Simulation of the Insulin Pathway

Master Thesis at ETH Zurich

Matthias Graf

March 15, 2012

Abstract

Insulin and insulin like growth factors play an important role in embryonic and postnatal cell growth and they are also suspected to influence tumor growth. The aim of this thesis is to present a biochemical reaction model of the insulin signaling pathway and use it to simulate this system using different techniques like ordinary differential equations (ODE) and stochastic simulate algorithms (SSA) under various conditions. From those result the sensitive parameters can be extracted and their their effects explored. Finally the thesis shows some guideline how those parameters in order to optimize the model under different assumptions.

Contents

1. Introduction

1.1. Insulin Induced Cell Growth

Insulin and insulin like growth factors play an important role in embryonic and postnatal cell growth[2][7]. Besides that, they are also suspected to influence tumor growth. [6].

The cells reaction to insulin is handled by the insulin signal pathway. In order to understand this reaction, we would like to understand the biochemical processes forming this pathway.

Out of experimental data collected by the group of Ernst Haven[1], Rajesh Ramaswamy[2] was able to generate a model of this pathway.

1.2. Biochemical Modeling of the Insulin Pathway

Insulin	High	Low
Foxo	Low	High
Foxo-P	High	Low
4EBP-P	High	Low
4EBP	Low	High
S6K	Low	High
S6K-P	High	Low

Figure 1.1.: The behaviour of the steady concentration level of the most important output species for varying insulin inputs. The suffix -P stands for the phosphorized version of the protein

The biochemical reaction network behind the insulin signaling pathway can be modeled as an open driven reaction system, consisting of 67 species and 132 elementary reactions. It is weakly coupled, and we do not have any further information about reaction rates, fluxes or populations.

The system is driven by the insulin input, given as the flux through the input reaction rin.

The outputs of the system are the concentrations of the three species Forkhead box O3 (Foxo), 4E-BP1 (4EBP) and S6 kinease (S6K).

The general behaviour of the steady concentrations of those output species have already been described [15] and can be seen in figure 1.1.

To determine the system behaviour more accurate we have to collect experimental data by measuring the molecular concentration of different species for various insulin inputs. In

[1]Institute of Molecular System Biology,ETH Zurich
[2]Institute of Theoretical Computer Science, ETH Zrich

order to optimize those measurements it is necessary to understand the system behaviour as good as possible.

2. Model Refinements

The reaction network described so far bases on experimental data and has some defects that need to be fixed.

2.1. Remove Elementary and Reversible Reactions

The model consists out of 132 elementary reactions. In order to get an broad overview over the model behaviour, I converted those into non elementary reactions.

Now we can see that the model contains some duplicates (like reaction $r13$ is identical to reaction $r19$ etc.) and several reactions have their reverse included in the system.

I removed all the duplicates and put the reversible reactions together.

2.2. Correct Source and Sink Reactions

Due to the conservation of mass every species consumed (or generated) has to be generated (consumed). In the original model the 6 species ADP, Foxo, Lobe, TSC, PKBaa, and PIK3ka were generated but not consumed.

Furthermore the species PKBa1 is consumed only if the concentration of TORC2a is high enough, which might not be the case.

To model the process consuming those species I added 7 destruction reactions.

2.3. Modelling Errors

In the original model there was the reaction

$$InR2a \ + \ TORC2 \ \rightarrow \ TORC2a \tag{2.1}$$

Since InR2a, InR2 and InR1 are supposed to form a conserved moiety (thus none of them is generated nor consumed, they are only transformed into each other, see chapter 3.1.1) none of them can be used as a reactant. There are three possible corrections for this error

- The InRXes do not form a conserved moiety. Since they are not yet generated a source reaction for at least one of them has to be added.

$$\emptyset \ \rightarrow \ InR2 \tag{2.2}$$

- The reaction uses InR2a not as an reactant, but as a modifier, thus reaction 2.1 has to be replaced with

$$InR2a \ + \ TORC2 \ \rightarrow \ TORC2a \ + \ InR2a \tag{2.3}$$

- The reaction do not use InR2a at all, thus reaction 2.1 has to be replaced with

$$TORC2 \rightarrow TORC2a \tag{2.4}$$

Since the correction 2 is the most likely, I replaced reaction 2.1 with 2.3.

2.4. The Behavior of S6 Kinease

We know from figure 1.1 that the population of S6 Kinease (S6K) should drop and the population of S6Ka should increase for growing insulin input.

So far there are three reactions involving S6K

$$\begin{align} S6Ka &\rightarrow S6K \tag{2.5}\\ S6K + TORC1a &\rightarrow S6Ka + TORC1a \tag{2.6}\\ \emptyset &\leftrightarrow S6K \tag{2.7}\end{align}$$

Beside those there is only one additional reaction that involves S6Ka

$$TORC2a + S6Ka \rightarrow S6Ka + TORC2 \tag{2.8}$$

One can see from those reactions that at a steady state of S6Ka the flows through of the reactions 2.5 and 2.6 has to be equal, therefore the steady concentration of S6K is defined only by the reaction 2.7.

Since this reaction is constant the steady concentration of S6K has to be constant as well, opposite to the perquisites expected in figure 1.1.

There are two ways how we can correct this error

- The reaction 2.7 is either wrong or has a far lower time scale that the rest of the model. In both cases we can ignore the effects of this reaction for the time scale we are interested in

- This reaction is correct but there is another reaction (possible an output reaction) that consumes S6Ka.

$$S6Ka \rightarrow \emptyset \tag{2.9}$$

Since variant 1 is the most likely, reaction 2.7 has to be removed from the model.

2.5. Model Describing Language

In order to use standard tools to examine the model, it has to be brought into a standard format. The most widely used format to describe biochemical networks is SBML[12] (Systems Biology Markup Language). SBML is a XML based format especially developed to describe such networks. There are a lot of tool using SBML, including *Cell Designer*[5] and the *Systems Biology Toolbox for MATLAB*[18].

2.6. Naming Conventions

To be able to distinguish between the different kinds of variables all the reaction name will begin with a small r, followed by either a number or $outXXX$ / $inXXX$ if it is a in-/ output reaction of species XXX. If the reaction is opposite to another one, the name will be followed by the suffix r.

The reaction rates for each reaction will be named after the reaction name followed by a capital P, and eventually a number (if there are several reaction rates parameters for a single reaction). The parameters of the reverse reaction will be followed by the suffix Rev.

Due to the technical limitations variable names starting with a number are causing problems. Therefore variable names starting with 4EBP will be renamed to X4EBP

2.7. Conclusions

The model after those transformations can be found in appendix B. It consists of 36 species, 56 reversible and 35 non reversible reactions.

3. Network Overview

3.1. General Remarks

3.1.1. Conserved Moieties

There are several molecular subgroups that have a constant mass over time. Such subgroups are called *conserved moieties*[3]. Given the stoichiometry matrix $\mathbf{V}$ with size $[\#species \times \#reactions]$ the number of independent species is given by $rank(\mathbf{V})$[3].

For the model given the rank of the stoichiometry matrix is 35 and the number of species 38, so there have to be 3 dependent species. There are different possibilities how to choose the dependent species (if the sum of populations of A and B is constant, you could either claim that A is dependent of B, or B is dependent of A). A set of dependent rows in a rank deficient matrix can be found using the **LU** decomposition[3]. For the model given the dependencies can be expressed as

$$\frac{d}{dt}[InR2a] = -\frac{d}{dt}([InR2] - \frac{1}{2}[InR1]) \tag{3.1}$$

$$\frac{d}{dt}[S6K] = -\frac{d}{dt}[S6Ka] \tag{3.2}$$

$$\frac{d}{dt}[TORC2] = -\frac{d}{dt}[TORC2a] \tag{3.3}$$

$$\tag{3.4}$$

So the sums

$$[S6K] + [S6Ka] \tag{3.5}$$

$$[TORC2] + [TORC2a] \tag{3.6}$$

$$[InR2a] + [InR2] + \frac{1}{2}[InR1] \tag{3.7}$$

are constant over time, and are thus conserved moieties. I will denote them as InRX, S6KX and TORC2X.

3.1.2. Species with Constant Concentration Levels

A species S has a constant concentration level S_{CONST} if the species is not involved as reactant or product in any reaction with a nonzero flux while its concentration is S_{CONST}.

If the species is involved only in a in-/output reaction

$$\emptyset \leftrightarrow S \tag{3.8}$$

with a flux

$$S_{in} - [S] * S_{out} \tag{3.9}$$

it has a constant level at $S_{CONST} = S_{IN}/S_{OUT}$. The species PTEN, GEF and PDK1 are not involved as reactant or product in any other reaction that their input reaction and are therefore constant.

The species Rheba, TORC1 and PIP2 are involved in other reactions than their input reaction, namely

$$\begin{aligned} PIP2 \ + \ PI3Ka \ &\rightarrow \ PI3Ka \ + \ PIP3 && (3.10)\\ PIP3 \ + \ PTEN \ &\rightarrow \ PTEN \ + \ PIP2 && (3.11)\\ Rheba \ + \ TSCa \ &\rightarrow \ TSCa \ + \ Rheb && (3.12)\\ Rheb \ + \ GEF \ &\rightarrow \ GEF \ + \ Rheba && (3.13)\\ TORC1 \ + \ Lobea \ &\leftrightarrow \ Lobea \ + \ TORC1a && (3.14)\\ TORC1a \ + \ Rheba \ &\leftrightarrow \ Rheba \ + \ TORC1 && (3.15)\\ & && (3.16) \end{aligned}$$

Since also Rheb,TORC1a and PIP3 are not involved in other fluxes than those the flux through reaction 3.10 and 3.10, 3.12 / 3.13 and 3.10 / 3.10 have to be equal at steady state, since if they would not, the population of Rheb, TORC1a or PIP3 would not be steady.

Therefore the concentration of Rheba,TORC1 and PIP3 at steady state has to be S_{CONST} defined by their input reaction, so they have a constant steady state concentration.

