Deterministic and Stochastic Dynamics of Hepatic Insulin Receptor

Baihan Lin[1, 2, 3, *], **Timothy Welsh**[4], **Hyeon-Jin Kim**[4], **and Trent Nivala**[5]

[1]Institute for Protein Design, Department of Biochemistry, University of Washington, Seattle, WA 98195, USA
[2]Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA
[3]Department of Physics, University of Washington, Seattle, WA 98195, USA
[4]Department of Chemistry, University of Washington, Seattle, WA 98195, USA
[5]Department of Neurobiology, University of Washington, Seattle, WA 98195, USA
[*]doerlbh@gmail.com

Introduction

Insulin is a hormone used for metabolic processes in the body, specifically the metabolism of sugar in the form of glucose. Insulin travels extracellularly in the body and binds to the surface of target cells, causing signal transduction cascades which promote differential gene expression. This insulin signal ultimately causes changes in protein production, and thus the function of target cells is altered.

The insulin receptor (IR) is a membrane glycoprotein that is used to transduce the extracellular insulin signal. The IR does this through both conformational changes and location within the cell. In 2006, a paper was published by Hori *et al.*[1] which modeled these dynamics mathematically using a system of linear ordinary differential equations.

Here, we expand on this work, building on the basic six pool model created by Hori *et al.*[1] and exploring the dynamics when extracellular insulin concentrations were variable. In addition, we conducted parameter sensitivity analysis to hypothesize which state transitions have the strongest effects on the dynamics of the system. Finally, we explored how the system behaves when we introduce various types of stochastic dynamics to the original six pool model.

Model

The original model from Hori *et al.*[1], which is extended in this work, treated the insulin receptor as having seven distinct states constituting a six-pool model. The model is summarized below:

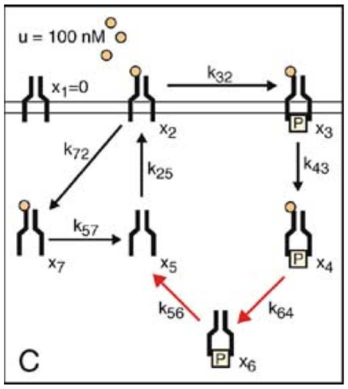


Figure 1. Schematic of six-pool model from Hori *et al.*[1]

Six Pool Model
$\frac{dx_1}{dt} = 0$
$\frac{dx_2}{dt} = k_{52}x_2 - (k_{32}+k_{72})x_2$
$\frac{dx_3}{dt} = k_{32}x_2 - k_{43}x_3$
$\frac{dx_4}{dt} = k_{43}x_3 - k_{64}x_4$
$\frac{dx_5}{dt} = k_{56}x_6 + k_{57}x_7 + -k_{25}x_5$
$\frac{dx_6}{dt} = k_{64}x_4 - k_{56}x_6$
$\frac{dx_7}{dt} = k_{72}x_2 - k_{57}x_7$

Table 1. ODEs of original six-pool model[1]

	Rate constants
k_{25}	$0.0737 min^{-1}$
k_{32}	$1.29\ min^{-1}$
k_{72}	$0.0411\ min^{-1}$
k_{43}	$0.0212\ min^{-1}$
k_{64}	$0.23\ min^{-1}$
k_{56}	$0.101\ min^{-1}$
k_{57}	$0.23\ min^{-1}$

Table 2. Reaction rate parameters of original six-pool model

The variables of the model are defined as follows: u = insulin concentration in nM, x_n = state of receptor in six-pool schematic, k_{mn} = reaction rate of transition from state x_n to x_m.

As the schematic shows, the receptor has three conditions: bound by insulin vs. not bound by insulin, phosphorylated vs nonphosphorylated, and internalized vs. on surface. The change in concentration of each state could then be described by

ODEs involving the rate constants for the reactions that move receptors either into or out of the state of interest. Those original ODEs are shown in Table 1 along with the reaction rate parameters in Table 2.

After developing the six-pool model using data fitting and parameter optimization methods, the model was compared to experimental data which monitored the concentration of insulin receptors in different states over time. Those results[1] are shown in Figure 2.

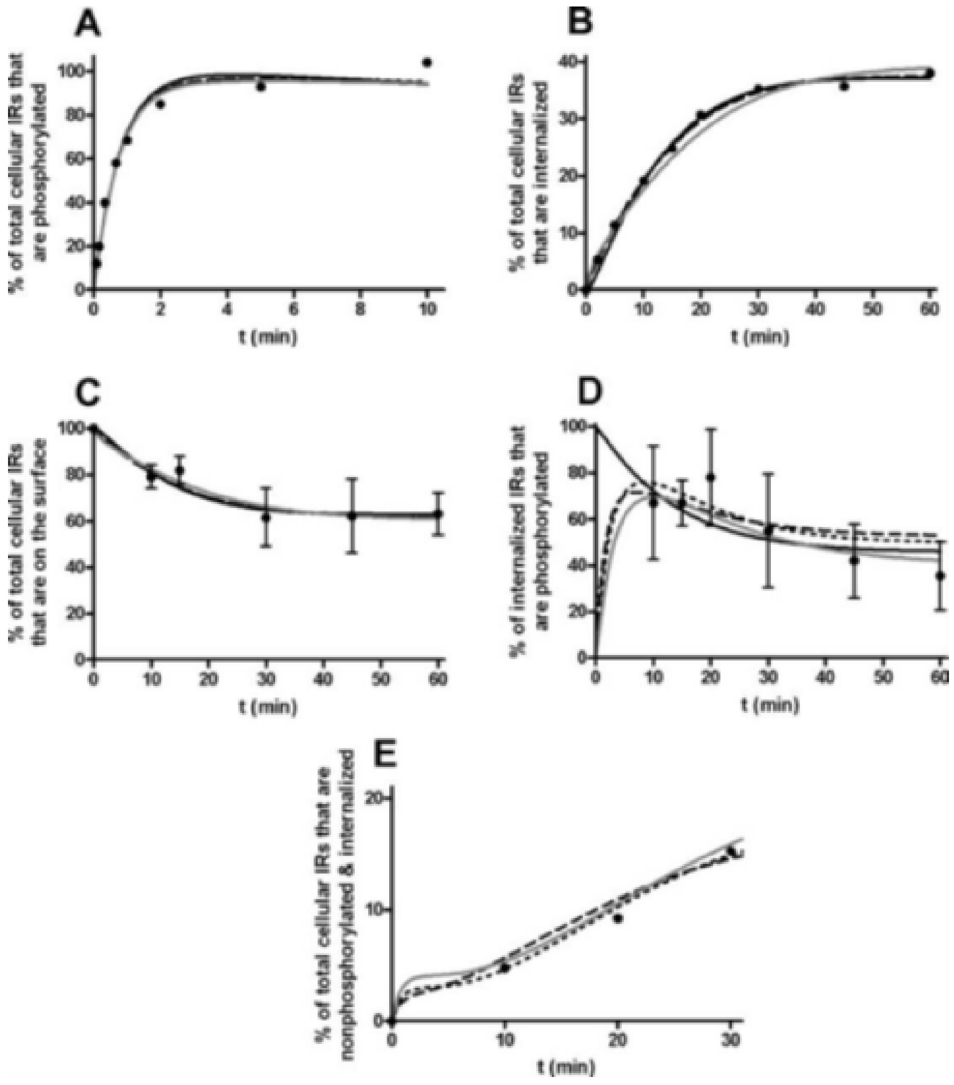


Figure 2. Comparison of model and experiment from paper. Dashed line (–) corresponds to the six-pool model of interest

By iterating the ODEs of the six-pool model using the parameters defined in the paper and using a starting condition of 100% of receptors being in the x_2 state, the results of the model from Hori *et al.*[1] could be reproduced exactly as shown in Figure 3. The ability to re-create the same results found in the original paper allowed for the model to be further analyzed and expanded on.

