, DP and PP, the deterministic solution with the calcium equations described by Pederson et al. (red) compared with the deterministic solution with the close fits of the calcium equations (black) plotted against time (seconds).



(a) Highly Calcium Sensitive Pool

(b) Immediately Releasable Pool

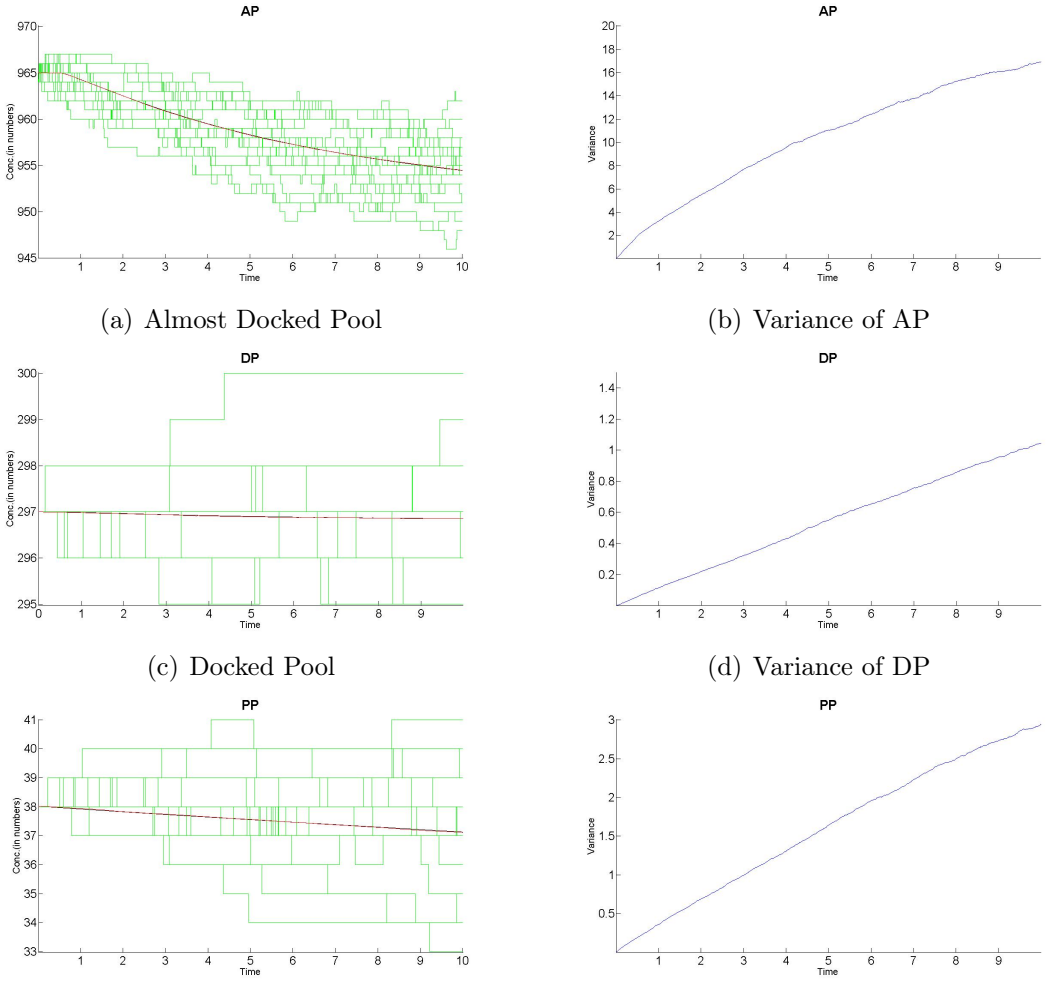
(c) Fusion Pool of HCSP

(d) Fusion Pool of IRP

(e) Releasing Pool of HCSP

(f) Releasing Pool of IRP

Figure 5.7: Results for the fast protocol with no variance in the initial conditions. For the HCSP chain and the IRP chain of the model, the deterministic solution with the calcium equations described by Pederson et al. (red) and the deterministic solution using the close fits of the calcium equations (black) plotted against time (seconds).



(a) Almost Docked Pool

(b) Variance of AP

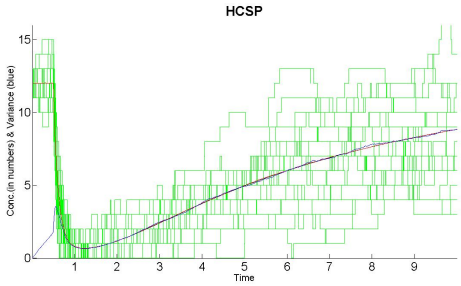
(c) Docked Pool

(d) Variance of DP

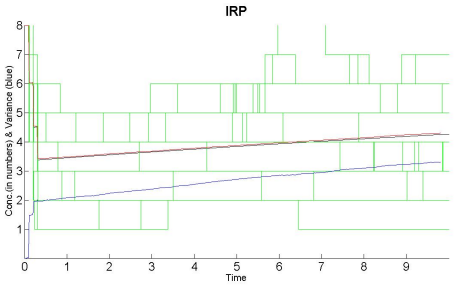
(e) Primed Pool

(f) Variance of PP

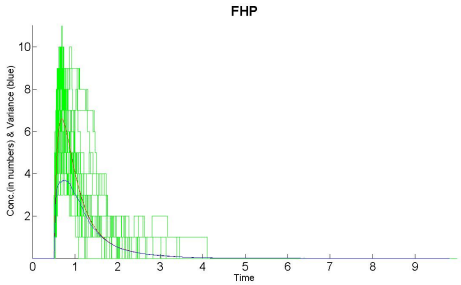
Figure 5.8: Results for the fast protocol with close fits of the calcium equations and no variance in the initial conditions. Stochastic runs (green), deterministic solution (black) and mean (red) over stochastic runs for (a) AP, (c) DP, (e) PP and (b) variance for AP, (d) variance for DP and (f) variance for PP are plotted versus time (*seconds*). For all the large pools the mean over stochastic simulations matches very closely to the deterministic solution and the variance is comparatively low.