3.1.3. In- and Output Fluxes during Steady States

The insulin signaling pathway is not a closed system. At steady state there are 6 flows interacting with the environment. Those can be seen in figure B where they are marked with different colors

- The Insulin inflow is consumed by the InRX-cycle

$$\begin{aligned} InR2a \ &\rightarrow \ InR2 && (3.17)\\ InR2 \ + \ Insulin \ &\rightarrow \ InR2a && (3.18) \end{aligned}$$

  Due to the conservation of mass this cycle can be supposed to produce a unknown by-product S, thus reaction 3.17 should be replaced with the reactions

$$\begin{aligned} InR2a \ &\rightarrow \ InR2 \ + \ S && (3.19)\\ S \ &\rightarrow \ \emptyset && (3.20)\\ & && (3.21) \end{aligned}$$

  but since S is not used in this model it can be left away.

- The *chicoFlow* through Chico, Chicoa and p60/p110, PI3K to PIK3K. On figure B the *chicoFlow* is marked in red.

- The $PKBFlow$ thought PKBI to PKB and then via PKBa1 or PKBa2 to PKBaa. On figure B the $PKBFlow$ is marked in green.

- The 3 *Deactivation flows* from Foxoa, TSCa and Lobea to Foxo, TSC and Lobea. On figure B the deactivation flows are marked in different blue colors.

3.1.4. Assumption on the ATP Level

Several reactions in the model consume energy by transforming ATP to ADP. Since I do not assume the insulin signaling pathway to be a critical process in the energy household of a cell, I assume the concentration of ATP not to drop significant if the insulin concentration grows. Therefore I model the concentration of ATP as constant.

3.1.5. The Effects of Insulin Input

If the insulin input increases, the following happens in the system:

- A part of the mass in the conserved moiety of InRX transforms into InR2a. Since the mass of InR2a cannot become bigger than the total mass in InRX, the concentration saturates for high insulin inputs.
- A nonzero concentration of InR2a induces the *chicoFlow* through Chico, Chicoa and PI3Ka. This flow not only increases the concentrations of Chicoa and PI3Ka, but also decreases the concentrations of p110, p60, PI3K and Chico.
- A growing concentration of InR2a also causes the concentration of TORC2a to grow.
- A growing concentration of PI3Ka translates into a growing concentration of PIP3.
- A growing concentration of PIP3 induces the $PKBFlow$ through PKBI, PKB to PKBa1, PKBa2 and PKBaa. The total mass of this flow is determined by the concentration of PIP3, the percentage that goes through PKBa1 and PKBa2 by the concentration of TORC2a.
- A growing concentrations of the different active variants of PKB induces the deactivation flows for the species TSC, Lobe and Foxo. This causes the concentration of the active variants of those species to drop and the concentration of their deactivated variant to increase.
- A dropping concentration of TSCa increases the concentration of Rheb. It also temporarily drops the concentration of Rheba, but this levels out as described in chapter 3.1.2.
- A dropping concentration of Foxoa causes the concentration of 4EBPa to drop, this again drops the concentration of 4EBP.
- A dropping concentration of Lobea has an influence on TORC1a and a temporary influence on TORC1. But since the reaction $r51$ is reversible is not clear whether this influence is positive or negative. A change in the concentration of Rheba has the same (unknown) effect, but since the concentration of Rheba only temporarily differs from its constant steady concentration this influence is again only temporary.
- A growing concentration of TORC1a causes a growing concentration of 4EBPa. Due to figure 1.1 we know that this has to be the case if the insulin input grows. Since there are no other reactions that could cause this effect we can conclude that a dropping concentration of TSCa increases the concentration of TORC1a and drops the concentration of TORC1.
- A growing concentration of TORC1a causes the concentration of S6K to drop and the concentration of S6Ka to grow.

- A growing concentration of S6Ka drops the concentration of TORC2a and increases the concentration of TORC2. This effect is opposite to the increase of TORC2a induced by the growth of the InR2a concentration.

3.2. Division into Subsystems

For further analyzes I divided the system into three subsystems. This division can be seen in figure B.

3.2.1. The Input-Subsystem

The input-subsystem works as a signal converter. It receives a signal (encoded in the insulin input) and applies two transformations:

- The input range of insulin is open, thus there could be any (nonnegative) insulin input. The input-subsystem transform this into a suitable range.

- The insulin input might be noisy and unstable. The input-subsystem smoothes out fast changes

The outputs of the input-subsystem are the two concentrations (PIP3 and InR2a) in a well defined range.

Note that InR2a reacts faster and less smooth on a change in the insulin input than PIP3.

3.2.2. The PKB-Subsystem

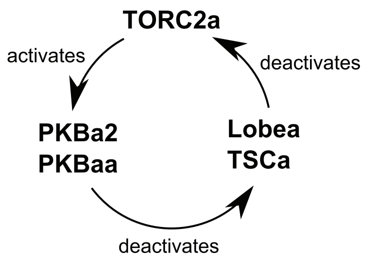


Figure 3.1.: The self regulating circle of TORC2a

The PKB-subsystem produces PKB (the amount depends on the concentration of PIP3) and transforms this into its active (phosphorized) variants. PKB can be phosphorized using PDK1 (then denoted as PKBa1), TORC2a (then denotes as PKBa2) or one after the other (then denoted as PDKaa). Since the concentration of PDK1 is constant (see chapter 3.1.2) the percentage of PKB that is activated to those different forms depends only on the concentration of TORC2a.

All those three variants induces the deactivation flows for Lobea, TSCa and Foxoa, but we do not know which variant influences which species how fast.

However we know that a low concentrations of Lobea and TSCa indirectly deactivates TORC2a (done in the regulation-subsystem). Therefore we can assume that the species activated by TORC2a especially deactivates those species, so that we get a self regulating circle as can be seen in figure 3.1.

The remaining concentration of the activated versions of Lobe, Foxo and TSC influences the regulation-subsystem. The deactivated species have no further use and will be removed.

3.2.3. The Regulation-Subsystem

The regulation-subsystem controls the concentrations of TORC2a and the output species 4EBP and S6K. They are steered by the concentrations of the active variant of TSC, Lobe and Foxo.

3.3. A Reduced Steady State Model

In order to do a first examination of the system behaviour on steady states, I tried to find a simplified model that shows the same steady state behaviour as the model described so far.

3.3.1. Model Description

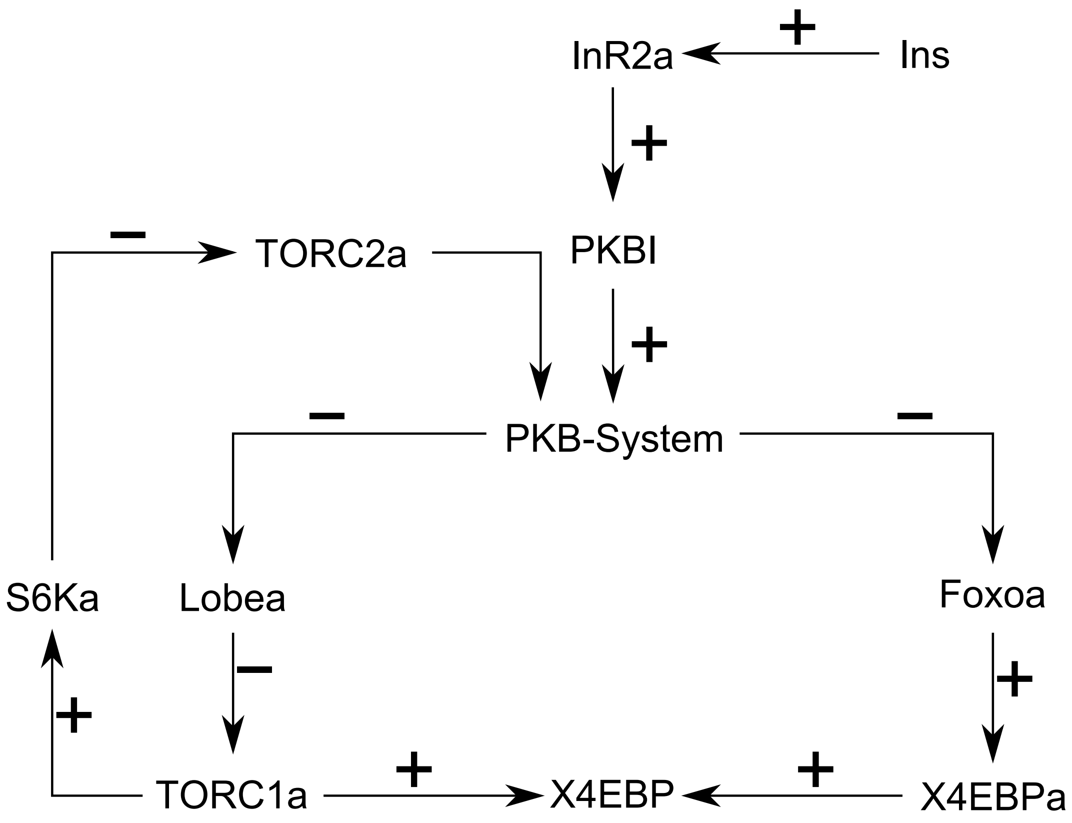


Figure 3.2.: The reduced steady state model. A plus indicated that a species grows if its input does, a minus the opposite.

The reduced model does not need to include all species, since we are only interested in the behavior of the output species Foxo, S6K and Lobe. The model should include the following species:

- The output species 4EBP, 4EBPa, Foxoa and S6K. Since S6Ka and S6K form a conserved moiety, the concentration of S6Ka can be computed from the concentration of S6K directly and there is no need to simulate it.
- The input species insulin
- To simplify the computations the intermediate species InR2a, PKBI, Lobea, TORC1a and TORC2a.

The resulting system can be seen in figure 3.2.
In order to find the relations between those species at steady state the equation system

$$\frac{\partial}{\partial t}\vec{p} = \vec{0} \tag{3.22}$$

for the concentrations $\vec{p}$ can be solved analytically for every species in the model. This is straight forward and I have done it in appendix A.2. However there were some relations that were hard to compute:

- Ins $\rightarrow$ InR2a: The concentration of InR2a depends only on the concentration of Ins. If we solve the equation system defined by the steady state hypothesis we can compute the concentration for InR2a as

$$InR2a = \frac{Ins * initInRX/2}{\frac{r2rP}{r2P} + Ins} + \frac{r2P(Ins * (r2P \pm \sqrt{...})}{r1P * (r2P + Ins * r2P)^2} \tag{3.23}$$

  This is nearly a Michaelis-Menten relation, but due to the quadratic kinetics of reaction $r1$ (two InR1 form one InR2a) there is this other (higher order) term. In order to simplify this model further I approximate this with a Michaelis-Menten relation.