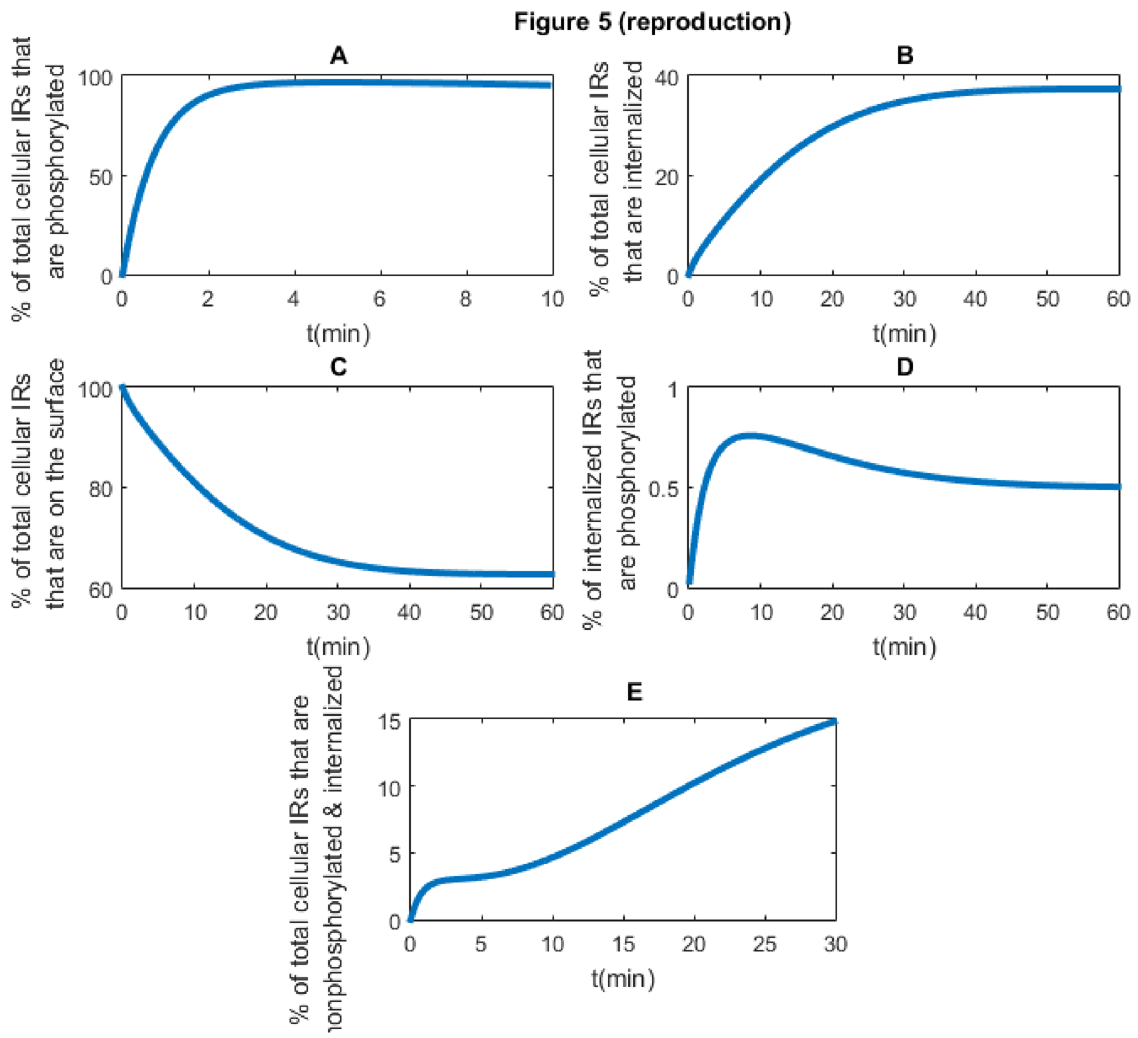


Figure 3. Reproduction of Six-pool model simulation from Figure 5. of original paper

Insulin Concentration Dependent Six-Pool Model

The original model by Hori *et al.*[1] was developed for mouse Fao heptamo cells at saturating levels of 100nM insulin. Although one of the goals of the paper was to enhance the existing insulin receptor dynamic model proposed by Backer *et al.*[3], their model fails to apply for a system that is not in saturating insulin condition. Not all biological systems are in this ideal condition and we hypothesized that changing the concentration of insulin could affect the behaviors of the receptors. Understanding how insulin concentrations affect the receptor dynamics and equilibrium state distribution is important for accurately modeling insulin and insulin receptor dynamics in a real biological system. This gives our motivation for exploring how insulin receptor dynamics behave when insulin is not necessarily saturated in the system.

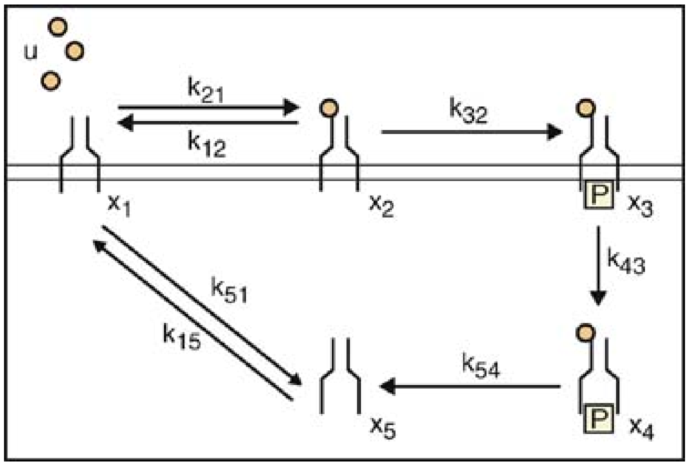


Figure 4. A general concentration dependent five pool model proposed in the original paper.[1]

Model Development

Since the receptor state is dependent on the concentration of insulin, we had to modify the ordinary different equations for the original six pool model. The paper by Hori *et al.*[1] suggested a general concentration dependent five pool model (Figure 4), however, the model did not exactly fit the six pool model and the values for rate constants (k_{12}, k_{21}, k_{15}, k_{51}) were not reported.

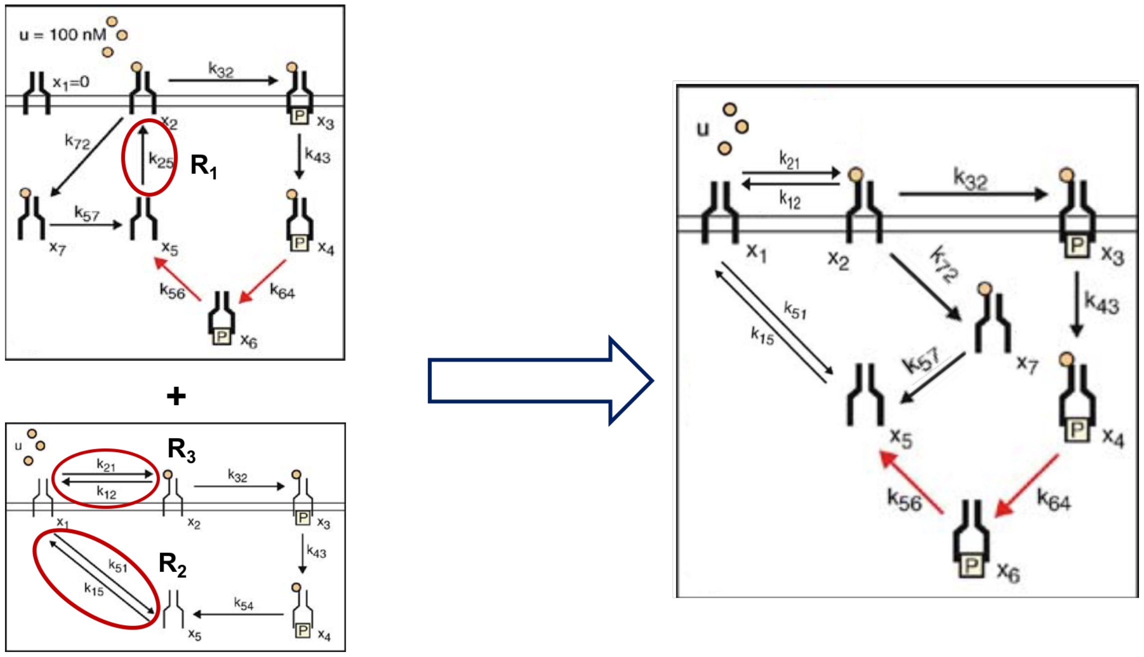
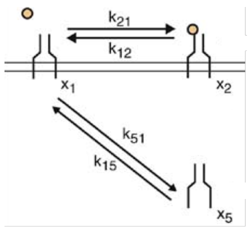


Figure 5. Combination of the original six pool model and general concentration dependent model (left) produces our insulin concentration dependent six-pool model. (right)

Our approach for developing a concentration dependent six pool model was to combine the original model and the general concentration dependent five pool model and approximate what the rate constants are based on the equilibrium constant values reported in the literature. As shown in Figure 5, recycling of receptors in x_5 state to the membrane receptor x_2 (R_1) was omitted and replaced with recycling of receptors in x_5 state to x_1 state (R_2) in our model because insulin does not bind to the membrane receptors instantly in a limiting insulin condition. Insulin binding kinetics (R_3) was also included in our model to accurately approximate the relative percentage of receptors in x_1 and x_2 state for a system with low levels of insulin.

Figure 6. Zoomed in version of concentration dependent six pool model.[3]



$$K_{eq} = \frac{k_{15}}{k_{51}} = 7.667$$

Table 3. Linear ODEs used for concentration dependent six pool model

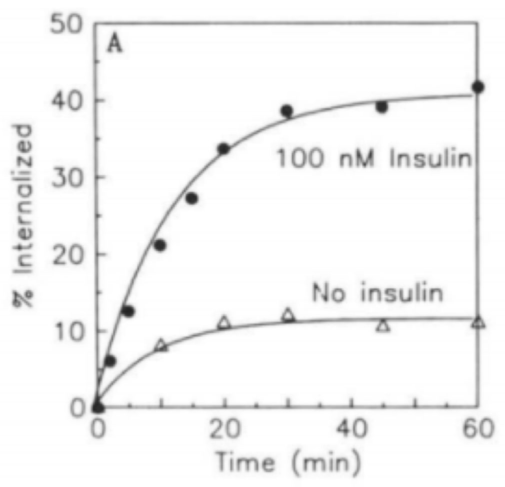
Insulin Concentration Dependent Six Pool
$\frac{dx_1}{dt} = k_{21}x_2 + k_{15}x_5 - (k_{12}u + k_{51})x_1$
$\frac{dx_2}{dt} = k_{12}ux_2 - (k_{32} + k_{72} + k_{21})x_2$
$\frac{dx_3}{dt} = k_{32}x_2 - k_{43}x_3$
$\frac{dx_4}{dt} = k_{43}x_3 - k_{64}x_4$
$\frac{dx_5}{dt} = k_{56}x_6 + k_{57}x_7 + k_{51}x_1 - k_{15}x_5$
$\frac{dx_6}{dt} = k_{64}x_4 - k_{56}x_6$
$\frac{dx_7}{dt} = k_{72}x_2 - k_{57}x_7$