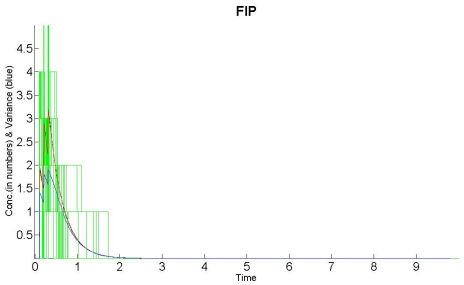
(a) Highly Calcium Sensitive Pool



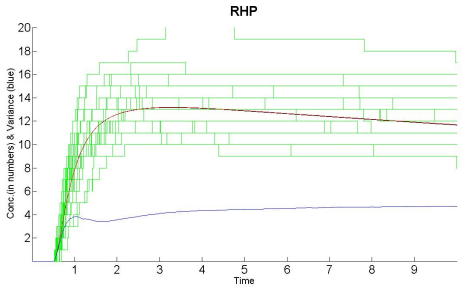
(b) Immediately Releasable Pool



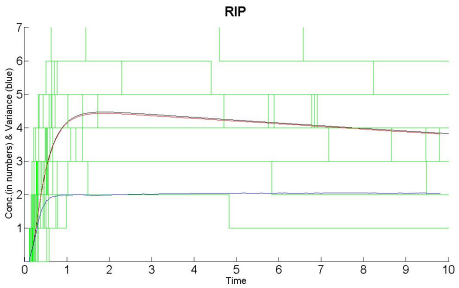
(c) Fusion Pool of HCSP



(d) Fusion Pool of IRP

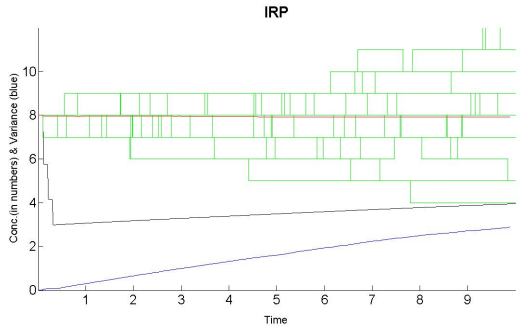


(e) Releasing Pool of HCSP

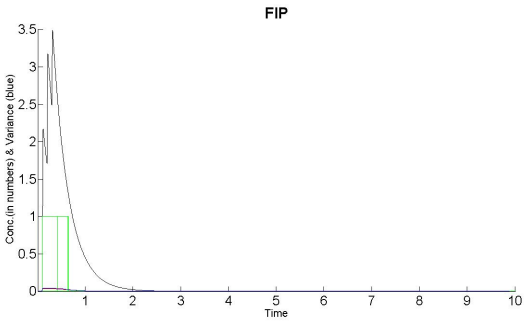


(f) Releasing Pool of IRP

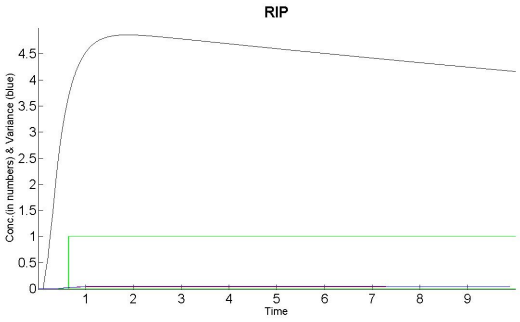
Figure 5.9: Results for the fast protocol with close fits of the calcium equations and no variance in the initial conditions. Stochastic runs (green), deterministic solution (black), mean (red) and variance (blue) over stochastic runs are plotted versus time (*seconds*) for HCSP and IRP chain of the model. For small pools the mean over stochastic solution is very close to the deterministic solution and the variance is comparatively large.



(a) Immediately Releasable Pool

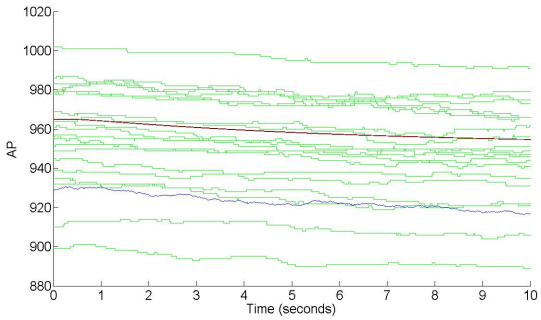


(b) Fusion Pool of IRP

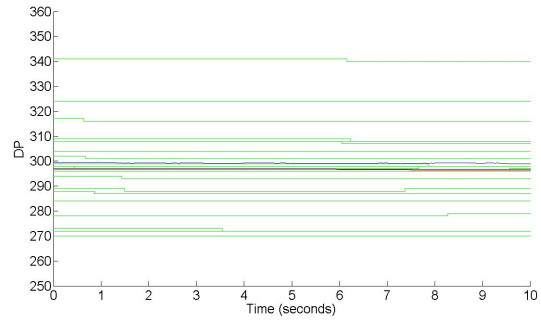


(c) Releasing Pool of IRP

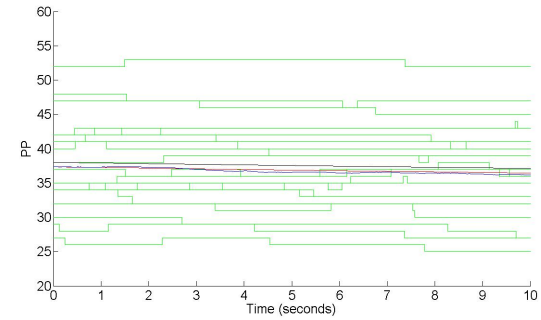
Figure 5.10: Results for the fast protocol where simulations are carried out with the Arthur Sherman's description of the calcium compartment equations. The stochastic solution (green) does not match up with the deterministic solution (black) and also the variance (blue) is very low for (a) IRP, (b) FIP and (c) RIP. The mean and variance over 5000 stochastic simulations don't agree with the deterministic solution.



(a) Almost Docked Pool

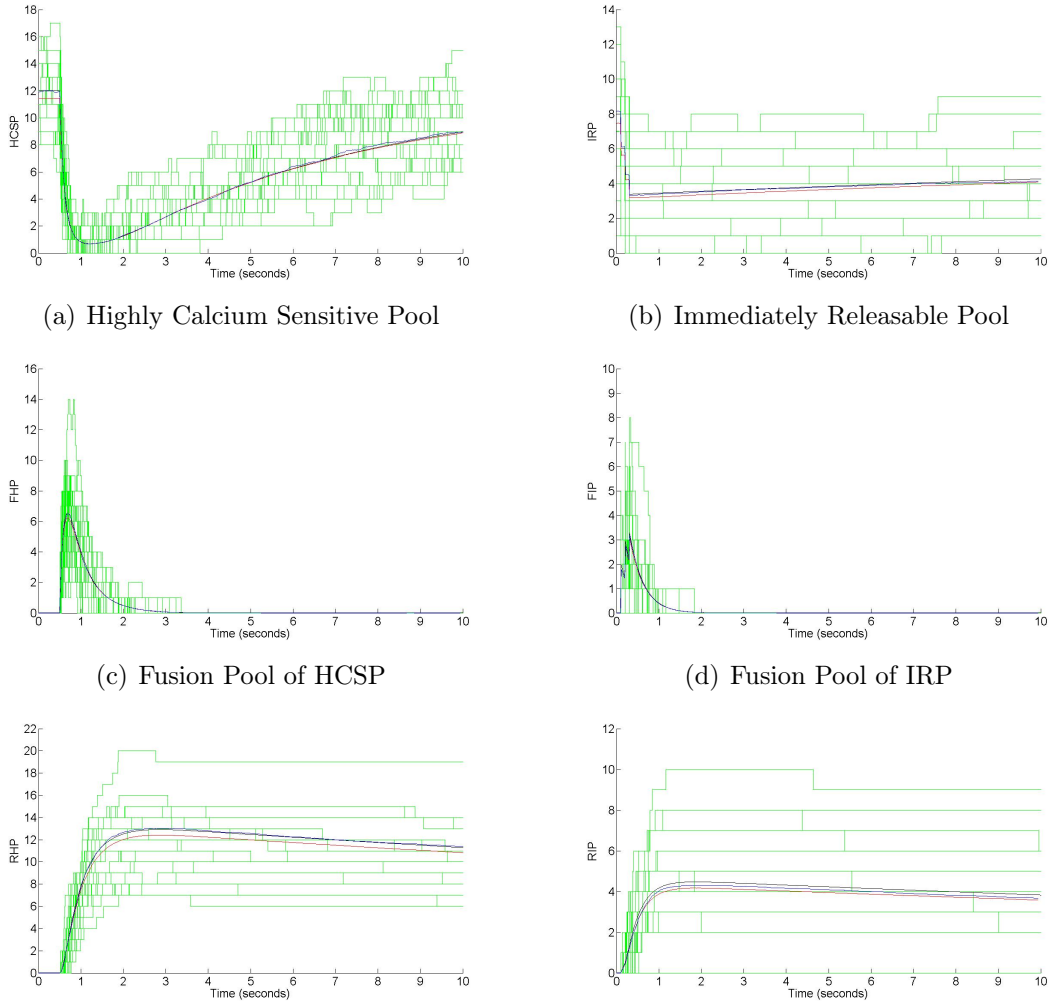


(b) Docked Pool



(c) Primed Pool

Figure 5.11: Results for the fast protocol with close fits of the calcium equations and random initial conditions for each run. Evolution of AP, DP and PP for the 3 square pulse and protocol over 10 seconds. Mean (red) of AP, DP and PP over 5000 runs and variance (blue) are seen to lie close to each other. Also overlaid are 20 stochastic runs (green) and the deterministic solution (black).