- In order to compute the influence of TORC2a and PKBI on the concentration of Lobea and Foxoa I first compute the cumulated sum through the deactivation fluxes. Therefore I need to compute the concentrations of the different PKB variants. In appendix A.2 I show that those concentrations can be expressed as a broken rational function of degree 2 (in TORC2a) and 1 (in PKBI). Therefore also the deactivation flux (that is essential a weighted sum over those concentrations) can be expressed in this way. For the concentrations of Foxoa and Lobe we can then use

$$concentration = \frac{input}{output + deactivationFlow(TORC2a, PKBI)} \tag{3.24}$$

Now all this relations still have some parameters that have to be set. There is a total of 46 parameters in this system:

- 10 parameters are defining the concentrations of the species for maximal insulin input. Since the system is scaling invariant, those parameters define the scale but do not influence the system any further.
- 6 parameters are defining how far the concentration maximally drops (or increases) if insulin grows to infinity.
- 5 parameters are defining the reactions rates of the source reactions into the systems (for PKBI, Foxoa, TSCa, Lobea and 4EBPa).
- 25 Parameters are fully unknown and have to be guessed.

3.3.2. Simulation Results

The simulation results are showed in figure 3.3. As expected, the behaviour is similar to the results of simulating the full model using ODEs (in chapter 5) or SSA (in chapter 7).

The exact computation and the parameter chosen can be found in appendix A.2.

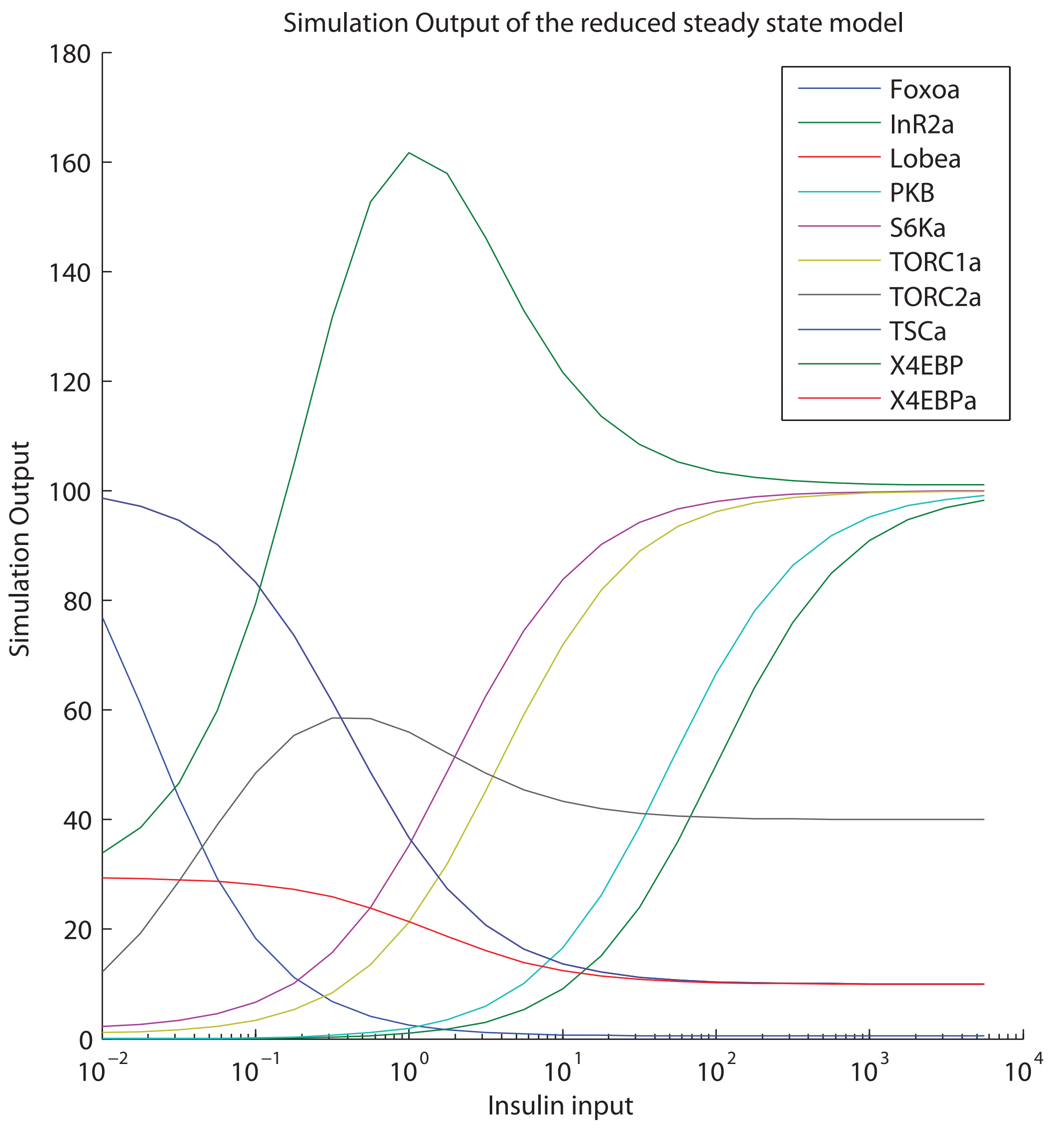


Figure 3.3.: Simulation of the reduced steady state model

4. Model Parameters

In this chapter I try to identify the variables of the model.

4.1. The Kinetic Law

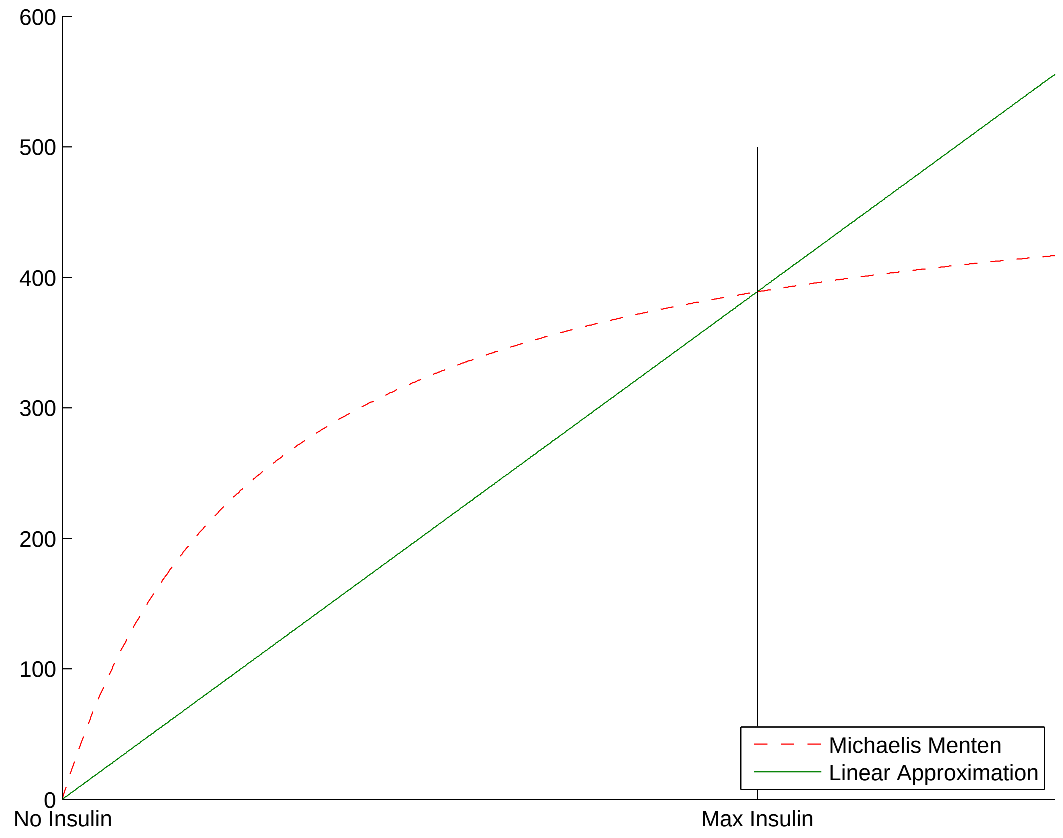


Figure 4.1.: Linear approximation of Michaelis-Menten

Several reactions in the model are transitions of one species into another using a third as modifier. One way to describe the kinetics of such a reaction is Michaelis-Menten[13].

Given the concentration of the species S and the concentration of the enzyme E the kinetics of the reaction is given by

$$Flow := \frac{V_{Max} * [S][E]}{K_m + [S]} \tag{4.1}$$

Since I want to reduce the number of parameters, I am using the linear approximation

$$Flow := \frac{V_{Max}}{K_m + [S]}[S][E] \approx \mu[S][E] \tag{4.2}$$

with a constant reaction rate μ. In order to find μ it has to be approximated at a specific point p. As we can see in figure 4.1 the approximation error is bounded as long we are below this point, and grows to infinity if we get above.

As approximation point I use the steady state with maximum (infinity) insulin input. As we saw in chapter 3.2.1 the system still saturates and we can be sure that we will never get above this approximate point.

4.2. Dimension Analysis

4.2.1. Model Variables

If we assume linear kinetics as described in chapter 4.1 the 77 reaction rates are the system variables.

Beside the rates the conserved moieties described in chapter 3.1.1 need to be filled with an initial amount. Last but not least we have to define the size of the .

So there is a total of 81 variables in this system.

4.2.2. Buckingham Π-Theorem

As described by the Buckingham Π Theorem [20] we can define the system by "number of variables" minus "number of dimensions" independent dimensionless groupings.

The dimensions of the parameters in this model are

- For the reactor size: $Size^3$
- For the initial filling of the conserved moieties: $Mass * Size^{-3}$
- For dimension of the reaction rates depends on the number of inputs (reactants and modifiers) of the reaction:

$$Mass^{1-\#Inputs} Size^{-3*\#inputs} * Time^{-1}$$

Since this model uses Time, Space and Mass as dimensions, 78 parameters are enough to describe the system.