Table 4. Rate constants used for concentration dependent six pool model[2]

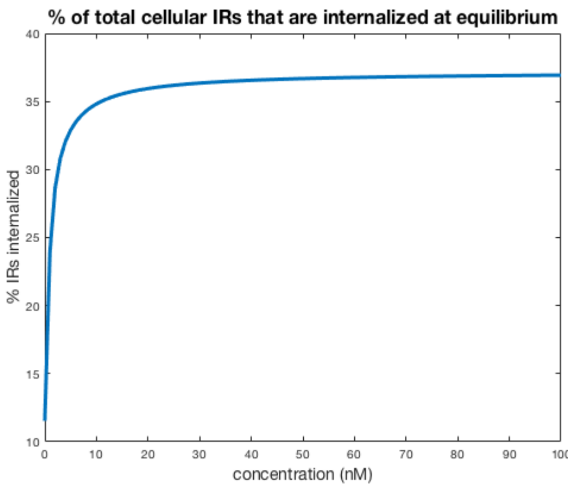
	Rate constants
k_{21}	0.0004 min^{-1}
k_{15}	0.0737 $nM^{-1}min^{-1}$
k_{12}	0.001 min^{-1}
k_{51}	0.009613 min^{-1}
k_{32}	1.29 min^{-1}
k_{72}	0.0411 min^{-1}
k_{43}	0.0212 min^{-1}
k_{64}	0.23 min^{-1}
k_{56}	0.101 min^{-1}
k_{57}	0.23 min^{-1}

The rate constants were approximated using the equilibrium constant of internalization and recycling reaction[3] of x_1 and x_5 state (Figure 6). We approximated k_{15} to be about equal to k_{25} using the fact that binding of insulin to receptors occurs significantly faster than unbinding[2]. Then, using the equilibrium constant equation, we found the value of the rate constant k_{51}. The rate constant values and differential equations used are reported in Table 3 and Table 4. In our ODEs, it was safe to assume that the insulin concentration of the system was constant because the diffusion rate of insulin through the membrane is significantly greater than the rate of internalization and recycling of insulin receptors.

Results and Discussions



(a) The percentage of internalized insulin receptors over time for a system that contains no insulin and saturated insulin.[3]



(b) The percentage of internalized insulin receptors at equilibrium as a function of time

Figure 7. A comparison of the plot reported in the literature (a) and our concentration dependent six pool model (b)

We solved the ODEs for the insulin concentration dependent six pool model with concentration of insulin ranging from 0 to 100 nM. Then we extracted the equilibrium concentration of internalized receptors for each value of insulin concentration from the model and plotted them as a function of concentration of insulin (Figure 7b). We compared our results to the plot that was reported in the literature (Figure 7a), specifically the equilibrium percentage for each system at 0 and 100nM of insulin. We weren't able to obtain exact values from Figure 7a by Backer *et al.*[3], however, the values from the endpoints of our model (7b) and the plot from the literature qualitatively matched.

Our model predicts the equilibrium percentage of internalized receptors to reach the equilibrium percentage for saturated insulin system very rapidly. The equilibrium percentage increases exponentially, until the concentration reaches 9-10 nM, then plateaus till the percentage reaches the value of saturated insulin system, 36.93%.

Parameter Sensitivity Analysis

The original six-pool model provided by Hori *et al.*[1] solved for one set of reaction rate parameters which were able to accurately match the experimental insulin receptor dynamical data. One inherent question that comes up after their analysis is, how sensitive is the concentration of each state at equilibrium to changes in the reaction rate parameters? Parameter sensitivity analysis methods from Hamby[4] were used in order to do this study. Hamby[4] reviews many various methods for parameter sensitivity analysis of biological models, thus the work is directly applicable to this six-pool model. This sensitivity analysis may provide biologically relevant insight into which transitions have the largest effect on the overall system. Understanding which reaction rate parameters induce the largest change in equilibrium states can provide possible drug target pathways for studying how varied equilibrium distributions can induce phenotypic changes in how well individuals are able to respond to insulin levels in the blood. Deeper understanding of how the reaction rate parameters affect the receptor states at equilibrium may also help suggest which transitions are disrupted in individuals who are unable to respond the insulin in a normal healthy manner. Two modes of parameter sensitivity analysis chosen from Hamby[4], differential analysis and one-at-a-time (OAT) analysis, will be applied to both the saturated insulin model and the concentration dependent insulin model. Throughout the sensitivity analysis, each state's sensitivity to each parameter will be quantified. In addition, the states will be grouped into two super-states ”phosphorylated” and ”internalized”. These two super-states are of particular biological significance and thus their sensitivities to changes in each parameter will also be shown during the OAT analysis.

Differential Parameter Sensitivity Analysis

The first necessary step in order to be able to do differential parameter sensitivity analysis on the kinetic model was to solve the equilibrium concentration of each state as a function of the reaction rate parameters. These solutions were defined as follows for a specific state x_n at equilibrium:

$$x_n(equilibrium) = F_n(\vec{k})$$

$$\text{Where the reaction rate parameter vector } \vec{k} = [k_{25}, k_{32}, k_{72}, k_{43}, k_{64}, k_{56}, k_{57}]$$

The equilibrium amounts of each state in the original six pool model were found to be the following:

$$F_1(\vec{k}) = 0 \; F_2(\vec{k}) = \frac{k_{57}z}{k_{72}}; F_3(\vec{k}) = \frac{k_{57}k_{32}z}{k_{72}k_{43}}; F_4(\vec{k}) = \frac{k_{57}k_{32}z}{k_{72}k_{64}}; F_5(\vec{k}) = \frac{k_{57}(k_{32}+k_{72})z}{k_{25}k_{64}}; F_6(\vec{k}) = \frac{k_{57}k_{32}z}{k_{56}k_{72}}; F_7(\vec{k}) = z$$
$$\text{where } z = R_{tot}/(\frac{k_{57}}{k_{72}} + \frac{k_{57}k_{32}}{k_{72}k_{43}} + \frac{k_{57}k_{32}}{k_{72}k_{64}} + \frac{k_{57}(k_{32}+k_{72})}{k_{25}k_{64}} + \frac{k_{57}k_{32}}{k_{56}k_{72}})$$

R_{tot} is defined as the total starting number of receptors in states $x_2 : x_7$ which is set as 100 in the context of the six pool model.

The sensitivity coefficient of state x_i with respect to parameter k_{mn} is defined as follows:

$$\phi^i_{mn} = \frac{\partial F_i(\vec{k})}{\partial k_{mn}} \frac{k_{mn}}{F_i(\vec{k})}$$

Sensitivity Coefficients of all parameters corresponding to each state						
Parameter	States					
	x_2	x_3	x_4	x_5	x_6	x_7
k_{25}	0.1834	0.1834	0.1834	-0.8166	0.1834	0.1834
k_{32}	-0.9824	0.0176	0.0176	-0.0132	0.0176	-0.9824
k_{72}	-0.0075	-0.0075	-0.0075	0.0234	-0.0075	0.9925
k_{43}	0.6179	-0.3821	0.6179	0.6179	0.6179	0.6179
k_{64}	0.0570	0.0570	-0.9430	0.0570	0.0570	0.0570
k_{56}	0.1297	0.1297	0.1297	0.1297	-0.8703	0.1297
k_{57}	0.0018	0.0018	0.00180	0.0018	0.0018	-0.9982

Table 5. This table summarizes the sensitivity coefficients of each state with respect to each parameter

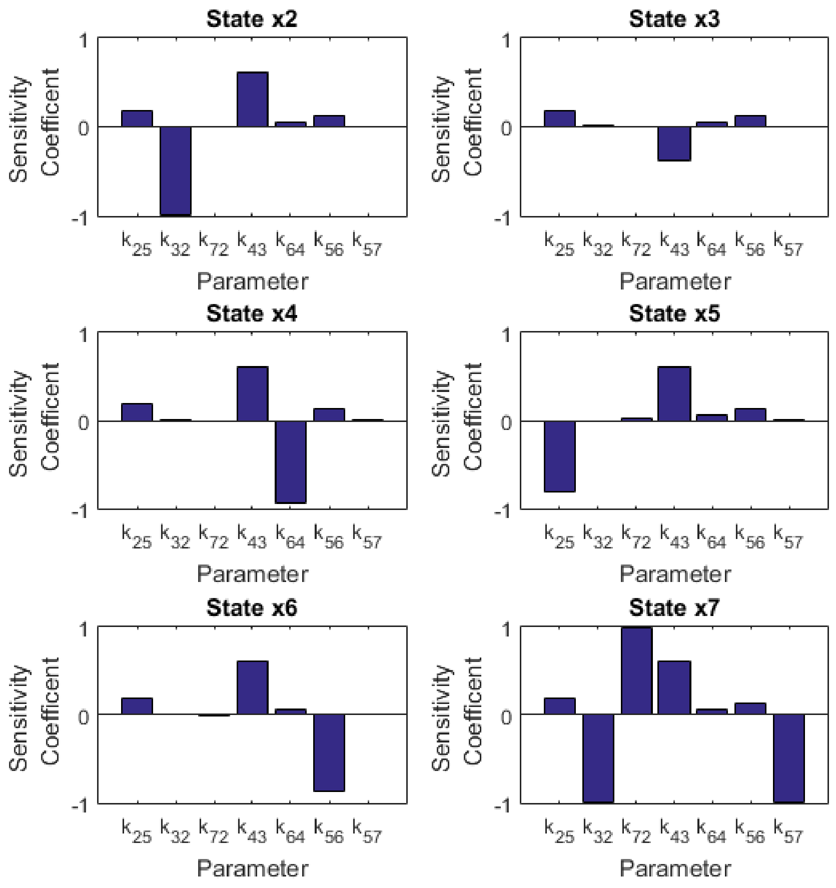


Figure 8. This figure provides a graphical representation of the sensitivity coefficients from Table 5.