(a) Highly Calcium Sensitive Pool

(b) Immediately Releasable Pool

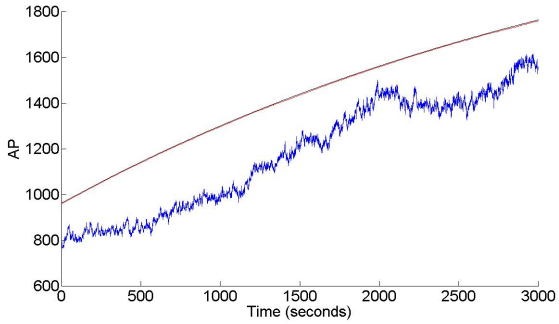
(c) Fusion Pool of HCSP

(d) Fusion Pool of IRP

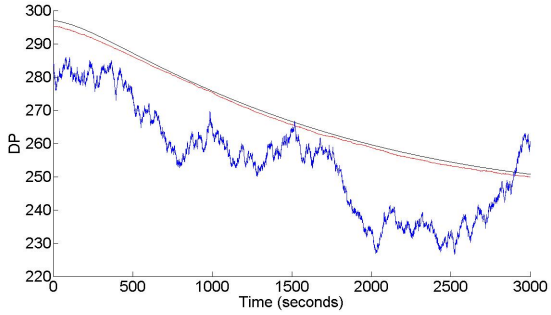
(e) Releasing Pool of HCSP

(f) Releasing Pool of IRP

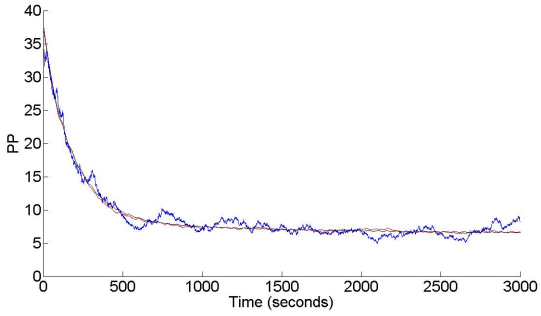
Figure 5.12: Results for the fast protocol with close fits of the calcium equations and random initial conditions for each run. Evolution of HCSP and IRP chain for the 3 square pulse protocol over 10 seconds. HCSP pathway responds to the the sudden rise of the cytosolic calcium at t = 0.5 seconds. Mean (red) of HCSP, IRP, FHP, FIP, RHP and RIP over 5000 runs and variance (blue) are seen to lie close to each other. Also overlaid are 20 stochastic runs (green) and the deterministic solution (black).



(a) Almost Docked Pool

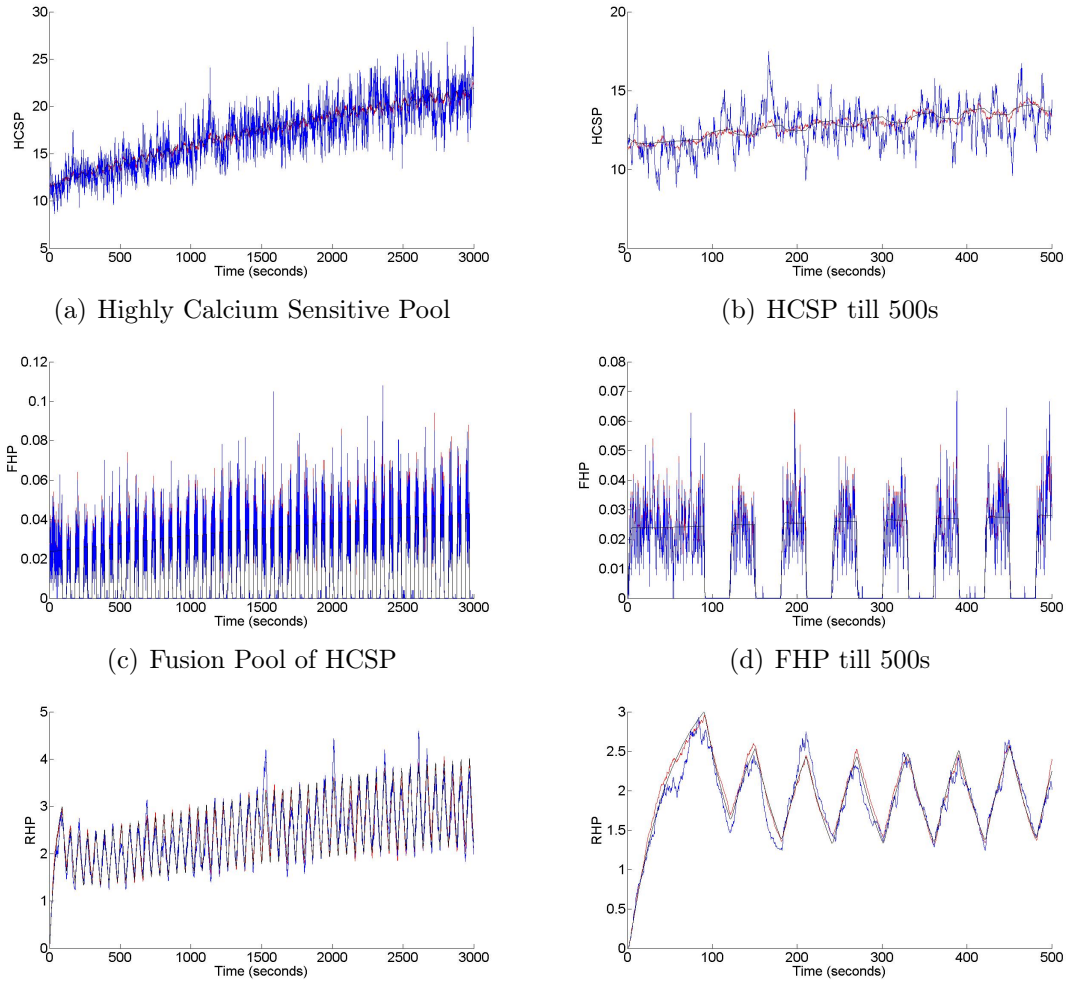


(b) Docked Pool



(c) Primed Pool

Figure 5.13: Evolution of AP, DP and PP for the slow depolarisation protocol over 3000 seconds as shown in Fig. 5.5 for 500 stochastic runs . Mean (red) of AP, DP and PP over 500 runs and the deterministic solution are seen to lie close to each other. Variance (blue) is slightly away.