4.2.3. A First Grouping

A dimensionless grouping can be found using E.S. Taylor's method [20]:

- Multiply the initial fillings of the conserved moieties once and the reaction rates $\#Input$ times with the reactor size and remove the reactor size as parameter.
- Choose one of the initial fillings and divide all other fillings once and the reaction rates $(1 - \#input)$ times through this filling. Then remove the filling chosen.
- Choose one reaction rate and divide all the other reaction rates by this one. Then remove it

This way we end up with 78 dimensionless parameters.

4.2.4. Disadvantages of the First Grouping

This simple dimensionless grouping has a big disadvantage: Each parameter influences the whole system. So for example the reaction rates $r2P$ and $r2rP$ of the reactions $r2$ and $r2r$ both influences the steady concentrations of all species in the model. If we would use the reaction rate $r2P$ and the ratio $r2P/r2rP$ between the reaction rates as variables instead, only the ratio would influence the steady states, whereas $r2P$ would determine how fast the steady state will be reached.

We can come up with (nearly) infinitive possible groupings, and all might have their advantages and disadvantages.

4.2.5. An optimized Grouping

I tried to find a grouping that uses a small number of parameters to describe the steady states for all species. Therefore I came up with my own parameter grouping that uses as often as possible the concentrations of a certain species for a given insulin concentration (zero or infinity) as parameter. This has several advantages

- Due to the scaling invariance of this model (at least as long as we use ODE to simulate is) those parameters have no influence on the other species.
- I can guarantee that the concentration of all species lies within a reasonable range.
- The concentration ranges of the species are similar, so they can be compared easily.

Besides this scaling parameters I try to isolate parameters that have no influence on the steady states as described int he example in chapter 4.2.4.

Parameter Range Transformations

In order to end up in a valid model, I have to make sure that the reaction rates computed out of my parameter set will be positive all the time. For some parameters this defines a range within they have to be. Since I want the parameters to be valid for all nonnegative values, I have to transform those parameter ranges.

If a parameter P_{old} is only valid in the range $[upper, lower]$ it has to be replaced with the parameter P_{new} with

$$P_{old} = \frac{(upper - lower) * P_{new}}{P_{new}} + lower \tag{4.3}$$

Now for every nonnegative choice of P_{new} the parameter will lie within the valid range.

The parameter set and the description of a dimensionless grouping using this set can be found in appendix A.1.

4.3. Parameter Choice

Now we have to assign values to those parameters. I will do this so that the maximal population is 100 for all species. If a species has a constant populations it should be 100.

Besides this all the reaction rates have to lie within the borders described in table 4.2.

Due to the dimension reaction done in chapter 4.2.5 I can choose the size of the reactor arbitrarily. I chose it as $10^{-6}M^{-1}$.

The exact values values chosen for the other parameters can be found in appendix A.1.

Number of reactants	Lower Boundary	Upper boundary
0	10^2 M s^{-1}	10^4 M s^{-1}
1	10^3 s^{-1}	10^6 s^{-1}
2	10^4 M^{-1} s^{-1}	10^6 M^{-1} s^{-1}

Figure 4.2.: Range of the reaction rates for different kind of elementary reactions as described by [4]

4.4. Steady-State Analysis

Now I used those parameters to compute the reaction rates and then use those reaction rates to find all possible steady states.

In appendix A.1 I show that the system is monostable for every insulin input.

5. Simulation Using ODEs

The parameters described in chapter 4 can now be used to simulate the model using a system of ordinary differential equations[17]. To do this simulation I used the *Systems Biology Toolbox for MATLAB* [18] that is using the standard Matlab ODE solver.

5.1. Results

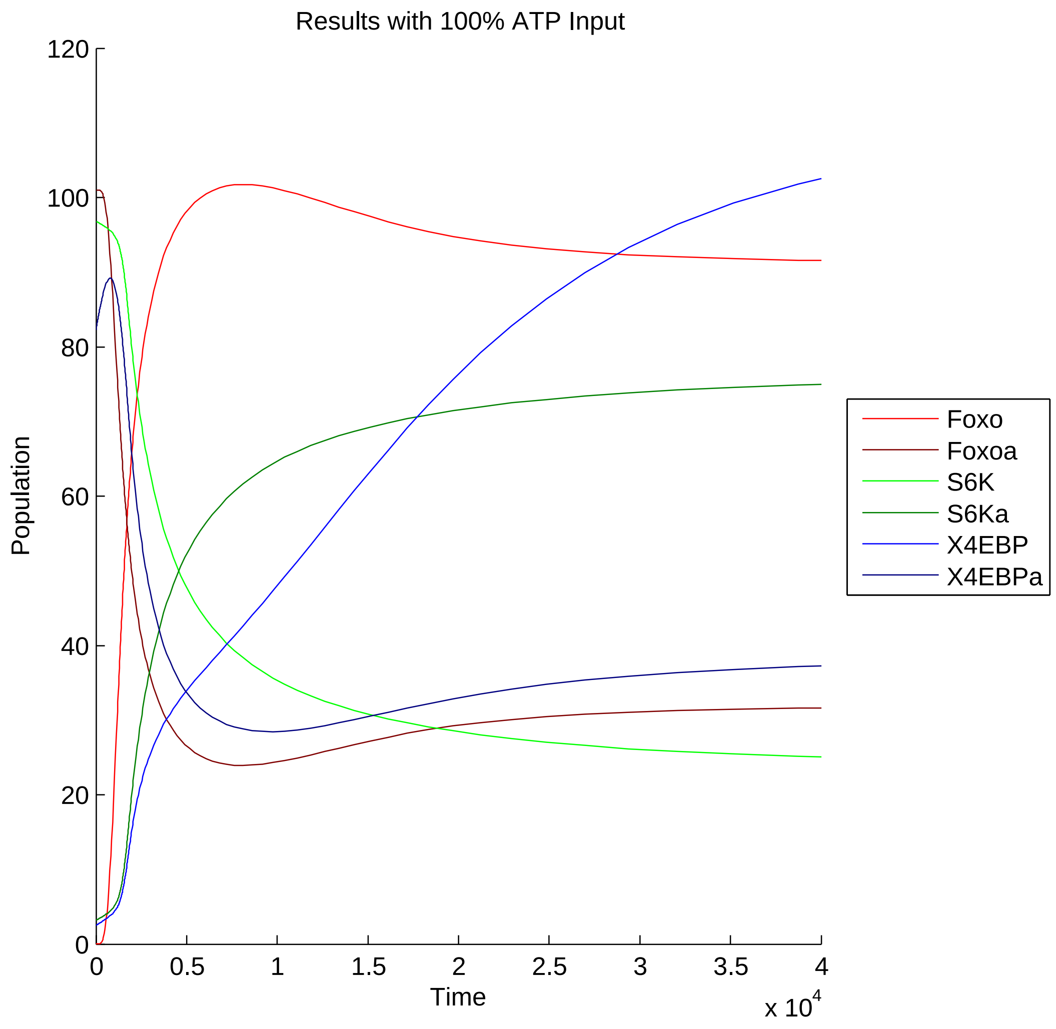


Figure 5.1.: Simulation result for the most important species

The behaviour of the simulated concentration over time for the output species using the parameters defined in chapter 4.3 can be seen in figure 5.1. As can be verified easily, the model behaves as required in table 1.1.

5.2. The Effect of Different Insulin Inputs

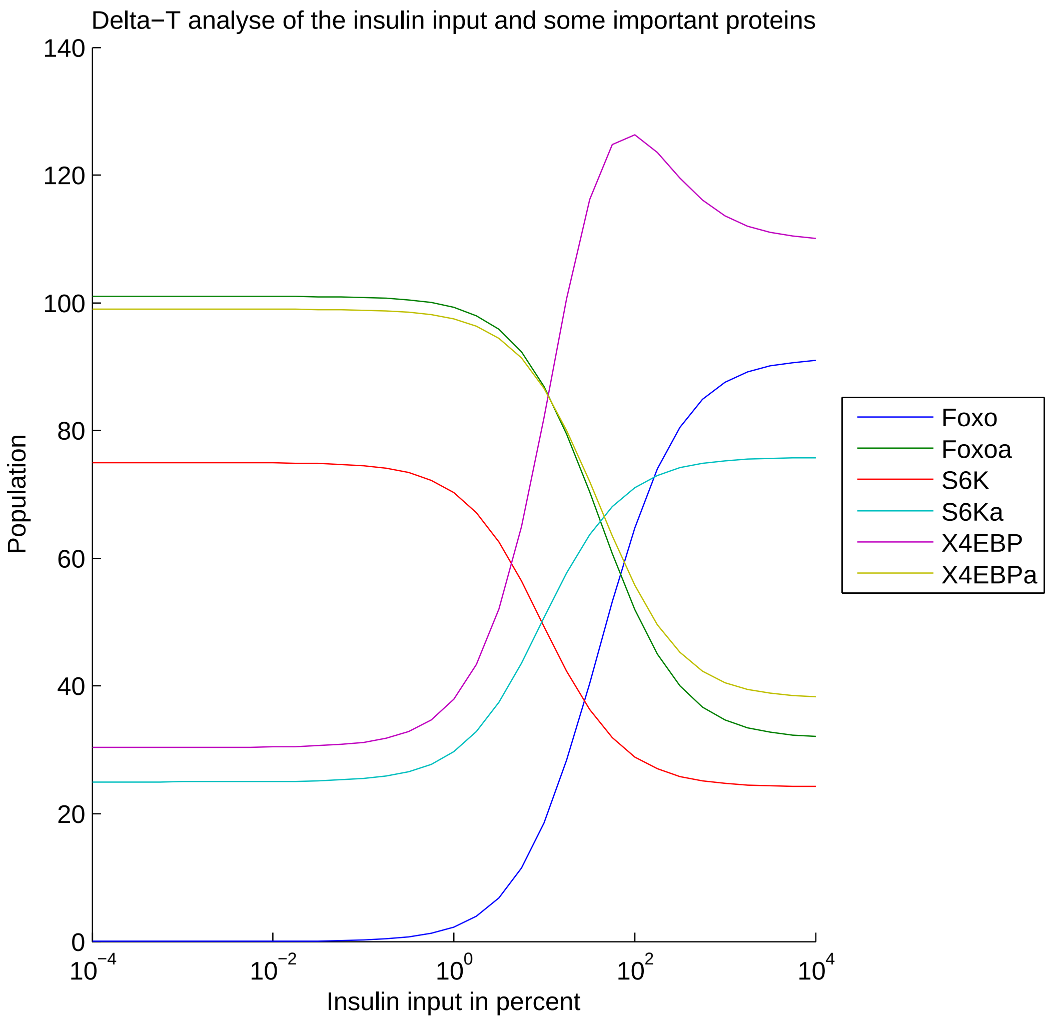


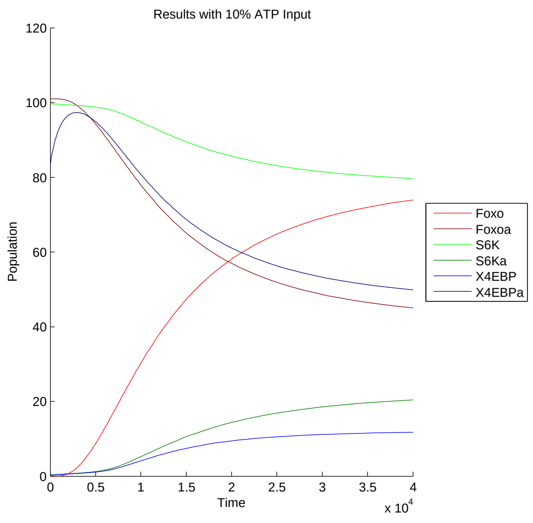
Figure 5.2.: Simulation result for different insulin inflows

As I showed in chapter 4.4 there is exactly one steady state for every insulin input. The development of the steady concentrations for a growing insulin input can be seen in figure 5.2.