Overall, the differential parameter sensitivity analysis showed that each state had varying levels of sensitivity to change in each parameter. One trend that was noticed was that most parameters affected one state unequally to the others. This essentially shows us that each parameter will cause a unique sensitivity for one state and thus matching those parameters and states may be helpful for inducing targeted pertubations in the system.

One-at-a-time Sensitivity Measures

The second method utilized in order to analyze the sensitivity of the equilibrium state distribution with respect to change in each of the reaction rate parameters was one-at-a-time (OAT) perturbations. This method was defined as follows:

$$\text{Again, } X_n(equilibrium) = F_n(\vec{k})$$
$$\text{Where the parameter vector } \vec{k} = [k_{25}, k_{32}, k_{72}, k_{43}, k_{64}, k_{56}, k_{57}]$$

One of the parameters was then varied by a perturbation factor Δ while the others were held constant. For example, if k_{25} was being varied then the perturbed parameter vector $\vec{k}_p = [k_{25} + \Delta k_{25}, k_{32}, k_{72}, k_{43}, k_{64}, k_{56}, k_{57}]$. The deviation from equilibrium was then measured as a percentage change:

$$Deviation = \frac{F_n(\vec{k}_p) - F_n(\vec{k})}{F_n(\vec{k})}$$

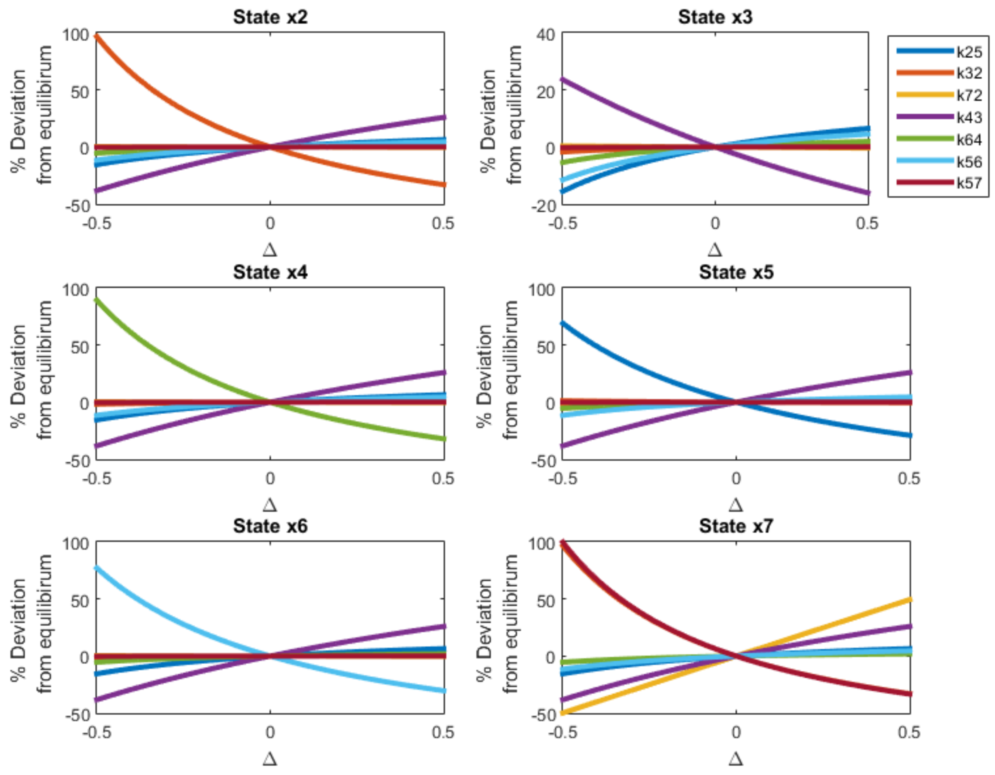


Figure 9. Deviation from equilibrium of each state as each parameter is varied

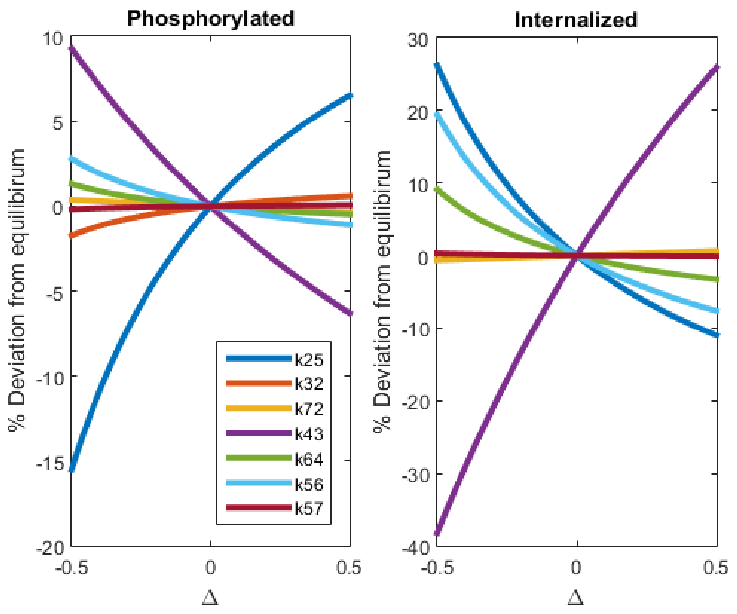


Figure 10. Deviation from equilibrium of total phosphorylated and internalized states as each parameter is varied

Results and discussion from Parameter Sensitivity Analysis

Overall, the parameter sensitivity analysis showed that each state concentration at equilibrium has varying levels of sensitivity to changes in the reaction rate parameters. The two methods of sensitivity analysis utilized agreed in the directionality and relative amount of sensitivity that each state had to changes in each parameter. There are no conclusions in terms of which method of analysis was better, but computationally the differential analysis is much less expensive compared to the OAT analysis because the OAT analysis requires running a simulation of the differential equations for each new parameter set. Comparatively, the differential analysis only requires taking the derivative of each state solution once with respect to each parameter.

Doing this analysis allowed us to make some important conclusions about which parameters in the original six-pool model were biologically important. We can see that there are a few parameters such as k_{25} and k_{43} for which the phosphorylated and internalized super-states were highly sensitive. This means that the transitions between states x_5 and x_2 as well as x_3 and x_4 are the most important with regard to the overall distribution of receptors between in the two super-states. This makes intuitive sense because states x_2 and x_4 have the highest turn over rate and thus the largest effect on the distribution and flux through the model at equilibrium. These two transitions therefore may be interesting transitions to target with drugs in order to study how different insulin receptor state distributions cause changes in glucose metabolism. These sensitivities also give us insight into which reactions may be affected in diabetic individuals who have problems with insulin receptor response.

Stochastic Description of Insulin Receptor Dynamics

In the paper by Hori et al.[1], they simulated insulin receptor dynamics by solving deterministic linear differential equations. Although their model was sufficient enough to simulate insulin receptor dynamics, we wanted to illustrate how making the transition probabilistic would affect the relative amount of receptors in each state. To do so, we implemented stochasticity in two different ways: a Bayesian model and a Monte Carlo simulation.

Bayesian Model for Stochastic Dynamics

The first approach was based on the Bayesian model where the probability of one receptor going to the next state at some time Δt is the product of the rate constant of a reaction and Δt. The model is specified as below:

In one directional reaction:

$$\mathrm{A} \xrightarrow{\mathrm{k\dot{}BA}} \mathrm{B}$$

$$P(X(t+\Delta t)=B|X=A)=k_{BA}\Delta t \tag{1}$$

$$P(X(t+\Delta t)=A|X=A)=1-k_{BA}\Delta t \tag{2}$$

In two directional reactions:

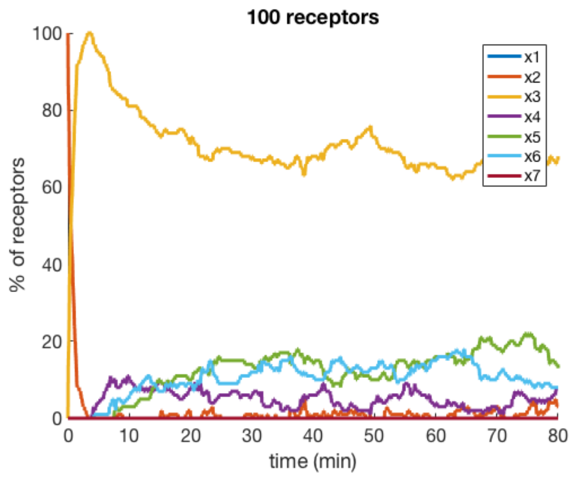
$$\mathrm{C} \xleftarrow{\mathrm{k\dot{}CA}} \mathrm{A} \xrightarrow{\mathrm{k\dot{}BA}} \mathrm{B}$$

$$P(X(t+\Delta t)=B|X=A)=\frac{k_{BA}}{k_{BA}+k_{CA}}k_{BA}\Delta t \tag{3}$$

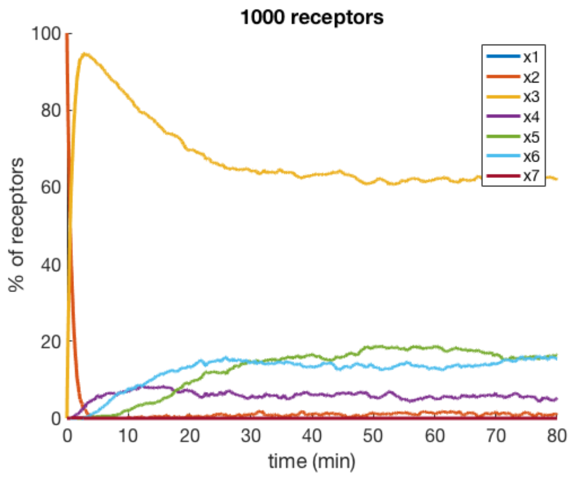
$$P(X(t+\Delta t)=C|X=A)=\frac{k_{CA}}{k_{BA}+k_{CA}}k_{CA}\Delta t \tag{4}$$

$$P(X(t+\Delta t)=A|X=A)=1-(\frac{k_{BA}}{k_{BA}+k_{CA}}k_{BA}\Delta t+\frac{k_{CA}}{k_{BA}+k_{CA}}k_{CA}\Delta t) \tag{5}$$