(a) Highly Calcium Sensitive Pool

(b) HCSP till 500s

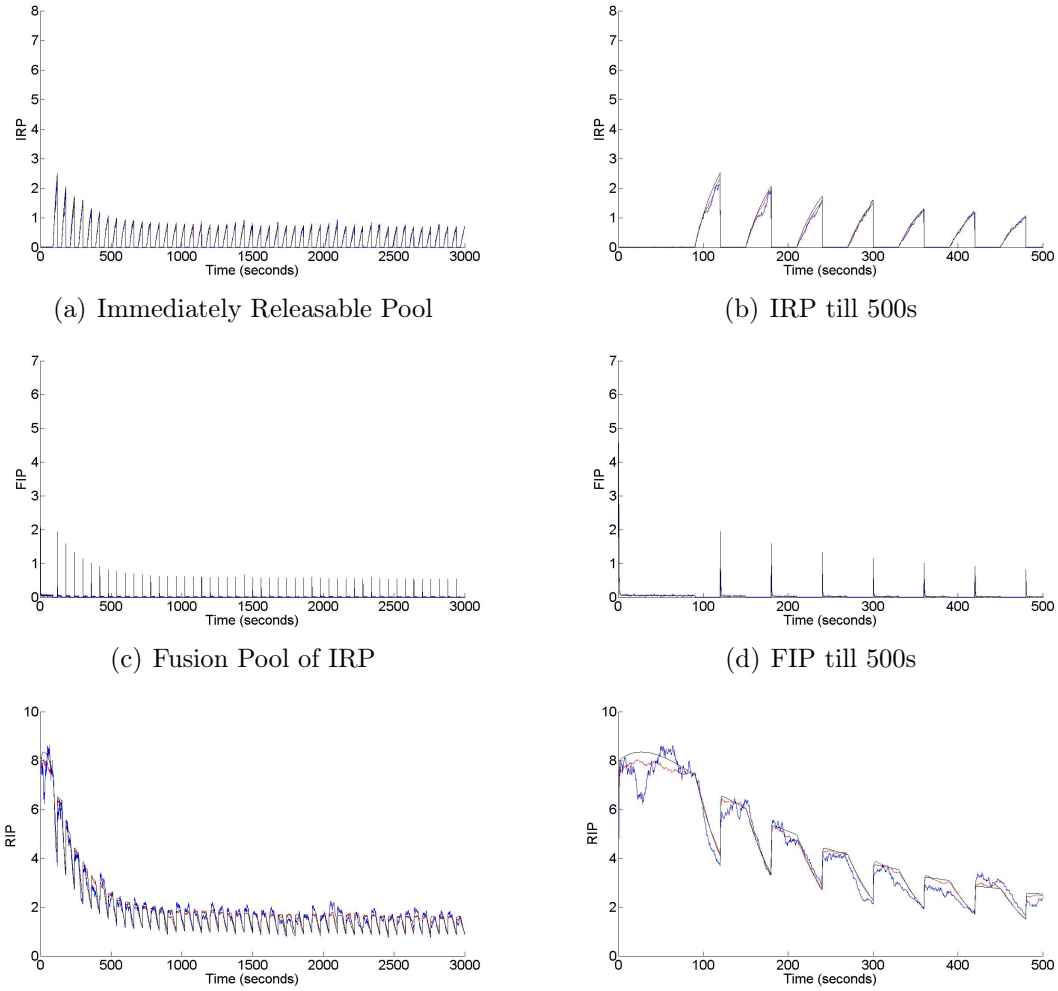
(c) Fusion Pool of HCSP

(d) FHP till 500s

(e) Releasing Pool of HCSP

(f) RHP till 500s

Figure 5.14: Evolution of HCSP, FHP and RHP for the slow depolarisation protocol over 3000 seconds as shown in Fig. 5.5 for 500 stochastic runs. (b), (d) and (f) are the zoomed view of HCSP, FHP and RHP respectively shown till 500 seconds.



(a) Immediately Releasable Pool

(b) IRP till 500s

(c) Fusion Pool of IRP

(d) FIP till 500s

(e) Releasing Pool of IRP

(f) RIP till 500s

Figure 5.15: Evolution of IRP, FIP and RIP for the slow depolarisation protocol over 3000 seconds as shown in Fig. 5.5 for 500 stochastic runs. (b), (d) and (f) are the zoomed view of IRP, FIP and RIP respectively shown till 500 seconds.

Chapter 6

Discussion

The primary aim of the project was to develop a stochastic model of glucose-stimulated exocytosis adapted from Pederson et al. to account for integral copy numbers of the granules instead of concentrations. The equations were solved with a Gillespie stochastic simulation algorithm modified to include the time-dependent propensities. The simulations recover the deterministic solution in the mean and the variance that is expected.

6.1 The hybrid algorithm

For carrying out the stochastic simulations, a hybrid Gillespie algorithm is used which is modified to include the time-dependent propensities (in Section. 2.2). In the model of Insulin granule pools described by Pederson et al. [1], some of the rates are dependent on the concentration of calcium and hence dependent on time. The rates r_5, r_3, r_2 and $f_H(C_i)$ are dependent on the concentration of cytosolic calcium and the rate $f_I(C_{md})$ is dependent on the concentration of microdomain calcium which makes the total propensity a function of time.

The difference in the usual and hybrid Gillespie SSA is the step to calculate time of occurrence of the next reaction t_{next}. The mean and variance over stochastic simulations using hybrid Gillespie algorithm show expected results.

6.2 Verification

For verifying the correctness of the deterministic solution, the solution for low pool sizes for the IRP chain of the model was compared with the solution of the Master equation. Both the solutions match up exactly showing that the code for producing the deterministic solution is correct.

We have also checked the algorithm on a trivial example with a time dependent rate and compared the solutions of usual Gillespie SSA and the solutions of hybrid Gillespie SSA. Both the solutions match up for both the species A and B as expected. For A both the solutions are also shown matching up with the analytical solution for A. For B the analytical solutions has not been calculated as it is non-trivial.

Previous trials of simulating the model with Gillespie algorithm did not produce the correct results. Also, for this model the usual Gillespie algorithm can't be applied for the given initial conditions as the total propensity goes to 0 at some point of time which causes $t_{next} = \infty$. When the model was simulated using hybrid Gillespie SSA the discrepancies in the IRP chain of the model were more prominent than the other pools. Instead solving the involved ordinary differential equations using MATLAB ode solver, we used Euler's method to solve the ODE. Comparison of the stochastic solution with the deterministic solution was done for different $f_I(C_{md})$ functions showing that for constant functions and step functions the solutions match up very closely. Also for the functions with square pulses the stochastic solution diverges from the deterministic solution as the width of the pulse is decreased. The reason for the prominent discrepancies in the IRP chain of the model can be the dependence on the microdomain calcium with very fine spikes due to which the spikes are not detected by the hybrid Gillespie SSA and less run are contributing to the mean. We also show that changing the Euler time step and increasing the number of runs the mean over deterministic solution can be smoothed and can be made more closer to the deterministic solution.

6.3 Results

In Chapter 4, we have shown the analytical solution for the mean and variance of the open and closed system. For the close system we see the mean for the reactant specie AP starting from the initial condition and settling to 0, and the variance starts from 0 and settles with the mean to 0 as expected. For the open system i.e AP coupled to a infinite pool RP, if Var[AP] = 0 at t = 0 and the mean starts with the initial condition both mean and variance settle to a non-zero equilibrium value near r_1/r_2 as expected. It can be shown analytically that if the value of $Var[AP] \neq 0$ at t = 0 or mean is started from some other value than the initial conditions, the results will still settle near the value r_1/r_2.

In chapter 5, we show the stochastic results for the fast and slow protocol described by Pederson et al. (2009) in Chapter 5. For both the cases the calcium compartment functions used are the close fits of the Arthur's Sherman representation of the calcium compartment equations of the Insulin granule compartments model. We use the close fits for simplicity and decreasing the run time. Also for all the pools we have used random initial conditions for each stochastic run selected from a normal distribution where the mean is the size of the pool. This causes the variannce to start from the steady state instead of 0. The mean is slightly below the deterministic solution as we use floor values of the random initial conditions in order to select discrete numbers.

We also show the comparison of the deterministic solution produced using the fits of the actual calcium compartment equations and the deterministic solution by Pederson et al. For all the pools except PP and IRP chain of the model, both the solutions follow the same trend with some difference because of different initial conditions. For PP the deterministic solution using the close fits is slightly away beacause of coupling to IRP. As we have used square pulses instead of peaks for microdomain calcium and IRP chain depends on microdomain calcium the difference is prominent in the IRP chain and PP.

For the fast protocol, when the model was simulated using the actual calcium compartment equations the mean over stochastic runs was not following the trend of the deterministic solution only for the IRP chain of the model. For all the other pools the mean over stochastic runs was similar to the mean when simulated using the calcium equation fits. As mentioned in above section that the reason could be the dependence of IRP chain on the microdomain calcium with very fine spikes. There is a possibility that the variation in the stochastic solution from the deterministic

solution can be decreased by increasing the number of runs or decreasing the Euler time step (not tested).Simulating the model with fast protocol using the fits of the actual calcium equations we see the mean, variance and the deterministic solution matching up closely as expected.

For the slow protocol, the width of the pulses are very large and are continued for a long time. The code for the slow protocol is modified from that of the fast protocol. We show the mean matching up with the deterministic solution, and also the variance following the trend, but not very smoothly. We have calculated the mean over 1000 stochastic runs for slow protocol. The variance could be smoothed by increasing the number of runs.

In our knowledge this study of the stochastic version of the model of insulin secretion from pancreatic islets of Langerhans hasn't be done before. We achieve the goal of presenting the stochastic version of the model and getting the estimate of the integral copy numbers of the granules. The algorithm can be optimized using other languages and modified to speed up the simulations. Also, different method such as Tau leaping method can be used and tested for faster stochastic simulations. In Appendix: Code, we present the code for the hybrid Gillespie stochastic simulation algorithm which can be implemented to produce the stochastic mean and variance of other chemical and biochemical systems.

Bibliography

[1] Morten Gram Pederson and Arthur Sherman, “Newcomer insulin secretory granules as a highly calcium-sensitive pool", PNAS, Vol. 106, 7432-7436, May 2009.

[2] Daniel T. Gillespie, “Exact stochastic simulation of coupled chemical reactions", Research Department, Na Val Weapons center , China lake, California 93555, 1997.

[3] Aurelien Alfonsi, et al.,“Exact simulation of hybrid stochastic and deterministic models for biochemical systems", ISRN INRIA, RR-5435, December 2004.