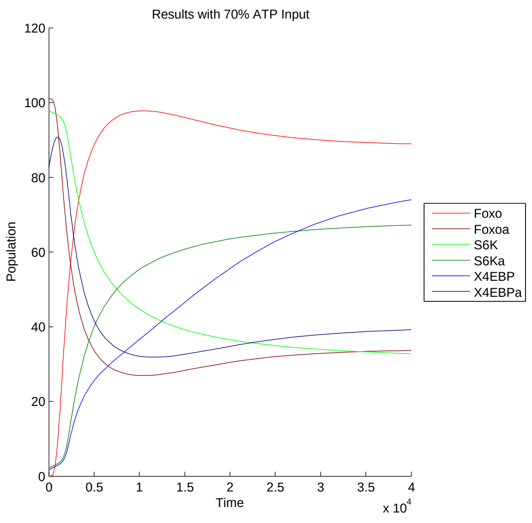
As expected in figure 1.1 there are 3 species with decreasing, and 3 with increasing populations. Most of them behaves monotonous for raising insulin, only $X4EBP$ shows its maxima at a midrange insulin input. I further examine this behaviour in chapter 6.5.

Due to the range transformation of the input-subsystem (described in chapter 3.2.1) the system saturates for high insulin inflows.

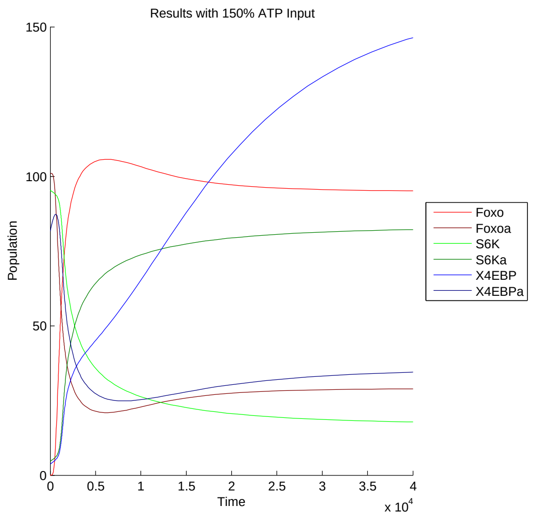
5.3. The Influence of ATP



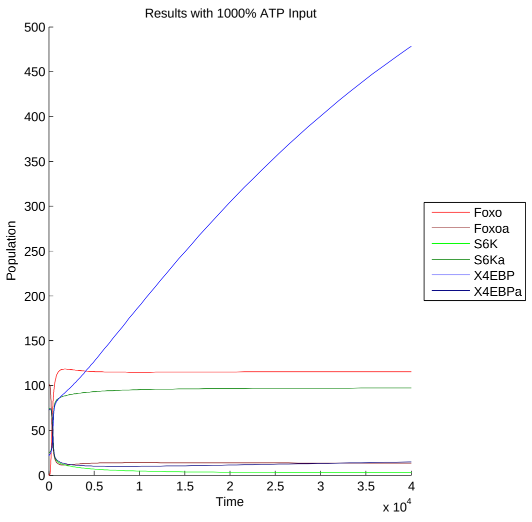
(a) Fixed ATP level of 10%



(b) Fixed ATP level of 70%



(c) Fixed ATP level of 150%



(d) Fixed ATP level of 1000%

Figure 5.3.: Behaviour for different levels of ATP

In chapter 5.3 I assumed there is a fixed concentration of ATP in the system. But what happens if the cell starves and the ATP concentration drops?

If we decrease the ATP-level, the system generally reacts slower and less intense on a change of insulin inflow. The concentrations of S6K and 4EBPa do not drop and the levels of S6Ka and 4EBP do not increase as strong as for full insulin input.

If the ATP level is too small the level of S6Ka and 4EBP might not even become higher than the level of S6K and 4EBPa. So for lower levels of ATP the system will not show the behaviour defined in figure 1.1 any more.

If we increase the ATP-level in the cell the concentrations of S6Ka and 4EBP without insulin present grows. For an ATP inflow high enough, the levels of S6Ka and 4EBP might even be higher than the level of S6K and 4EBPa even if no insulin is present.

Thus in order for the system to work as defined in table 1.1 the concentration of ATP in the cell has to be in a well defined range. If the concentration drops below this level the concentration of the output species will not grow even for the highest insulin inflow, and if the ATP level is to high the output species are already maximal if no insulin is present. In between the ranges the level of ATP has a big influence on the sensitivity for insulin.

6. A closer look at the influence of some selected Parameters

6.1. Important Parameters for Steady State with no Insulin

From the set of parameters derived in chapter 4.2.5 we want to identify the parameters most important for the behaviour of the output species.

If no insulin is present the populations of those species can be computed directly from the parameters:

$$\begin{aligned}
Foxoa &= foxoaNoP \\
Foxo &= 0 \\
4EBPa &= X4EBPaMax \\
4EBP &= X4EBPMax * (1 \\
&- ((TORCRangeParameter + 1) * (LobeaDrop + 1))^{-1}) \\
S6K &= initS6K * \tfrac{ratioS6KNoparameter}{ratioS6KNoparameter+1} \\
S6Ka &= initS6K - S6K
\end{aligned} \tag{6.1}$$

This is a direct consequence from the parameter choice.

6.2. Important Parameters for Steady State with Maximal Insulin

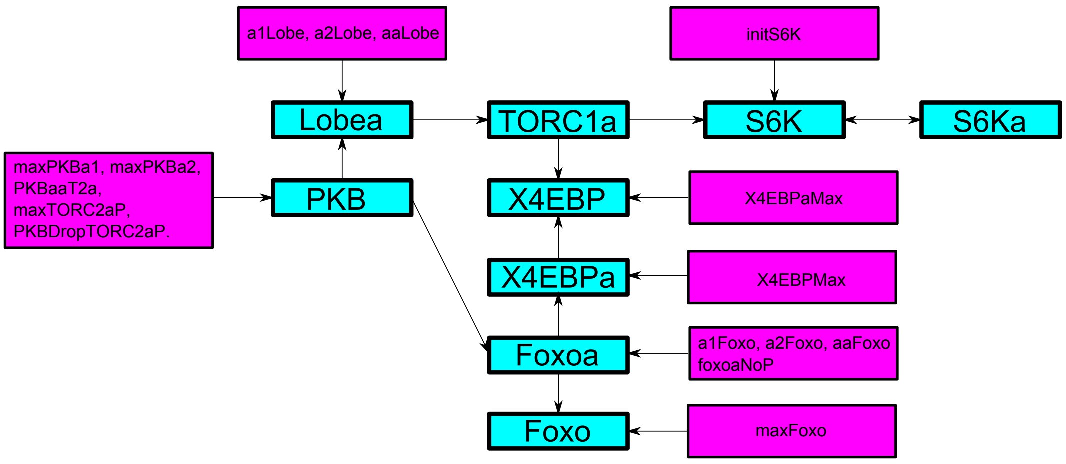


Figure 6.1.: The influence of the parameters on the important species on the steady state with maximal insulin input

The populations for maximal insulin influence cannot be computed directly. To get them one has to simulate the whole system. To detect the parameters that have an influence for maximal insulin input I simulated the system with different values for each parameter and check whether this change influences the steady population for any species. In figure 6.1 you can see all the parameters that have an influence on the steady state population of the output species of figure 1.1. As you can see, there is only a small number of parameters that influences this state. In this chapter I will have a closer look at them.

6.2.1. Scaling Parameters

Each of the output species needs a scaling factor, that defines the scale of the output. Since the model is scaling invariant, those parameters do not have any further influence.

A special case are the species S6K and S6Ka. Since they form a conserved moiety, they have only one scaling factor ($initS6Ka$).

initS6K

The parameter $initS6K$ describes the total population of S6K and S6Ka. Since they form a conserved moiety, this sum is constant. In figure 6.2 you can see how the populations of S6K and S6Ka grow with $initS6K$.

foxoNoP

$foxoNoP$ describes the population of Foxoa if there is no insulin present (thus the maximum population of Foxoa). This parameters defines the scale of Foxoa. In figure 6.3 you can see how the population of Foxoa grow with $foxoaNoP$

maxFoxo

$maxFoxo$ defines the maximal population of Foxo. This is the case if Foxoa is minimal, thus all active variants of PKB are maximal.

Those variants will never be maximal at the same time, so this minimal population will never be reached, but depending on the parameter choice it might get infinitely close.

In figure 6.4 you can see how Foxo grow with $maxFoxo$.

X4EBPMax

$X4EBPMax$ defines the maximal population of 4EBP. In figure 6.5 you can see how 4EBP grow with $X4EBPMax$.

X4EBPaMax

$X4EBPaMax$ defines the maximal population of 4EBPa. In figure 6.6 you can see how 4EBPa grow with $X4EBPaMax$.