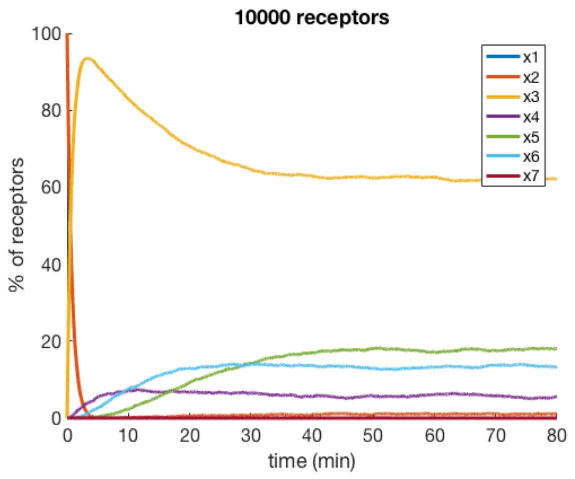
For two directional reactions, the weighted rate constant is used to calculate the probability, which is the product of rate constant of a given reaction and the ratio between the given rate constant over the sum of rate constants for all possible reactions.



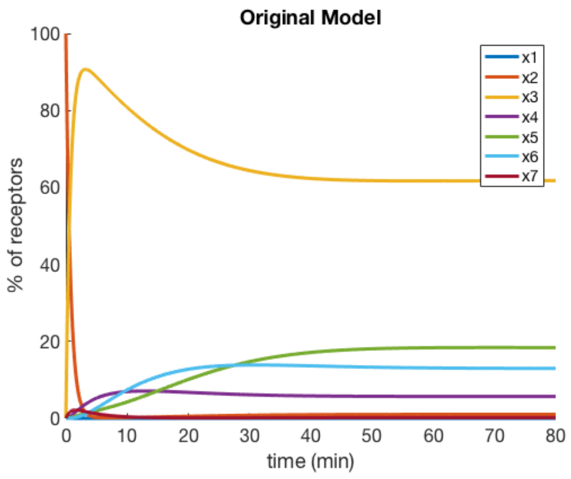
(a) 100 receptors, Δt = 0.05 min



(b) 1000 receptors, Δt = 0.05 min



(c) 10000 receptors, Δt = 0.05 min



(d) Linear ODE model

Figure 11. Simulations of receptor dynamics using Bayesian model (a)-(c) and deterministic linear ODE model (d)

Based on this stochastic model, at every Δt, one receptor could go to another state or stay in the same state based on the transition probability. As shown in Figure 11, increased number of receptors decreases the fluctuations of distribution of receptors in each state at equilibrium. As a result, the model 11c closely resembles the deterministic linear ODE model.

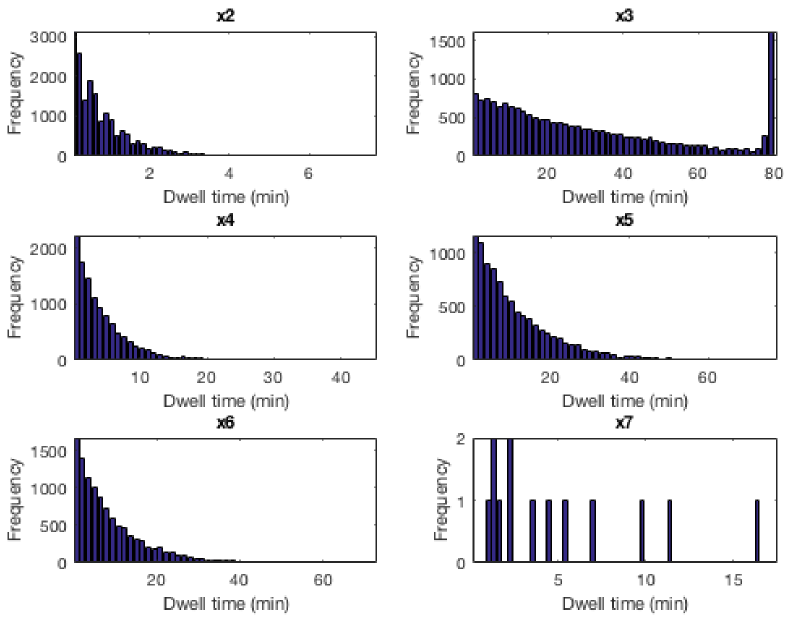


Figure 12. Dwell time of 10000 receptors for each state

Although this method was computationally inefficient, we were able to compute dwell times of receptors in each state. As shown in Figure 12, the dwell time distribution generated from Bayesian model suggested that dwell times for x_2,x_4,x_5, and x_6 were exponentially distributed. We can also see a significant amount of receptors in x_3 staying in x_3 state for most of the simulation time. This can be explained by the fact that all the receptors were in x_2 state initially and the probability of going to x_4 state from x_2 state was relatively low. In addition, a highly variable behavior in dwell times of x_7 was observed for Bayesian model due to significantly low number of receptors in x_7.

Poisson Waiting Time Model for Stochastic Dynamics

Inspired by Gillespie[5] and Dykeman et al.[6], we took second approach with a logarithmic waiting time assumption for this system. If we consider all the reactions involved in this system as independent reactions, then we can assume a Poisson distribution of the waiting time until next reaction occurs.

```
rates = [   k32 * y(2);
            k43 * y(3);
            k64 * y(4);
            k56 * y(6);
            k25 * y(5);
            k72 * y(2);
            k57 * y(7); ];
```

Figure 13. The reaction rates of each possible reaction excluding state x_1

For each reaction type, the mean time till the next reaction occurs again is the inverse of the reaction rate. The rate at which anything will happen is simply the sum of all the reaction rates (Figure 13), which will be denoted as λ. We can also write the mean time until any reaction occurs as

$$\tau_m = \frac{1}{\lambda}$$

Then, under the assumption of a Poisson process, it's fairly straightforward to pick the next most likely reaction and the amount of time until the reaction occurs. For time, τ, we have

$$\tau = \frac{1}{\lambda}\log(\frac{1}{U_1})$$

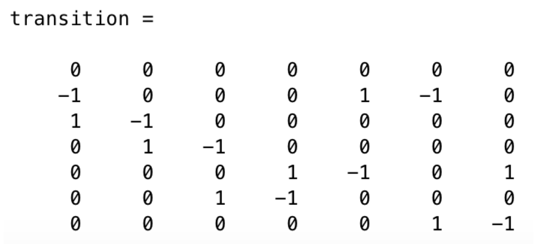
where U_1 is drawn uniformly from the unit interval. Likewise, if we denote the individual reaction rates as a_i we can select one reaction based on the relative reaction rates by using a random variable,

$$r = \lambda * U_2$$

where U_2 is also drawn uniformly from the unit interval. This gives us a random variable in the range $(0, \lambda)$ which we can use to select the next reaction j by finding j such that

$$\sum_{0}^{j-1} \leq r \leq \sum_{0}^{j}$$

We then update the current time step with the selected reaction from the transition matrix (Figure 14). This process was repeated for number of desired time steps for our simulation. The update matrix that was output stored the number of receptors in each state for every time step.

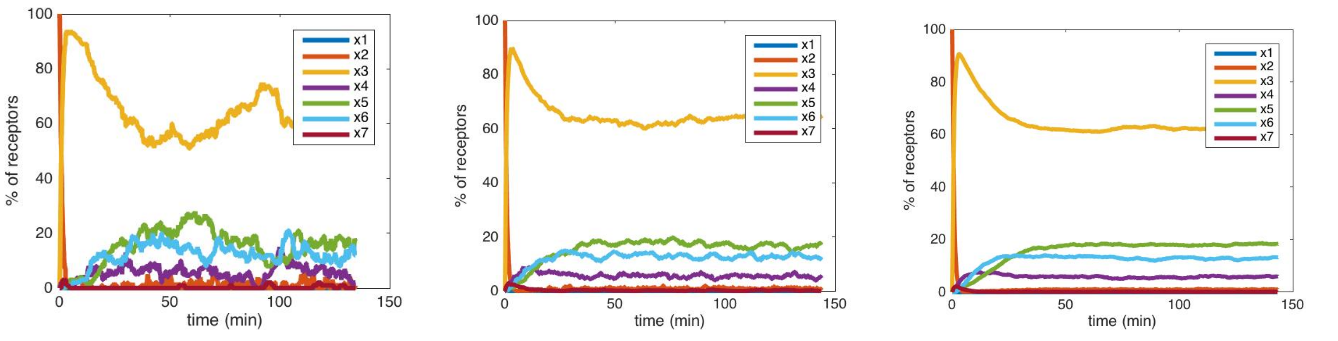


```
transition =

     0     0     0     0     0     0     0
    -1     0     0     0     1    -1     0
     1    -1     0     0     0     0     0
     0     1    -1     0     0     0     0
     0     0     0     1    -1     0     1
     0     0     1    -1     0     0     0
     0     0     0     0     0     1    -1
```

Figure 14. The transition matrix for Monte Carlo stochastic system. Each column represents a possible transition the rate of which is described in the corresponding entry in the previously defined rate vector and each row represents the state of a receptor x_i where i is the row index

Monte Carlo Simulation and Waiting Time Analysis



(a) 100 receptors 1000 simulation steps **(b)** 1000 receptors 10000 simulation steps **(c)** 10000 receptors 100000 simulation steps

Figure 15. Monte Carlo simulation of the stochastic insulin receptor dynamics

From the stochastic simulation shown in Figure 15, we can observe that the long-term behavior of the insulin receptor activity is relatively stable, similar to what we observed in the Bayesian model and deterministic model. As shown in Figure 16, the Monte Carlo simulation would eventually reside near the equilibrium region as expected.