[4] G.Iyengar, Lecture notes on inverse transform method, IEOR E4404, February 2002.

[5] Michael A. Gibson and Jehoshua Bruck,“Efficient exact stochastic simulation of chemical systems with many species and many channels", J. Phys. Chem. A 2000, 104, 1876-1889, December 1999.

[6] David G. Kendall,“An artificial realization of a simple birth-and-death process", JSTOR, Vol. 12, No. 1 (1950), pp. 116-119, November 1949.

[7] Shantanu Kadam and Kumar Vanka,“A new approximate method for the Stochastic Simulation of Chemical Systems:The Representative Reaction approach", Journal of Computational Chemistry, 33:276-285, 2012.

[8] Daniel T. Gillespie,“A general method for numerically simulating the stochastic time evolution of coupled chemical reactions", Research Department, Na Val Weapons center , China lake, California 93555, April 1976.

[9] Radek Erban, S. Jonathan Chapman and Philip K. Maini,“ A practical guide to stochastic simulations of reactio-diffusion processes", arXiv:0704.1908v2 [q-bio.SC], Nov 2007.

[10] Nicolas Wieder, et al.,“Exact and approximate stochastic simulation of intracellular calcium dynamics", Journal of Biomedicine and Biotechnology, Article ID 572492, August 2011.

[11] David F. Anderson,"A modified next reaction method for simulating chemical systems with time dependent propensities and delays", August 2007.

[12] Nicolas Wieder, et al.,"Exact and approximate stochastic simulation of intracellular calcium dynamics", Journal of Biomedicine and Biotechnology, Article ID 572492, August 2011.

[13] Henquin J C, et al.,"Invivo and in vitro glucose-induced biphasic insulin secretion in the mouse: pattern and role of cytoplasmic Ca_{2+} and amplification signals in beta-cells", Diabetes, 55:441-451, 2006.

[14] Chen Y, Wang S, Arthur Sherman,"Identifying the targets of the amplifying pathway for insulin secretion in pancreatic beta-cells by kinetic modeling of granule exocytosis", Biophys J, 95:2226-2241, 2008.

[15] Rorsman P and Renstorm E,"Insulin granule dynamics in pancreatic beta cells", Diabetologia, 46:1029-1045, 2003.

[16] Barg S, et al.,"Fast exocytosis with few Ca^{2+} channels in insulin-secreting mouse pancreatic B cells", Biophys J, 81:3308-3323, 2001.

[17] Henquin J-C ,"Triggering and amplifying pathways of regulation of insulin secretion by glucose", Diabetes, 49:1751-1760, 2000.

[18] Barg S, et al.,"Delay between fusion pore opening and peptide release from large dense-core vesicles in neuroendocrine cells", Neuron, 33:287-299, 2002.

[19] Yang Y and Gillis K D,"A highly Ca^{2+} sensitive pool of granules is regulated by glucose and protien kinases in insulin-secreting INS-1 cells", J Gen Physiol, 124:641-651, 2004.

[20] Ohara Imaizumi M et al.,"TIRF imaging of docking and fusion of single insulin granule motion in primary rat pancreatic beta-cells: Different behaviour of granule motion between normal and Goto-Kakizaki diabetic rat beta-cells", Biochem J, 381:13-18, 2004.

[21] Curry D L, et al., "Dynamics of insulin secretion by the perfused rat pancreas", Endocrinology, 83:572-584, 1968.

[22] Kasai K, et al., "Docking is not a prerequisite but a temporal constraint for fusion of secretory granules", Traffic, 9:1191-1203, 2008.

[23] Wan Q F, et al., "Protien Kinase activation increases insulin secretion by sensitizing the secretory machinery to Ca^{2+}", J Gen Physiol, 124:653-652, 2004.

[24] M. Ullah, H. Schmidt, K.-H. Cho and O. Wolkenhauer, "Deterministic modelling and stochastic simulation of biochemical pathways using MATLAB", IEE Proc.-Syst. Biol., Vol. 153, No. 2, March 2006.

Appendix: Supporting Information

The main source of the supporting information is the Arthur Sherman's description of the Insulin secretion model by Pederson et al. (2009) [1].
Ordinary differential equations corresponding to each pool except RP as it is considered to be infinity [1]. Rates are measured in seconds. The rates r_5, r_3, r_2 and $f_H(C_i)$ are cytosolic calcium dependent and $f_I(C_{md})$ is microdomain calcium dependent.

$$
\begin{aligned}
IRP' &= (r_1\,PP - r_{-1}\,IRP - f_I(C_{md})\,IRP) && (1)\\
PP' &= (r_{-1}\,IRP - (r_1 + r_{-2})\,PP + r_2\,DP) && (2)\\
DP' &= (r_3\,HCSP + r_{-2}\,PP - (r_{-3} + r_2)\,DP) && (3)\\
AP' &= (r_5 - r_{-5}\,AP - r_4\,AP + r_{-4}\,HCSP) && (4)\\
HCSP' &= (r_4\,AP - (r_{-4} + r_3)\,HCSP + r_{-3}\,DP - f_H(C_i)\,HCSP) && (5)\\
FIP' &= (f_I(C_{md})\,IRP - u_2\,FIP) && (6)\\
RIP' &= (u_2\,FIP - u_3\,RIP) && (7)\\
FHP' &= (f_H(C_i)\,HCSP - u_2\,FHP) && (8)\\
RHP' &= (u_2\,FHP - u_3\,RHP) && (9)
\end{aligned}
$$

where,

$$
\begin{aligned}
r_2 &= r_{20}\,\tfrac{C_i}{C_i + Kp2} && (10)\\
r_3 &= r_{30}\,\tfrac{C_i}{C_i + Kp} && (11)\\
r_5 &= r_{50}\,\tfrac{C_i}{C_i + Kp} && (12)
\end{aligned}
$$

Fusion rate from IRP ($f_I(C_{md})$) and HCSP ($f_H(C_i)$) follow hill functions.

$$f_I(C_{md}) = f_I^{max} \frac{C_{md}^n}{K_I^n + C_{md}^n} \tag{13}$$

$$f_H(C_i) = f_H^{max} \frac{C_i^n}{K_H^n + C_i^n} \tag{14}$$

Microdomain and cytosolic calcium compartments are modelled and described by

$$C'_{md} = (-f_{md}\, J_L - f_{md}\, B\, (C_{md} - C_i)) \tag{15}$$

$$C'_i = (-f_i\, J_R + f_v\, f_i\, B\, (C_{md} - C_i) - f_i\, L) \tag{16}$$

Molar fluxes through L-type and R-type channels are

$$J_L = \alpha\, I_L / v_{md}, \tag{17}$$

$$J_R = \alpha\, I_R / v_{cell} \tag{18}$$

with respective currents

$$I_L = g_L\, m_\infty(v)\, (V - V_{Ca}), \tag{19}$$

$$I_R = g_R\, m_\infty(v)\, (V - V_{Ca}) \tag{20}$$

where,

$$m_\infty(v) = 1/(1 + exp((V_m - V)/s_m)). \tag{21}$$

Calcium pumps and stores fluxes are given by

$$J_{serca} = J_{serca}^{max} \frac{C_i^2}{K_{serca}^2 + C_i^2}, \tag{22}$$

$$J_{pmca} = J_{pmca}^{max} \frac{C_i}{K_{pmca} + C_i}, \tag{23}$$

$$J_{ncx} = J_{ncx0}\, (C_i - 0.25), \tag{24}$$

$$L = J_{serca} + J_{pmca} + J_{ncx} + J_{leak}. \tag{25}$$

Table 1 shows the random initial conditions corresponding to each pool and Table 2 shows the initial microdomain and cytosolic calcium concentrations [1].
Instead of the steady state values, for stochastic simulations we used random and discrete initial values which are choosen from the normal distribution where the mean is the size of the pool.
For IRP, the expression implies that the initial condition for each run is selected from a normal distribution were mean = 8 and the standard deviation = $\sqrt{8}$. As we need discrete values, we use the 'floor' condition . Same follows for the other pools.