6.2.2. Parameters that Influences the PKB-Subsystem

We know that the active variants of PKB deactivates Foxoa, Lobea and TSCa, but we do not know which variant deactivates which species how strong, nor do we know how much of which variant will be produced. Unfortunately those two uncertainties could hide each other (for example we could double the amount of PKBa1 but halve the sensitivity of Lobea for PKBa1 and would not notice any difference in the concentration of Lobea). Therefore it is impossible to describe the dynamics of the PKB-subsystem by measuring only the output species Foxoa, S6K and 4EBPa.

Parameters that Influences the Activation of PKB

First we want to examine the production of the PKB variants. The population of PKBa1, PKBa2 and PKBaa for different parameter choices are shown in figure 6.7.

As can be seen there, the parameters $PKBDropTORC2aParameter$ and $PKBa1DropTORC2aParameter$ have the biggest influence on the behaviour, and the parameters $maxPKBa1$ and $maxPKBa2$ on the scales of those species.

The influence of those parameters on Foxoa can be seen at figure 6.8. If $maxPKBa1$ is big, the influence of PKBa1 is bigger than the influence of $maxPKBa2$. Since PKBa1 will be present over proportionally when the insulin input is small (and therefore the population of TORC2a is low) a bigger influence of PKBa1 results in a fast drop of Foxoa, but the drop will not be that deep. If however the influence of PKBa2 is bigger, the drop will not be that fast, but will go deeper for big insulin input, since for big insulin inputs the influence of PKBa2 will be higher (due to the high population of TORC2a).

PKBDropTORC2aParameter

The parameter $PKBDropTORC2aParameter$ describes the drop in the PKB population if TORC2a becomes present. The population of PKB for maximal insulin and maximal TORC2a can be expressed as

$$maxPKB * \frac{PKBDropTORC2aParameter}{PKBDropTORC2aParameter + 1} \tag{6.2}$$

thus the lower the value of this parameter, the stronger the drop of PKB.

This parameters has not a big influence on the populations of PKBa2 and PKBaa since their behaviour for high values of TORC2a is steered by the parameters $maxPKBa2$ and $PKBaaT2a$.

Thus only the behaviour of PKBa1 will be changed. If the value of $PKBDropTORC2aParameter$ is low enough, the drop will be that fast that the population of PKBa1 will stay low, whereas for high values it will reach $maxPKBa1$ for higher insulin inflows. This behaviour can be seen in figure 6.9.

Due to the raising concentration of PKBa1 with raising level of $PKBDropTORC2aParameter$ the concentration of Lobea and Foxoa will be decreased. This can be seen in figure 6.10.

But what happens for high values of this parameter? Foxoa will only reach its minima, defined as

$$minFoxoa = foxoaNo * FoxoaDrop/(FoxoaDrop + 1) : \tag{6.3}$$

for the parameters $FoxoaDrop$ and $foxoaNo$, when the deactivation flow is maximal. The maximal deactivation flow is computed as

$$a1Foxo * maxPKBa1 + a2Foxo * maxPKBa2 + aaFoxo * maxPKBaa \tag{6.4}$$

This maxima will never been reached, since PKBa1 and PKBa2 will never have its maximal at the same time. But for intermediate values of $PKBDropTORC2aParameter$ the error of this approximation is low.

So for high values of $PKBDropTORC2aParameter$ the error becomes big, and therefore the concentration $minFoxoa$ cannot be reached any more. This can be seen in the fast raise in figure 6.10.

The dropping concentration of PKBa1 has an big influence on the levels of Foxoa and Lobea. We could expect that the higher we choose the value of $PKBDropTORC2a$-$Parameter$, and thus the lower the population of PKBa1, the slower but deeper Foxoa drops (as for $maxPKBa1$ in chapter 6.2.2). If we look at figure 6.10 we can see this behaviour for small values of $PKBDropTORC2aParameter$. But for big values the drop of Foxoa becomes smaller again. This is due to a limitation of the model: In order to compute the maximal drop of Foxoa I used

$$\begin{aligned} minFoxoaFlux &= minFoxoa * maxATP \\ &* (r27P * maxPKBa1 + r30P * maxPKBa2 \\ &+ r33P * maxPKBaa) \end{aligned} \tag{6.5}$$

to compute the induced deactivation flux of Foxoa. But since not all variants of PKB can be maximal at the same time this formula is a upper bound instead of a exact computation. Therefore the minimal population of Foxoa (defined by the parameter $foxoaDrop$) will never been reached. However, for small values of $PKBDropTORC2aParameter$ this approximation is quite good, only for big values the error causes a growing population of Foxoa on figure 6.10.

PKBa1DropTORC2aParameter

The parameter $PKBa1DropTORC2aParameter$ defines the drop of PKBa1 if we go from zero to maximal TORC2a for maximal insulin input. Thus $maxPKBa1$ describes the population of PKBa1 for no TORC2a but maximal insulin, and

$$maxPKBa1 * \frac{PKBa1DropTORC2aParameter}{PKBa1DropTORC2aParameter + 1} \tag{6.6}$$

describes the population of PKBa1 for maximal TORC2a and maximal insulin input. If we choose a bigger value for this parameter, the sensitivity of PKBa1 for TORC2a will be reduced.

The effects of this on the population of PKBa1 can be seen in figure 6.7c, the effects on Foxoa (for Lobe they are similar) in figure 6.11.

maxTORC2aParameter

The parameter $maxTORC2aParameter$ defines the maximum populations of TORC2a. Since a higher populations of TORC2a increases the populations of PKBa2 and PKBaa and drop the populations of PKBa1, the influence of $maxTORC2aParameter$ is similar to the influence of $maxPKBa2$ in chapter 6.2.2. Thus a higher value of $maxTORC2aParameter$ will result in slower but stronger drop of Lobea and Foxoa as can be seen in figure 6.12.

6.2.3. The Deactivation Flux Weights

In order to quantify the influence of an active variant of PKB on one of the species Foxo, Lobe and TSC I will define the deactivation weights $a1Lobe$, $a2Lobe$, $aaLobe$, $a1Foxo$, $a2Foxo$, $aaFoxo$, $a1TSC$, $a2TSC$ and $aaTSC$.

Those weights might hide the influence of the parameters described in chapter 6.2.2

a1Foxo, a2Foxo, aaFoxo

As can be seen in figure 6.13 a small value of $a1Lobe$ causes a fast but not that deep drop of Foxoa. This is caused by the fact that PKBa1 is mostly present if the insulin input is small and therefore the populations of TORC2a is small (similar to $maxPKBa1$ in chapter 6.2.2). This behaviour can be seen in figure 6.13.

6.3. Parameters that Influence the Input-Subsystem

The parameters of the input-subsystem have no influence at the steady state with no or maximal insulin input, since then the populations of PIP3 will simply be to 0 or $maxPIP3$, but they influences how fast this steady state will be reached and what happens for intermediate insulin inflows.

6.3.1. Parameters that Influence the Effect of the Input-Subsystem on intermediate Insulin Inputs

We know that PIP3 grows with a growing insulin input, and saturates at $maxPIP3$ for high inputs. But how fast? This would define the insulin range that has the maximal influence on the system. The sensitivity of PIP3 on insulin is defined by three parameters

InR2Ratio

The parameters $InR2Ratio$ defines the ratio between InR2 and InR2a for an insulin input of 100 (this value is chosen arbitrarily, we could also define this ratio for any other nonzero insulin input). This parameter defines how sensitive InR2a and therefore the whole system reacts on a change of insulin inflow as can be seen in figure 6.14.

chicoDropParameter

The parameters $chicoDropParameter$ defines how much Chico drops for a maximal insulin input. The minimal populations of Chico can be computed as

$$minimalChico = maximalChico * \frac{chicoDropParameter}{chicoDropParameter + 1} \tag{6.7}$$

If the drop is small then Chicoa and therefore PIP3 reacts more sensitive on a change in insulin input, as can be seen in figure 6.15.

InRXRatio

The parameter $InRXRatio$ describes the ratio between InR1 and InR2 if the insulin input is 100. This parameters steers how sensitive InR1 and therefore InR2a reacts on a change of the insulin inflow. This has an influence on the sensitivity on PIP3 as can be seen in figure 6.16

6.3.2. Parameters that Influences the Speed of the Input-Subsystem

Beside the sensitivity of the steady state of PIP3 the input-subsystem also defines how fast this steady state will be reached. This is done by several parameters. This influence can be seen in figure 6.17.

6.4. Parameter that Influences the Rheba-Cycle

As states in chapter 3.2.3 Rheba has only one steady populations. But during a transition between two steady states the populations of Rheba changes. The shape of this raise is given by the parameter $lengthrhebaraise$. Its influence can be seen in figure 6.18

6.5. Parameter that Influences the Maximum of 4EBP

Most of the species have their maximum if insulin is either minimal or maximal. There are two exceptions: PKBaa and 4EBP.

There are two species that influences the populations of 4EBP: 4EBPa and TORC1a. The populations of 4EBP drops if one of them drops and increases if one of them increases.

If we add insulin to the system we see a drop in 4EBPa and an increase in TORC1a. We know from figure 1.1 that the population of 4EBP grows for high insulin inputs, thus for high insulin inputs the influence of TORC1a is stronger.

But for some insulin inputs Foxoa might get deactivated faster than Lobea, and therefore TORC1a will increase slower than 4EBPa decreases and the concentration of 4EBP will drop for a growing insulin input.

In figure 6.19 you can see what happens with the 4EBP level for different values for $a1Lobe/a1foxo$.