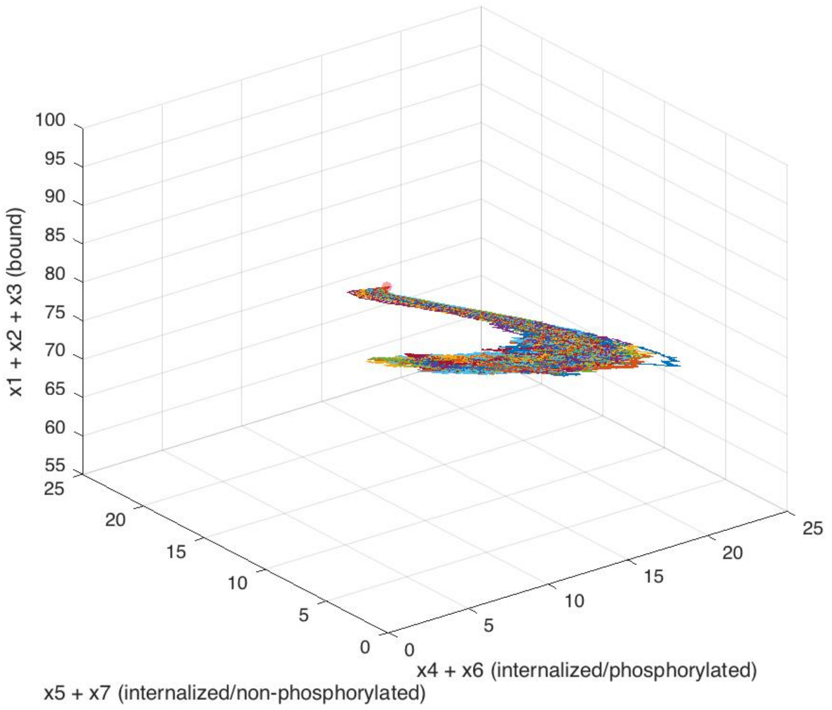
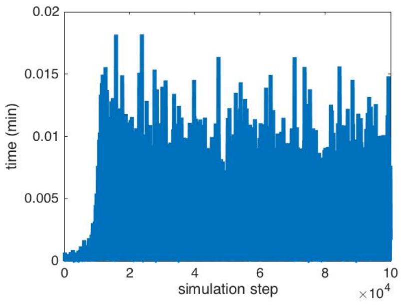
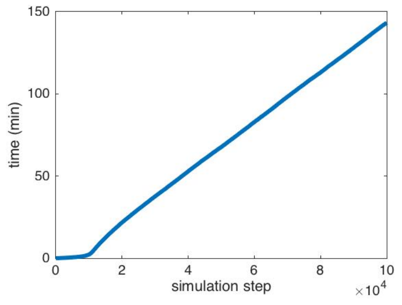


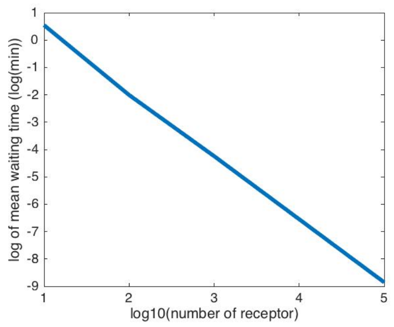
Figure 16. Monte Carlo simulation of internalized/phosphorylated, internalized/non-phosphorylated, and bound insulin receptors in 3 dimensional space with 1000 receptors and 5000 simulation steps. The red spot is the starting point where all receptors were initially in x_2 state. 100 independent simulations were performed displayed in this time space.



(a) The waiting time in each 100000 simulation step for 10000 receptors



(b) The cumulative time in 100000 simulation steps for 10000 receptors



(c) The relationship of number of receptors and mean waiting time in each step

Figure 17. Stochastic waiting time dynamics for Monte Carlo simulation

As shown in Figure 17a and 17b, comparing to the deterministic model which assumes the transition rates are continuous, the stochastic simulation shows a jump into a linear time rate after the first few steps. The short waiting time at the first few simulation steps can be attributed to the initial state (all receptors are in x_2) and the fast turnover rate of x_2 to x_3.

Based on the interesting finding from Figure 15, the simulation steps required to reach the same time spent seems to be positively correlated with total number of receptors. This implies that the waiting time in each step can be inversely correlated to the total number of receptors. We briefly explored this behavior in Figure 17c, and found that the average waiting time for each step of reaction follows an exponential relationship with the number of receptors. We proposed a possible explanation that the mean of the first time passage distribution in this system would decrease with an increased number of receptor. This first hitting time model is not an emphasis and is therefore not elaborated in this paper.

Discussion of the Stochastic Models

Although the deterministic model works well in capturing the behavior towards equilibrium, stochastic simulation more accurately describes the time-dependent dynamics of insulin receptor activity. While the deterministic model updates each step in by integrating the system of differential equations, it doesn't take into account the stochastic nature of biological systems, where the occurrence of each reaction are not continuous, but rather, probabilistic. Statistically speaking, the average behavior resembles the deterministic model, but realistically, stochastic model capture more delicacy regarding time scale.

Conclusion

Each one of our models provides additional insight into the behavior of insulin receptors. For our concentration dependent six pool model, we observe that the model matches data from the literature when insulin concentrations are near zero and fully saturated. However, we could not fully validate this model because we lack data regarding how the receptor states vary as a function of insulin concentration. In our parameter sensitivity analysis, we were able to hypothesize which state transitions have the greatest effect on the dynamics of the system. Interestingly, the parameters vary widely in their impact on the state distribution of the system at equilibrium. Specifically, we found that the parameters k_{25} and k_{43} have the largest effect on the phosphorylated and internalized states. As such, these state transitions would be excellent targets for a drug or other method which tries to affect the dynamics of the system. In addition, modeling the system stochastically yielded data that was very similar to our deterministic ODE model. This was more true in our Bayesian model when we used a theoretically high number of receptors. We also found that when using a Monte Carlo simulation that in order to resolve the system to equilibrium, the number of time steps must be proportional to the number of receptors.

Modeling insulin and insulin receptors could be very important from a medical and public health standpoint. Being able to construct models that are biologically relevant could assist in theoretically testing drugs and other methods whose aim is to affect the insulin signal transduction pathway. Insulin itself is an incredibly important hormone in biological systems with a wide range of effects. Dysfunction in the signaling system mediated by insulin are highly consequential in their impact on the function of a biological system. In humans, one such dysfunction is Diabetes mellitus, a disease in which the body cannot appropriately process sugars due to deficiencies in the metabolic systems regulated by insulin. Diabetes affects 415 million people worldwide. Therefore, it's possible that modeling insulin and insulin receptors could help in creating a treatment for Diabetes.

References

1. Hori Sharon S., Kurland Irwin J., Distefano Joseph J. *Role of Endosomal Trafficking Dynamics on the Regulation of Hepatic Insulin Receptor Activity: Models for Fao Cells.*. Annals of Biomedical Engineering 34.5 (2006): 879-92.

2. Wanant Sumanas, Quon Michael J. *Insulin Receptor Binding Kinetics: Modeling and Simulation Studies.*. Journal of Theoretical Biology 205.3 (2000): 355-64.

3. Backer John. *Tyrosine Phosphorylation of the Insulin Receptor during Insulin-stimulated Internalization in Rat Hepatoma Cells.*. The Journal of Biological Chemistry. U.S. National Library of Medicine, 1988.

4. Hamby D. M. *A Review of Techniques for Parameter Sensitivity Analysis of Environmental Models*. Environmental Monitoring and Assessment, 32.2:135–54, 1994.

5. Gillespie Daniel T. *A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions*. Journal of Computational Physics, 22.4 :403-34, 1976.

6. Dykeman Eric C. *An Implementation of the Gillespie Algorithm for RNA Kinetics with Logarithmic Time Update.*. Nucleic Acids Research 43.12 (2015): 5708-715.

Acknowledgments

Thank Prof. Shea-Brown for the great lectures and guidance!