Table 1

Pool	**Mean initial conditions**
IRP	floor(max(normrnd(8,$\sqrt{8}$)))
PP	floor(max(normrnd(38,$\sqrt{38}$)))
DP	floor(max(normrnd(298,$\sqrt{298}$)))
FIP	0
RIP	0
AP	floor(max(normrnd(965,$\sqrt{965}$)))
HCSP	floor(max(normrnd(12,$\sqrt{12}$)))
FHP	0
RHP	0

Table 2

Calcium domain	**Value**
C_{md}	0.0674
C_i	0.06274

Table 3 contains values of all the parameters in the Insulin granule compartments model. All the parameters are in reference with the Arthur Sherman's description of the model.

Table 3

Vesicle dynamics parameters, fusion constants, calcium currents and calcium fluxes.

Parameter	**Value**	**Parameter**	**Value**
$ficmd(0)$	2.641e-9 s^{-1}	f_H^{max}	30 s^{-1}
$fhci(0)$	0.00001189 s^{-1}	K_H	2.5 μM
r_1	0.005 s^{-1}	g_L	150 pS
r_{-1}	0.025 s^{-1}	g_R	150 pS
r_{20}	0.00015 s^{-1}	V_m	-20 mV
r_{-2}	0.001 s^{-1}	V_{Ca}	25 mV
r_{30}	0.002 s^{-1}	S_m	5 mV
r_{-3}	0.00007 s^{-1}	J_{Serca}^{max}	41 $\mu M/s$
r_4	0.002 s^{-1}	K_{Serca}	0.27 μM
r_{-4}	0.16 s^{-1}	J_{pmca}^{max}	21 $\mu M/s$
r_{50}	0.224 s^{-1}	K_{pmca}	0.5 μM
r_{-5}	0.0002 s^{-1}	J_{leak}	-0.94 $\mu M/s$
u_1	2000 s^{-1}	J_{ncx0}	18.67 s^{-1}
u_2	3 s^{-1}	f_{md}	0.01
u_3	0.02 s^{-1}	f_i	0.01
kp	0.01	B	17250 s^{-1}
$kp2$	0.01	α	5.18e-15 $\mu mol/s/fA$
f_I^{max}	30 s^{-1}	v_{cell}	1.15e-12pl
K_I	22 μM	v_{md}	0.00385e-15 pl
n	4	f_v	vmd/vcell

Appendix: Code

The code for the stochastic simulation using hybrid Gillespie algorithm. Programming work is done in MATLAB. Euler's method is used to solve the differential equations involved. The working of the program is explained below. Code is for the fast protocol. For the slow protocol replace the cmd and ci functions as mentioned in Fig. fitupdown calcium, increase tfinal to 3000s and replace r5 with r5 = gluc*r50*(ci/(ci+0.01)) where, gluc = 3 for t $>$ 10s and gluc = 1 for t$<$ 10s.

Steps for producing the results are as follows:
1. Run deterministic.m to calculate and plot the deterministic solution.
2. Run gelldatafile.m to create the data files for all the pools.
3. Read the files.
T=dlmread('datafile name.txt');
4. Run interpolation.m to calculate and plot mean and variance.
Ta=interpolation(T,1024);

File name = deterministic.m (MATLAB file)
Script to calculate and plot the deterministic solution. Euler Method is used to solve the differential equations involved. Calcium functions are taken to be close fits of the Arthur Sherman's description of the calcium compartments in Insulin granule pools model.

```
%Initialization of start time and stop time of the simulation:
tstart = 0;
tfinal = 10;

%Initialization Vectors for time, pools and calcium compartments:
time     = [];
```

```
rps      = [];
aps      = [];
hcsps    = [];
fhps     = [];
rhps     = [];
dps      = [];
pps      = [];
irps     = [];
fips     = [];
rips     = [];
cmds     = [];
cis      = [];

%Initial values of the pools (granule numbers) and calcium compartments:
rp    = 1;                      %rp is considered to be infinity. Hence,
                                ...it is not updated.
ap    = 965;
hcsp  = 12;
fhp   = 0;
rhp   = 0;
dp    = 297;
pp    = 38;
irp   = 8;
fip   = 0;
rip   = 0;
cmd   = 0.0674;
ci    = 0.06274;

%Parameters: Rates (/s):
r50   = 0.224;
rm5   = 0.0002;
r4    = 0.002;
rm4   = 0.16;
r30   = 0.002;
```

```
rm3   = 0.00007;
r20   = 0.00015;
rm2   = 0.001;
r1    = 0.005;
rm1   = 0.025;
fhci  = 0;
ficmd = 0;
u2    = 3;
u3    = 0.02;

%Fusion constants:
fim = 30;    %/s
fhm = 30;    %/s
ki  = 22;    %microM
kh  = 2.5;   %microM
n   = 4;

%Accumulating the initial values:
time     = [time;tstart];
rps      = [rps;rp];
aps      = [aps;ap];
hcsps    = [hcsps;hcsp];
fhps     = [fhps;fhp];
rhps     = [rhps;rhp];
dps      = [dps;dp];
pps      = [pps;pp];
irps     = [irps;irp];
fips     = [fips;fip];
rips     = [rips;rip];
cmds     = [cmds;cmd];
cis      = [cis;ci];

d = 0.0001;                %Euler time step.
```

```
t = tstart;                %Initializing t as tstart.

while(t<tfinal)            %Each simulation will run until t vector
                           ...reaches tfinal.

    %Cmd - function with three square pulses
    ...at t = 0.1, 0.2 and 0.3 secs for 0.01 secs:
    if (t>=0 && t<0.1)
        cmd = 0.1132;
    elseif(t>=0.1 && t<0.11)
        cmd = 48.13;
    elseif (t>=0.11 && t<0.2)
        cmd = 0.1132;
    elseif (t>=0.2 && t<0.21)
        cmd = 48.13;
    elseif (t>=0.21 && t<0.3)
        cmd = 0.1132;
    elseif (t>=0.3 && t<0.31)
        cmd = 48.13;
    else
        cmd = 0.1132;
    end

    %Ci - function with the value of ci raised to 2 at 0.5 secs:
    if (t<0.5)
        ci = 0.06419;
    else
        ci = 1.299*exp(-((t-(-0.6304))/1.2)^2)...
         + (3.285e+008)*exp(-((t-(-1036))/220.8)^2)...
          + 1.375*exp(-((t-0.4701)/2.028)^2);
    end

    t = t + d;             %Integrating time using Euler's method with a
                           ...Euler time step d.
```

```
%Time-dependent rates:
r5    = r50*(ci/(ci+0.01));
r3    = r30*(ci/(ci+0.01));
r2    = r20*(ci/(ci+0.01));
ficmd = fim*((cmd^n)/((cmd^n)+(ki^n)));
fhci  = fhm*((ci^n)/((ci^n)+(kh^n)));

%Differential equations corresponding to each pool:
drp   = -(r5)+(rm5*ap);
dap   = (r5)-(rm5*ap)-(r4*ap)+(rm4*hcsp);
dhcsp = (r4*ap)-(rm4*hcsp)-(fhci*hcsp)-(r3*hcsp)+(rm3*dp);
dfhp  = (fhci*hcsp)-(u2*fhp);
drhp  = (u2*fhp)-(u3*rhp);
ddp   = (r3*hcsp)-(rm3*dp)-(r2*dp)+(rm2*pp);
dpp   = (r2*dp)-(rm2*pp)-(r1*pp)+(rm1*irp);
dirp  = (r1*pp)-(rm1*irp)-(ficmd*irp);
dfip  = (ficmd*irp)-(u2*fip);
drip  = (u2*fip)-(u3*rip);

%Updating the pool vectors:
rp    = rp;                %rp is not updated, as it is considered
                           ...to be infinity.
ap    = ap+dap*d;
hcsp  = hcsp+dhcsp*d;
fhp   = fhp+dfhp*d;
rhp   = rhp+drhp*d;
dp    = dp+ddp*d;
pp    = pp+dpp*d;
irp   = irp+dirp*d;
fip   = fip+dfip*d;
rip   = rip+drip*d;

%Accumulating the values for all the pools and calcium compartments
```

```
    ...as evolve in time:
    time     = [time;t];
    rps      = [rps;rp];
    aps      = [aps;ap];
    hcsps    = [hcsps;hcsp];
    fhps     = [fhps;fhp];
    rhps     = [rhps;rhp];
    dps      = [dps;dp];
    pps      = [pps;pp];
    irps     = [irps;irp];
    fips     = [fips;fip];
    rips     = [rips;rip];
    cmds     = [cmds;cmd];
    cis      = [cis;ci];

end
%Plotting the pool concentrations and calcium concentrations with time:
%AP
figure(1)
hold on
xlabel('Time (seconds)');
ylabel('AP');
plot(time,aps,'r')

%Similarly plot for other pools and calcium concentrations.
```