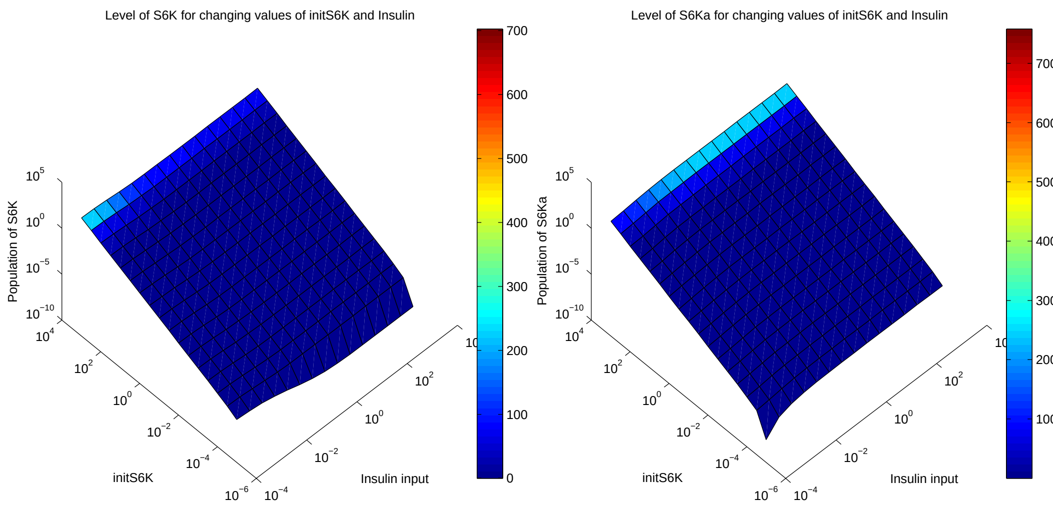


Figure 6.2.: The influence of the parameters $initS6K$ on the populations of S6K and S6Ka