Appendix

Supporting MATLAB codes are attached on the following pages:

Original Six Pool Model Simulation Function

```
function dy = sixpool(t,y,p)

   dy = zeros(7,1);

   dy(1) = 0;
   dy(2) = p(1)*y(5) - (p(2) + p(3))*y(2);
   dy(3) = p(2)*y(2) - p(4)*y(3);
   dy(4) = p(4)*y(3) - p(5)*y(4);
   dy(5) = p(6)*y(6) + p(7)*y(7) - p(1)*y(5);
   dy(6) = p(5)*y(4) - p(6)*y(6);
   dy(7) = p(3)*y(2) - p(7)*y(7);
```

Original Six Pool Model Simulation

```
clear all; close all

for numreps=1:1

p = [0.0737 1.29 0.0411 0.0212 0.23 0.101 0.23];
x0 = [0, 100, 0, 0, 0, 0, 0];

Tspan=0:0.1:80;
options = odeset('NonNegative',2:7); %make solutions nonnegative

[T,Y] = ode45(@(t,y) sixpool(t,y,p),Tspan,x0,options);

figure
set(gca, 'FontSize', 16); hold on;
plot(Tspan,Y,'LineWidth',3); hold on;
legend('x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7')
title('Original Model')
xlabel('time (min)');
ylabel('% of receptors');

end
```

Insulin Concentration Dependent Six Pool Model Function

```
function dy = usixpool(t,y,p)

   dy = zeros(7,1);

   dy(1) = p(1)*y(2) + p(2)*y(5) - (p(3)*p(11) + p(4))*y(1);
   dy(2) = p(3)*p(11)*y(1) - (p(5) + p(6)+ p(1))*y(2);
   dy(3) = p(5)*y(2) - p(7)*y(3);
   dy(4) = p(7)*y(3) - p(8)*y(4);
   dy(5) = p(9)*y(6) + p(10)*y(7) + p(4)*y(1)- p(2)*y(5);
   dy(6) = p(8)*y(4) - p(9)*y(6);
   dy(7) = p(6)*y(2) - p(10)*y(7);

   %p(1)=k21=0.0004    p(6)=k72=0.0411      p(11)=u
   %p(2)=k15=0.0737    p(7)=k43=0.0212
   %p(3)=k12=0.001     p(8)=k64=0.23
   %p(4)=k51=0.009613  p(9)=k56=0.101
   %p(5)=k32=1.29      p(10)=k57=0.23
```

Insulin Concentration Dependent Six Pool Model

```
clear all; close all;

internal=zeros(1,100);
```

```
for conc=0:100

p = [0.00074 0.0737 0.015 0.009613 1.29 0.0411 0.0212 0.23 0.101 0.23 conc];
x0 = [100, 0, 0, 0, 0, 0, 0];

Tspan=0:0.25:100;
options = odeset('NonNegative',1:7); %make solutions nonnegative

[T,Y] = ode45(@(t,y) usixpool(t,y,p),Tspan,x0,options);

internal(conc+1)=Y(401,4)+Y(401,5)+Y(401,6)+Y(401,7);
end

figure
plot(0:100, internal, 'LineWidth', 3);
title('% of total cellular IRs that are internalized at equilibrium', 'FontSize', 16);
xlabel('concentration (nM)', 'FontSize', 14);
ylabel('% IRs internalized', 'FontSize', 14);
```

Differential Parameter Sensitivity Analysis

```
%solve where all differential equations are zero
clear all; close all;
syms p1 p2 p3 p4 p5 p6 p7 y1 y2 y3 y4 y5 y6 y7;

dy1 = 0;
dy2 = p1*y5 - (p2 + p3)*y2;
dy3 = p2*y2 - p4*y3;
dy4 = p4*y3 - p5*y4;
dy5 = p6*y6 + p7*y7 - p1*y5;
dy6 = p5*y4 - p6*y6;
dy7 = p3*y2 - p7*y7;
eqns=[p1*y5 - (p2 + p3)*y2==0,p2*y2 - p4*y3==0,p4*y3 - p5*y4==0,p6*y6 + p7*y7 - p1*y5==0,p5*y4 - p6*y6==0,p3*y2 - p7
vars=[y1,y2,y3,y4,y5,y6,y7];
[a b c d e f g params conditions]=solve(eqns,vars,'ReturnConditions',true);
sys=[a b c d e f g];


%%
%solve z parameters and such....
%    p(1)=k25
%    p(2)=k32
%    p(3)=k72
%    p(4)=k43
%    p(5)=k64
%    p(6)=k56
%    p(7)=k57

syms z1 tot p1 p2 p3 p4 p5 p6 p7
zsol=solve(sum(sys)==tot,z1)

z=0;
z1=zsol;
sysnew=subs(sys);
tot=100;
p = [0.0737 1.0 0.0411 0.0212 0.23 0.101 0.23];
p1=p(1);p2=p(2);p3=p(3);p4=p(4);p5=p(5);p6=p(6);p7=p(7);


Yeq=double(subs(sysnew))


%%
clear p states_sens system_sens phos_sens internalized_sens d
p = [0.0737 1.29 0.0411 0.0212 0.23 0.101 0.23 ];
```

```
derivatives=zeros(length(p),7);
format long
states_sens=zeros(length(p),7);
system_sens=zeros(length(p),1);
phos_sens=zeros(length(p),1);
internalized_sens=zeros(length(p),1);
for i=1:length(p)
    syms p1 p2 p3 p4 p5 p6 p7 z1;
    parameters=[p1 p2 p3 p4 p5 p6 p7];
    d=diff((sysnew),parameters(i));
    p = [0.0737 1.29 0.0411 0.0212 0.23 0.101 0.23 0.181472575502645];
    p1=p(1);p2=p(2);p3=p(3);p4=p(4);p5=p(5);p6=p(6);p7=p(7);
    derivatives(i,:)=double(subs(d));
    states_sens(i,:)=derivatives(i,:)*(p(i))./double(subs(sysnew));
    dsystem=diff(sum(sysnew),parameters(i));
    system_sens(i)=double(subs(dsystem))*(p(i)./sum(Yeq));
    %phosphorylated=y3+y4+y6
    phos_der=diff(sysnew(3)+sysnew(4)+sysnew(6),parameters(i));
    phos_sens(i)=double(subs(phos_der))*(p(i)/(Yeq(3)+Yeq(4)+Yeq(6)));
    %internalized=4 5 6 7
    internalized_der=diff(sysnew(5)+sysnew(7)+sysnew(4)+sysnew(6),parameters(i));
    internalized_sens(i)=double(subs(internalized_der))*(p(i)/(Yeq(5)+Yeq(4)+Yeq(6)+Yeq(7)));

end

for x=2:7
    figure(6)
    subplot(3,2,x-1)
    bar(states_sens(:,x));
    name=sprintf('State x%i', x);
    title(name)
    set(gca,'XTickLabel',{'k_{25}','k_{32}','k_{72}','k_{43}','k_{64}','k_{56}','k_{57}'})

    ylabel(sprintf('Sensitivity \n Coefficent'))
    xlabel('Parameter')
    ylim([-1 1])
end
```

OAT sensitivity analysis

```
clear all; close all

p1=0.0737; %k25
p2=1.29; %k32
p3=0.0411; %k72
p4=0.0212; %k43
p5=0.23; %k64
p6=0.101; %k56
p7=0.23; %k57

var=zeros(201,7);
%parameter = 0:200:2p(i)

p = [p1 p2 p3 p4 p5 p6 p7];
x0 = [0, 100, 0, 0, 0, 0, 0];
list=[];

Tspan=0:0.25:200;
options = odeset('NonNegative',2:7); %make solutions nonnegative

[T,Y] = ode45(@(t,y) sixpool(t,y,p),Tspan,x0,options);

eq=Y(801,:);

for i=1:7

    vary=p;
    count=0;

    for k=0:p(i)/100:2*p(i)
        count=count+1;
        vary(i)=k;
        x0 = [0, 100, 0, 0, 0, 0, 0];

        Tspan=0:0.25:200;
        options = odeset('NonNegative',2:7); %make solutions nonnegative

        [T,Y] = ode45(@(t,y) sixpool(t,y,vary),Tspan,x0,options);
        list=[0];
        for x=2:7
            list=[list (Y(801,x)-eq(x))];
        end
        var(count,i,1:7) = list;
    end

end

%%
% phosphorylated = y(3) + y(4) + y(6)
figure(1)
subplot(1,2,1)
xax=[-1:1/100:1]
plot(xax,100*(var(:,:,3) + var(:,:,4) + var(:,:,6))/(eq(3)+eq(4)+eq(6)),'LineWidth',3);hold on;
legend('k25', 'k32', 'k72', 'k43', 'k64', 'k56', 'k57')
title('Phosphorylated')
xlabel('\Delta');
ylabel('% Deviation from equilibirum');
 xlim([-0.5 .5])

% internalized = y(4) + y(5) + y(6) + y(7)
subplot(1,2,2)
plot(xax ,100*(var(:,:,4) + var(:,:,5) + var(:,:,6) + var(:,:,7))/(eq(4)+eq(5)+eq(6)+eq(7)),'LineWidth',3);hold on;
title('Internalized')
xlabel('\Delta');
ylabel('% Deviation from equilibirum');
 xlim([-0.5 .5])

% deviation from equilibrium of each state
for x=2:7
    figure(2)
    subplot(3,2,x-1)
    plot(-1:1/100:1,var(:,:,x)*100/eq(x),'LineWidth',3);
    ylabel(sprintf('%% Deviation \n from equilibirum'));
    xlabel('\Delta')
    name=sprintf('State x%i', x);
    if(x==2)
        legend('k25', 'k32', 'k72', 'k43', 'k64', 'k56', 'k57')
    end
    title(name)
    xlim([-0.5 .5])
end
```