File name = hybridgell.m (MATLAB file)
Script for calculating the stochastic solution using hybrid Gillespie solution. Euler method is used to solve the differential equations with a Euler time step of e^{-4}. Function hybridgell.m returns the values to gelldatafile.m.

```
function [tnexts rps aps hcsps fhps rhps dps pps irps fips rips...
 cmds cis] = hybridgell
```

```
global tstart
global tfinal

tstart = 0;

%Initialization of time of next reaction:
tnext  = 0;

%Initialization Vectors:
tnexts   = [];
rps      = [];
aps      = [];
hcsps    = [];
fhps     = [];
rhps     = [];
dps      = [];
pps      = [];
irps     = [];
fips     = [];
rips     = [];
cmds     = [];
cis      = [];

%Initial values of the pools and calcium compartments. Here, the
...initial values are random and belong to a normal distribution
...where mean = size of the pool. As discrete values are required,
...'floor' is used.
rp    = 1;
ap    = floor(max(normrnd(965,sqrt(965)),0));
hcsp  = floor(max(normrnd(12,sqrt(12)),0));
fhp   = 0;
rhp   = 0;
dp    = floor(max(normrnd(297,sqrt(297)),0));
pp    = floor(max(normrnd(38,sqrt(38)),0));
```

```
irp   = floor(max(normrnd(8,sqrt(8)),0));
fip   = 0;
rip   = 0;
cmd   = 0.0674;
ci    = 0.06274;

%Parameters: Rates (/s):
r50   = 0.224;
rm5   = 0.0002;
r4    = 0.002;
rm4   = 0.16;
r30   = 0.002;
rm3   = 0.00007;
r20   = 0.00015;
rm2   = 0.001;
r1    = 0.005;
rm1   = 0.025;
fhci  = 0;
ficmd = 0;
u2    = 3;
u3    = 0.02;

%Fusion constants:
fim = 30;    %/s
fhm = 30;    %/s
ki  = 22;    %microM
kh  = 2.5;   %microM
n   = 4;

%Accumulating the initial values:
tnexts   = [tnexts;tstart];
rps      = [rps;rp];
aps      = [aps;ap];
hcsps    = [hcsps;hcsp];
```

```
fhps     = [fhps;fhp];
rhps     = [rhps;rhp];
dps      = [dps;dp];
pps      = [pps;pp];
irps     = [irps;irp];
fips     = [fips;fip];
rips     = [rips;rip];
cmds     = [cmds;cmd];
cis      = [cis;ci];

while (tnext<tfinal)  %Each simulation will run until tnext reaches
                      ...tfinal.
    u = rand(1,1);    %First random number for calculating tnext.

    d = 0.0001;       %Euler time step.
    x = 0;            %Dummy Variable.
    event = 0;        %Event checking variable.
    t = tstart;       %Initializing t vector as tstart.

    while(t<tfinal && event==0)

        %Cmd - function with three square pulses at
        ...t = 0.1, 0.2 and 0.3secs for 0.01 secs:
        if (t>=0 && t<0.1)
                cmd = 0.1132;
        elseif(t>=0.1 && t<0.11)
                cmd = 48.13;
        elseif (t>=0.11 && t<0.2)
                cmd = 0.1132;
        elseif (t>=0.2 && t<0.21)
                cmd = 48.13;
        elseif (t>=0.21 && t<0.3)
                cmd = 0.1132;
        elseif (t>=0.3 && t<0.31)
```

```
        cmd = 48.13;
else
        cmd = 0.1132;
end

%Ci - function with the value of ci raised to 2 at 0.5 secs:
if (t<0.5)
    ci = 0.06419;
else
    ci = 1.299*exp(-((t-(-0.6304))/1.2)^2)...
     + (3.285e+008)*exp(-((t-(-1036))/220.8)^2)...
      + 1.375*exp(-((t-0.4701)/2.028)^2);
end

t = t + d;       %Integrating time with a Euler time step d.

%Time-dependet rate; r5, r3, r2 and fhci depend on ci and
...ficmd depends on cmd:
r5    = r50*(ci/(ci+0.01));
r3    = r30*(ci/(ci+0.01));
r2    = r20*(ci/(ci+0.01));
ficmd = fim*((cmd^n)/((cmd^n)+(ki^n)));
fhci  = fhm*((ci^n)/((ci^n)+(kh^n)));

%Propensities:
p(1) = r5;             %rp    --r5-->  ap
p(2) = rm5*ap;         %ap    --rm5->  rp
p(3) = r4*ap;          %ap    --r4-->  hcsp
p(4) = rm4*hcsp;       %hcsp  --rm4->  ap
p(5) = fhci*hcsp;      %hcsp  --fhci->  fhp
p(6) = u2*fhp;         %fhp   --u2-->  rhp
p(7) = u3*rhp;         %rhp   --u3-->
p(8) = r3*hcsp;        %hcsp  --r3-->  dp
p(9) = rm3*dp;         %dp    --rm3->  hcsp
```

```
p(10)= r2*dp;          %dp    --r2-->  pp
p(11)= rm2*pp;         %pp    --rm2->  dp
p(12)= r1*pp;          %pp    --r1-->  irp
p(13)= rm1*irp;        %irp   --rm1->  pp
p(14)= ficmd*irp;      %irp   -ficmd->  fip
p(15)= u2*fip;         %fip   --u2-->  rip
p(16)= u3*rip;         %rip   --u3-->

%Reaction intervals:
p1   = p(1)/sum(p);
p2   = p(2)/sum(p);
p3   = p(3)/sum(p);
p4   = p(4)/sum(p);
p5   = p(5)/sum(p);
p6   = p(6)/sum(p);
p7   = p(7)/sum(p);
p8   = p(8)/sum(p);
p9   = p(9)/sum(p);
p10  = p(10)/sum(p);
p11  = p(11)/sum(p);
p12  = p(12)/sum(p);
p13  = p(13)/sum(p);
p14  = p(14)/sum(p);
p15  = p(15)/sum(p);
p16  = p(16)/sum(p);

%Initializing update vectors:
z1 = 0;
z2 = 0;
z3 = 0;
z4 = 0;
z5 = 0;
z6 = 0;
z7 = 0;
```

```
z8 = 0;
z9 = 0;
z10= 0;
z11= 0;
z12= 0;
z13= 0;
z14= 0;
z15= 0;
z16= 0;

x = x + (sum(p)*d);     %Integrating dummy variable using
                        ...Euler's method with a Euler time
                        ...step d = e-4

%Event checking:
if (x >= -log(u))       %Event occurs when  integration of
                        ...dummy variable reaches -ln(u).

    event=1;
    x = 0;              %Setting x again to 0 after an event
                        ...has occurred

    k = rand(1,1);      %Second random number for selecting
                        ...the reaction.

    %Reaction selection:
    if (k<1)
        if (k<p1)
           z1 = 1;
        end
        if (k>p1 && k<p1+p2)
            z2 = 1;
        end
        if (k>p1+p2 && k<p1+p2+p3)
```

```
    z3 = 1;
end
if (k>p1+p2+p3 && k<p1+p2+p3+p4)
    z4=1;
end
if (k>p1+p2+p3+p4 && k<p1+p2+p3+p4+p5)
    z5=1;
end
if (k>p1+p2+p3+p4+p5 && k<p1+p2+p3+p4+p5+p6)
    z6=1;
end
if (k>p1+p2+p3+p4+p5+p6 && k<p1+p2+p3+p4+p5+p6+p7)
    z7=1;
end
if (k>p1+p2+p3+p4+p5+p6+p7 &&...
 k<p1+p2+p3+p4+p5+p6+p7+p8)
    z8=1;
end
if (k>p1+p2+p3+p4+p5+p6+p7+p8 &&...
 k<p1+p2+p3+p4+p5+p6+p7+p8+p9)
    z9=1;
end
if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9 &&...
 k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10)
    z10=1;
end
if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10 &&...
 k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11)
    z11=1;
end
if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11 &&...
 k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12)
    z12=1;
end
```

```
            if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12 &&...
             k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13)
                z13=1;
            end
            if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13 &&...
             k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13+p14)
                z14=1;
            end
            if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13+p14 &&...
             k<p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13+p14+p15)
                z15=1;
            end
            if (k>p1+p2+p3+p4+p5+p6+p7+p8+p9+p10+p11+p12+p13+p14+p15)
                z16=1;
            end
        end

        %Updating the pool values:
        rp   = rp;       %rp is not updated,
                         ...as it is considered to be infinity.
        ap   = ap  +z1-z2-z3+z4;
        hcsp = hcsp+z3-z4-z5-z8+z9;
        fhp  = fhp +z5-z6;
        rhp  = rhp +z6-z7;
        dp   = dp  +z8-z9-z10+z11;
        pp   = pp  +z10-z11-z12+z13;
        irp  = irp +z12-z13-z14;
        fip  = fip +z14-z15;
        rip  = rip +z15-z16;

    else
        rp   = rp;
        ap   = ap;
        hcsp = hcsp;
```

```
            fhp  = fhp;
            rhp  = rhp;
            dp   = dp;
            pp   = pp;
            irp  = irp;
            fip  = fip;
            rip  = rip;

        end
    end

    %Initializing tstart as tnext:
    tnext=t;
    tstart=tnext;

    %Accumulating the values for the pools and calcium compartments:
    tnexts   = [tnexts;tnext];
    rps      = [rps;rp];
    aps      = [aps;ap];
    hcsps    = [hcsps;hcsp];
    fhps     = [fhps;fhp];
    rhps     = [rhps;rhp];
    dps      = [dps;dp];
    pps      = [pps;pp];
    irps     = [irps;irp];
    fips     = [fips;fip];
    rips     = [rips;rip];
    cmds     = [cmds;cmd];
    cis      = [cis;ci];

end
end
```