Stochastic Simulation - Bayesian Model

```
close all; clear all;

p1=0.0737; %k25
p2=1.29; %k32
p3=0.0411; %k72
p4=0.0212; %k43
p5=0.23; %k64
p6=0.101; %k56
p7=0.23; %k57
```

```
delta=0.05; % delta t
Tmax=80;
nums=1000; % number of receptors

p = [p1 p2 p3 p4 p5 p6 p7];
x0 = [0, 1, 0, 0, 0, 0, 0];
tot = zeros(Tmax/delta+1,7);
dwell=zeros(Tmax/delta+1,100);
y=zeros(Tmax/delta+1,7);

for receptors=1:nums
    y = bruteS(x0, p);
    tot = tot + y;

    for t=1:(Tmax/delta+1)
        k=find(y(t,:) == 1);
        dwell(t,receptors)=k;
    end
end

%% Simulate the model
figure(1)
set(gca,'FontSize',16); hold on;
plot(0:delta:Tmax, tot/10,'LineWidth',3);
title('1000 receptors')
xlabel('time (min)');
ylabel('% of receptors');
legend('x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7');

%% Dwell times

dwelltimes=zeros(7,1);
list2=[];
list3=[];
list4=[];
list5=[];
list6=[];
list7=[];

for receptors=1:nums
    value=dwell(1,receptors);
    time=1;
    for t=1:(Tmax/delta)
        next = dwell(t+1,receptors);
        if next ~= value || t==(Tmax/delta)
            realdwell = time*delta;
            if value == 2
                list2 = [list2 realdwell];
            elseif value == 3
                list3 = [list3 realdwell];
            elseif value == 4
                list4 = [list4 realdwell];
            elseif value == 5
                list5 = [list5 realdwell];
            elseif value == 6
                list6 = [list6 realdwell];
            else
                list7 = [list7 realdwell];
            time=1;
            end
        else
            time = time + 1;
        end
        value = next;
    end
end

    dwelltimes(2,1:length(list2)) = list2;
    dwelltimes(3,1:length(list3)) = list3;
    dwelltimes(4,1:length(list4)) = list4;
    dwelltimes(5,1:length(list5)) = list5;
    dwelltimes(6,1:length(list6)) = list6;
    dwelltimes(7,1:length(list7)) = list7;

for index=2:7
    xmax=max(dwelltimes(index,:));
    pIndex = dwelltimes(index,:) > 0;
    [nlist,centerlist]=hist(dwelltimes(index,pIndex),50);
    ymax=max(nlist);

    figure(2)
    subplot(3,2,index-1)
    bar(centerlist,nlist)
    axis([0.1 xmax+1 0 ymax])
    name=sprintf('x%i', index);
    title(name)
    xlabel('Dwell time (min)')
    ylabel('Frequency')
end
```

Stochastic Update - Bayesian Model

```
function update = bruteS(y0, p)

Tmax=80;
delta=0.05;
update = zeros(Tmax/delta+1,7);
update(1,:)=y0;

%k25 = p1
%k32 = p2
%k72 = p3
%k43 = p4
%k64 = p5
%k56 = p6
%k57 = p7


x2= p(2) + p(3);
x3= p(4);
x4= p(5);
x5= p(1);
x6= p(6);
x7= p(7);

x=[0 x2 x3 x4 x5 x6 x7];
for t=1:Tmax/delta
    s=rand();
    if update(t,2)==1
        is3 = delta * p(2)*p(2) / x2;
        is7 = delta * p(3)*p(3) / x2;
        if s <= is3
            update(t+1,3)=1;
        elseif s <= is3 + is7
            update(t+1,7)=1;
        else
            update(t+1,2)=1;
        end
    elseif update(t,3)==1
        is4 = delta * x3;
        if s <= is4
            update(t+1,4)=1;
        else
            update(t+1,3)=1;
        end
    elseif update(t,4)==1
        is6 = delta * x4;
        if s <= is6
            update(t+1,6)=1;
        else
            update(t+1,4)=1;
        end
    elseif update(t,6)==1
        is5 = delta * x6;
        if s <= is5
            update(t+1,5)=1;
        else
            update(t+1,6)=1;
        end
    elseif update(t,5)==1
        is2 = delta * x5;
        if s <= is2
            update(t+1,2)=1;
        else
            update(t+1,5)=1;
        end
    else
        is5 = delta * x7;
        if s <= is5
            update(t+1,5)=1;
        else
            update(t+1,7)=1;
        end
    end
end
```

Monte Carlo Stochastic Simulation

```
clear all; close all

p1=0.0737;  %k25
p2=1.29;    %k32
p3=0.0411;  %k72
p4=0.0212;  %k43
p5=0.23;    %k64
p6=0.101;   %k56
p7=0.23;    %k57

Nreceptor = 10000;
p = [p1; p2; p3; p4; p5; p6; p7];
x0 = [0; Nreceptor; 0; 0; 0; 0; 0];

rng(1);

Nstep=100000;
stptime = zeros(Nstep,1);
time = zeros(Nstep,1);
xall = zeros(Nstep,7);
xall(1,:) = x0;
x = x0;

for step = 1 : Nstep - 1
   [xnew, tau] = MC_stochastic_update(x, p);
   x = xnew;
   stptime(step+1) = tau;
   time(step+1) = time(step) + tau;
   xall(step+1,:) = x;
end

figure(1)
set(gca,'FontSize',16);
plot(time,100*xall(:,1:7)/Nreceptor,'LineWidth',3);
legend('x1','x2','x3','x4','x5','x6','x7');
title('Monte Carlo Stochastic Dynamics of 10000 Receptors vs. time')
```

```
xlabel('time (min)');
ylabel('% of receptors');

figure(2)
set(gca,'FontSize',16);
plot(1:length(stptime),stptime,'LineWidth',3);
ylabel('time (min)');
xlabel('simulation step');

figure(3)
set(gca,'FontSize',16);
plot(1:length(time),time,'LineWidth',3);
ylabel('time (min)');
xlabel('simulation step');

%% waiting time analysis

times = zeros(1,5);

for N = 1:5

Nir = 10^N;
p = [p1; p2; p3; p4; p5; p6; p7];
x0 = [0; Nir; 0; 0; 0; 0; 0];

rng(1);

Nstep=Nir*10;
stptime = zeros(Nstep,1);
time = zeros(Nstep,1);
xall = zeros(Nstep,7);
xall(1,:) = x0;
x = x0;

for step = 1 : Nstep - 1
   [xnew, tau] = MC_stochastic_update(x, p);
   x = xnew;
   stptime(step+1) = tau;
   time(step+1) = time(step) + tau;
   xall(step+1,:) = x;
end

times(N) = mean(stptime);

end

figure(4)
set(gca,'FontSize',16);
plot((1:5),log(times),'LineWidth',3);
ylabel('log of mean waiting time (log(min))');
xlabel('log10(number of receptor)');
```

Monte Carlo Stochastic Update

```
function [ynew, tau] = MC_stochastic_update(y, p)

k25 = p(1);
k32 = p(2);
k72 = p(3);
k43 = p(4);
k64 = p(5);
k56 = p(6);
k57 = p(7);

rates = [k32 * y(2);
    k43 * y(3);
    k64 * y(4);
    k56 * y(6);
    k25 * y(5);
    k72 * y(2);
    k57 * y(7);];

lambda = sum(abs(rates));

transition = [0 0 0 0 0 0 0;
    -1 0 0 0 1 -1 0;
    1 -1 0 0 0 0 0;
    0 1 -1 0 0 0 0;
    0 0 0 1 -1 0 1;
    0 0 1 -1 0 0 0;
    0 0 0 0 0 1 -1;];

ynew = y;

tau = log(1/rand()) / lambda;

r = rand()*lambda;

current = 0;
selection = 1;

for it = 1:length(rates)
    current = current + rates(it);
    if current > r
        selection = it;
        disp(selection);
        break;
    end
end

ynew = ynew + transition(:,selection);
ynew = ynew.*[ynew >= 0];

end
```

Monte Carlo Stochastic Timespace

```
clear all; close all

p1=0.0737;  %k25
p2=1.29;    %k32
p3=0.0411;  %k72
p4=0.0212;  %k43
p5=0.23;    %k64
p6=0.101;   %k56
p7=0.23;    %k57

Nsim = 100
ip = zeros(5000,Nsim);
in = zeros(5000,Nsim);
bd = zeros(5000,Nsim);

rng(1);

for sim = 1:Nsim

Nreceptor = 1000;
p = [p1; p2; p3; p4; p5; p6; p7];
x0 = [0; Nreceptor; 0; 0; 0; 0; 0];

Nstep=5000;
stptime = zeros(Nstep,1);
time = zeros(Nstep,1);
xall = zeros(Nstep,7);
```

```
xall(1,:) = x0;
x = x0;

for step = 1 : Nstep - 1
    [xnew, tau] = MC_stochastic_update(x, p);
    x = xnew;
    stptime(step+1) = tau;
    time(step+1) = time(step) + tau;
    xall(step+1,:) = x;
end

% phosphorylated = y(3) + y(4) + y(6)
% internalized = y(4) + y(5) + y(6) + y(7)

% internalized and phosphorylated = y(4) + y(6)
% internalized and non-phosphorylated = y(5) + y(7)
% bound = y(1) + y(2) + y(3) = y(2) + y(3)

ip(:,sim) = 100*(xall(:,4) + xall(:,6))/Nreceptor;
in(:,sim)  = 100*(xall(:,5) + xall(:,7))/Nreceptor;
bd(:,sim)  = 100*(xall(:,2) + xall(:,3))/Nreceptor;

end

figure(2)
% for sim = sim = 1:50
plot3(ip,in,bd,'LineWidth',0.5);hold on
% end
grid on
xlabel('x4 + x6 (internalized/phosphorylated)');
ylabel('x5 + x7 (internalized/non-phosphorylated)');
zlabel('x1 + x2 + x3 (bound)');
```