File name = gelldatafile.m (MATLAB file)

Script to write the concentrations of all the pools and tnext in data files and converting saw tooth form into square form. Data is written row wise in respective data files where for the single run first row contains all the tnext values upto tfinal, and the second row contains the values corresponding to each tnext and similarly other runs are appended row wise in the file. Saw-tooth form is written in the data files. Individual stochastic runs(square form) can be plotted. Calls the function hybridgell.m.

```
tic           %Timer start

%Variables for plotting:
global tnexts
global rps
global aps
global hcsps
global fhps
global rhps
global dps
global pps
global irps
global fips
global rips
global cis
global cmds

global tstart
global tfinal

N=20;            %Number of times the function hybridgell.m is called.
j=0;             %Number of runs written in the datafile.
tstart = 0;      %Start time of a simulation.
tfinal = 10;     %Stop time of a simulation.

%Profiler:
profile clear
profile on
```

```
while ( j < N)

    clear tnexts1
    clear rps1
    clear aps1
    clear hcsps1
    clear fhps1
    clear rhps1
    clear dps1
    clear pps1
    clear irps1
    clear fips1
    clear rips1
    clear fhcis1
    clear ficmds1

    %Function Call for the values of the pools and calcium compartments:
    [tnexts rps aps hcsps fhps rhps dps pps irps fips rips
    cmds cis] = hybridgell;

    %Loop for converting the saw tooth form into square form:
    for i=1:length(tnexts)-1

        tnexts1(2*i-1)   = tnexts(i);
        tnexts1(2*i)     = tnexts(i+1);
        rps1(2*i-1)      = rps(i);
        rps1(2*i)        = rps(i);
        aps1(2*i-1)      = aps(i);
        aps1(2*i)        = aps(i);
        hcsps1(2*i-1)    = hcsps(i);
        hcsps1(2*i)      = hcsps(i);
        fhps1(2*i-1)     = fhps(i);
        fhps1(2*i)       = fhps(i);
```

```
        rhps1(2*i-1)     = rhps(i);
        rhps1(2*i)       = rhps(i);
        dps1(2*i-1)      = dps(i);
        dps1(2*i)        = dps(i);
        pps1(2*i-1)      = pps(i);
        pps1(2*i)        = pps(i);
        irps1(2*i-1)     = irps(i);
        irps1(2*i)       = irps(i);
        fips1(2*i-1)     = fips(i);
        fips1(2*i)       = fips(i);
        rips1(2*i-1)     = rips(i);
        rips1(2*i)       = rips(i);
        fhcis1(2*i-1)    = fhcis(i);
        fhcis1(2*i)      = fhcis(i);
        ficmds1(2*i-1)   = ficmds(i);
        ficmds1(2*i)     = ficmds(i);

        %Plotting individual runs:

        %AP
        figure(1)
        hold on
        xlabel('Time (seconds)');
        ylabel('AP');
        plot(tnexts1,aps1,'r')

        %Similarly plot the other pools and calcium concentrations.

    end

    %Writting the values for only those simulations which are
    ...greater than some specified time. In this case 2.
    nt = length(tnexts);
```

```
    if (tnexts(nt)>2)
        %Filenames can be changed.
        dlmwrite('frphgellfast.txt',[tnexts';rps'],'-append');
        dlmwrite('faphgellfast.txt',[tnexts';aps'],'-append');
        dlmwrite('fhcsphgellfast.txt',[tnexts';hcsps'],'-append');
        dlmwrite('ffhphgellfast.txt',[tnexts';fhps'],'-append');
        dlmwrite('frhphgellfast.txt',[tnexts';rhps'],'-append');
        dlmwrite('fdphgellfast.txt',[tnexts';dps'],'-append');
        dlmwrite('fpphgellfast.txt',[tnexts';pps'],'-append');
        dlmwrite('firphgellfast.txt',[tnexts';irps'],'-append');
        dlmwrite('ffiphgellfast.txt',[tnexts';fips'],'-append');
        dlmwrite('friphgellfast.txt',[tnexts';rips'],'-append');

        j=j+1;

    end

end

time = toc        %Timer stop

profile viewer
profile off
```

File name = interpolation.m (MATLAB file)
Script to calculate mean and variance over T runs and interpolated over n = 1024 equal time points(can be changed). The data is read row wise where, for the single run first row contains all the tnext points and the second row contains the values corresponding to each tnext. Therefore, for 1000 runs there will be 2000 rows, where all the odd rows will have tnext values and even rows will have value corresponding to tnext.

```
function avg_vector = interpolation(T,n)
```

```
%Shortest runlength = minl:
%1)In every odd row find the last tnext value.
%2)Compare all the last tnext values.
%3)minl = smallest last tnext value.
minl = max(T(:,end-1));
for i = 1:2:length(T(:,1))
    idx = max(find(T(i,:)));
    if(T(i,idx) < minl)
        minl = T(i,idx);
    end
end

%Now minl contains the minimum time to which all runs have completed
...We average only till t = minl.
fprintf('shortest run length')
minl

%We divide minl into n equal time points and interpolate the runs at
...these "nodes".

Ta = [];     %Initializing the vector that will store values
             ...corresponding to each tnext till minl.

for t = 0:minl/n:minl
    for i = 1:2:length(T(:,1))
        T(i,:);
        find(T(i,:) >= t);
        id = min(find(T(i,:) >= t));
        T(i+1,:);
        if (id == 1)
            Ta((i+1)/2,t*n/minl+1) = T(i+1,id);
        else
            Ta((i+1)/2,int16(t*n/minl+1)) = T(i+1,id-1);
        end
```

```
    end
end

%Ta contains in its rows the different runs of pools interpolated...
...at n nodes.
avg_vector = Ta;

%Mean:(calculated for each column, and each column corresponds to a
...different node)
m = mean(Ta);
%Variance:(calculated for each column, and each column corresponds to
...a different node)
v = var(Ta,1);

%Plotting Mean versus time nodes:
figure(1)
hold on
plot(0:minl/n:minl,m,'r')
xlabel('time (seconds)');

%Plotting Variance versus time nodes:
figure(1)
hold on
plot(0:minl/n:minl,v,'b')
```