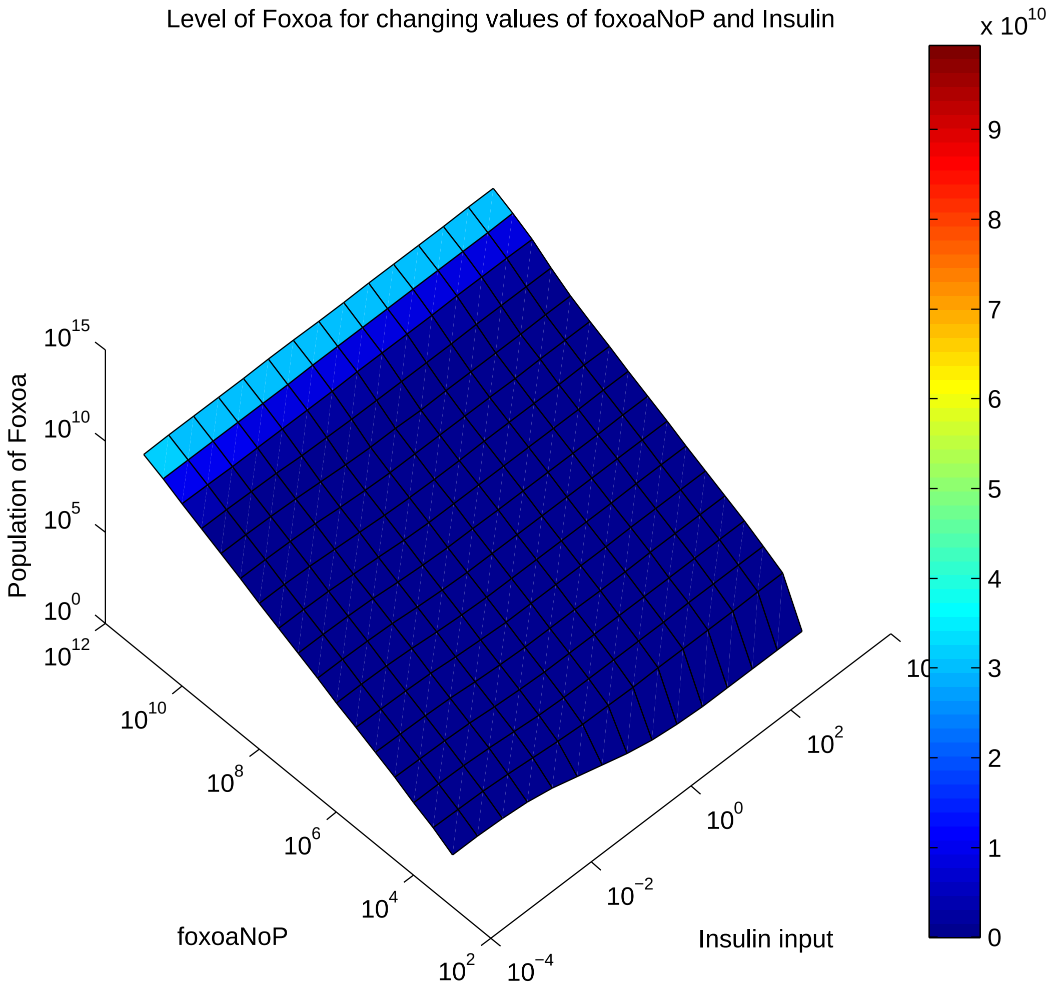


Figure 6.3.: The influence of the parameter $foxoNoP$ on the population of Foxoa

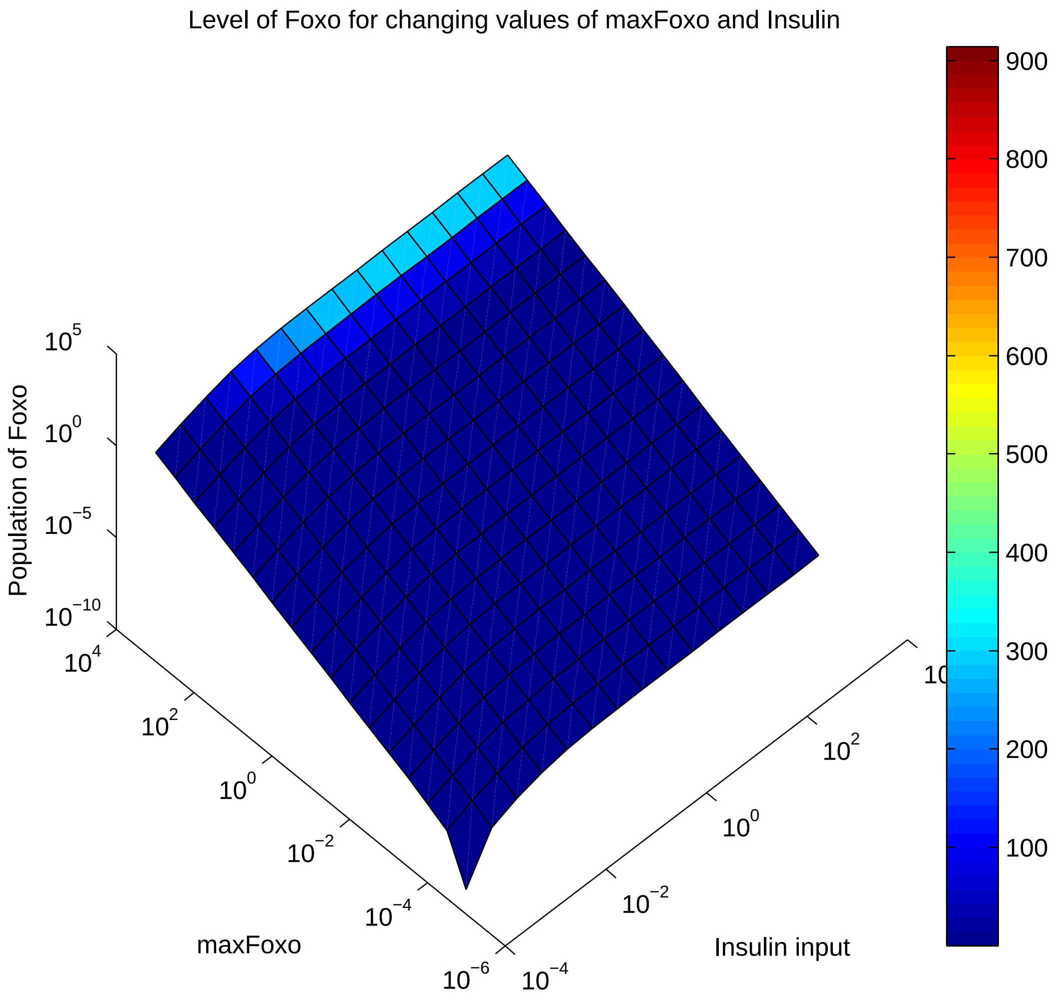


Figure 6.4.: The influence of the parameter $maxFoxo$ on the population of Foxo

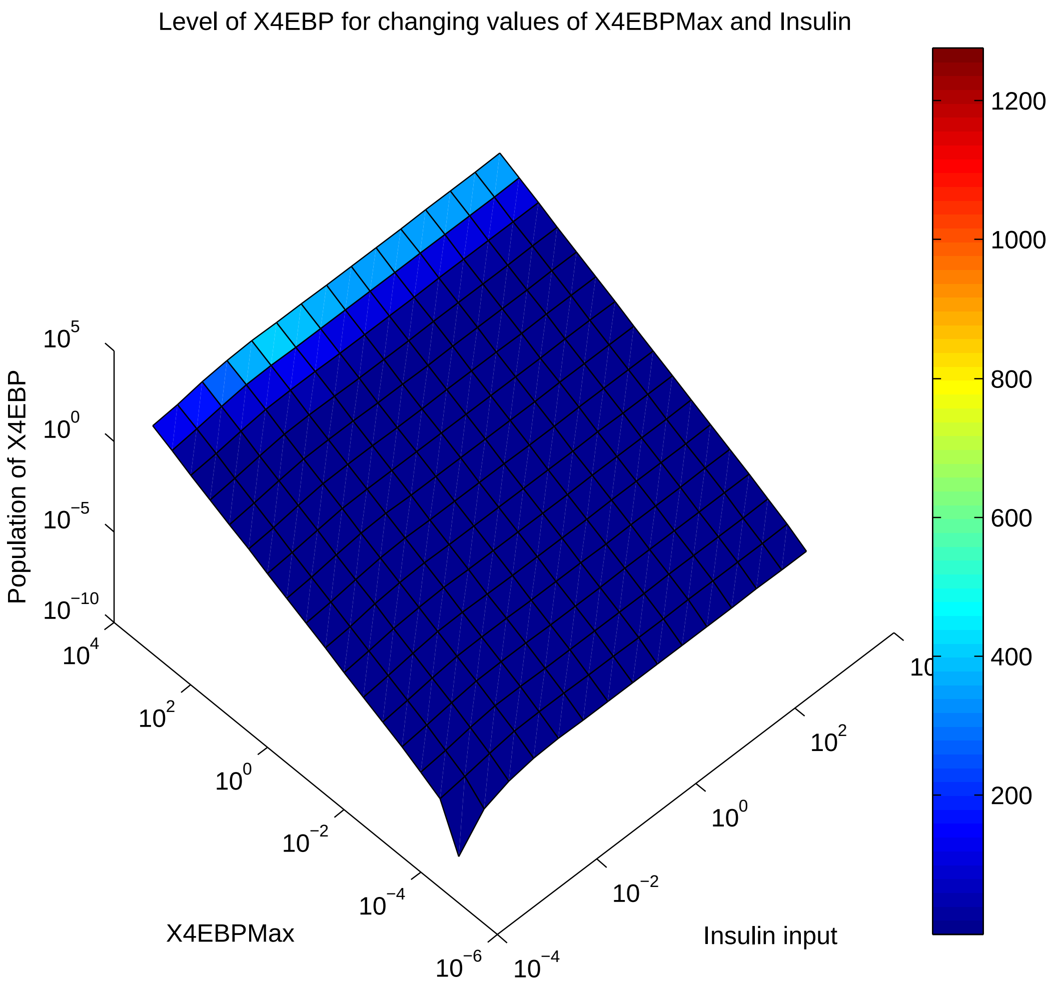


Figure 6.5.: The influence of the parameter $X4EBPMax$ on the population of 4EBP

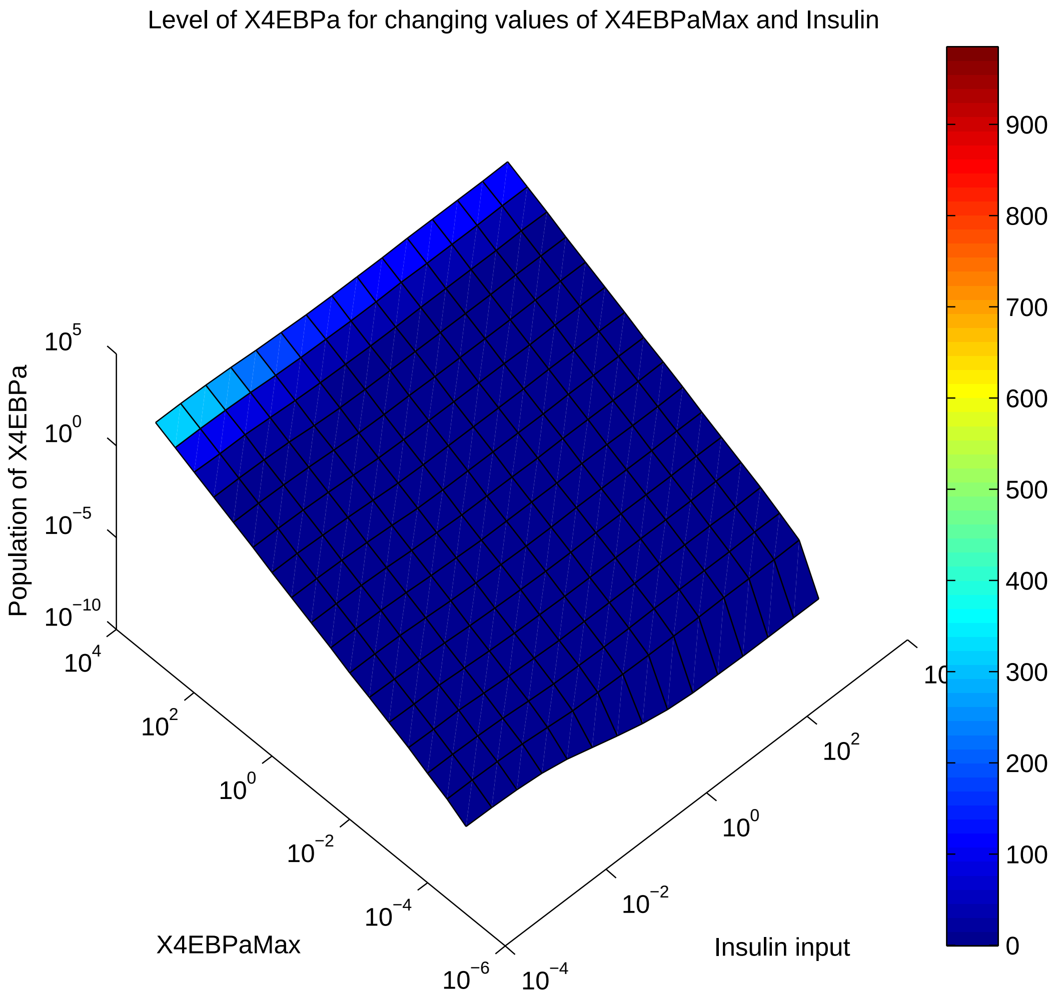


Figure 6.6.: The influence of the parameter $X4EBPaMax$ on the population of 4EBP

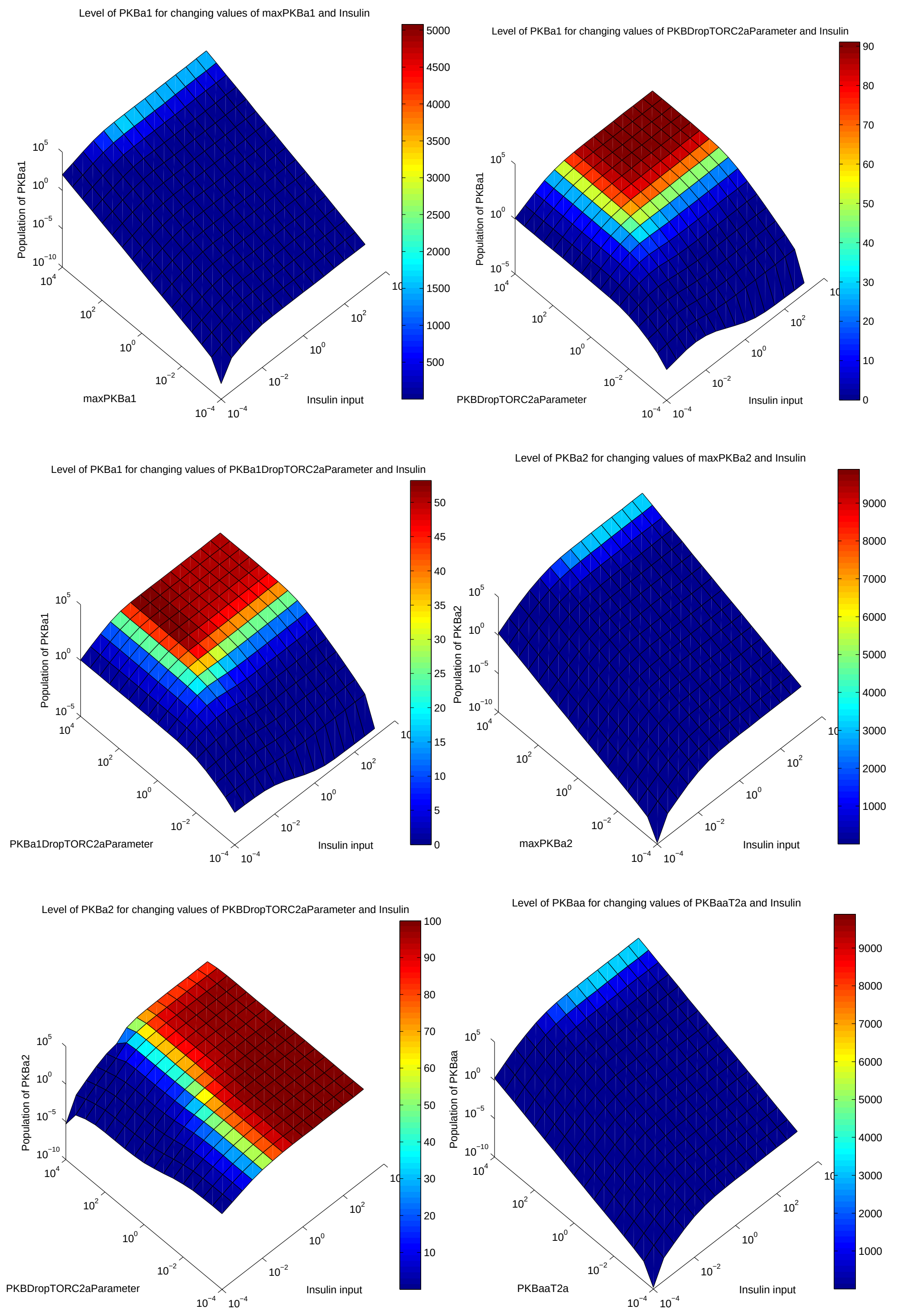
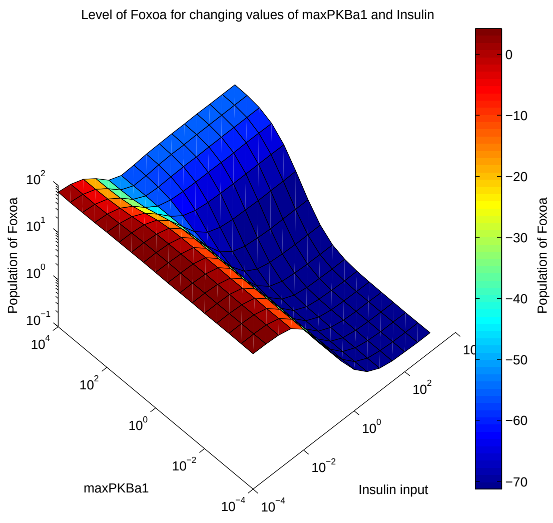


Figure 6.7.: The influence of different parameters on the PKB variants



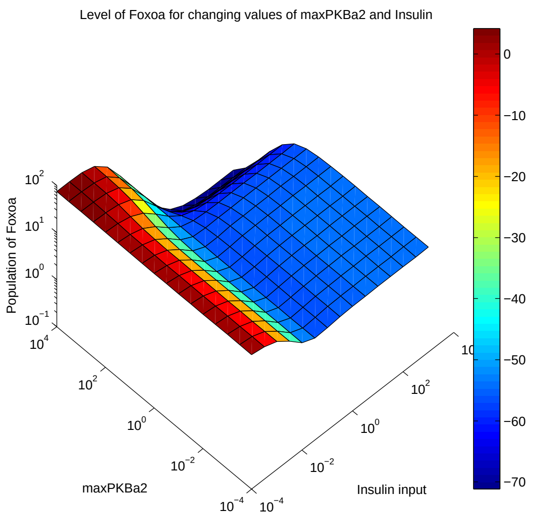


Figure 6.8.: The influence of $maxPKBa1$ and $maxPKBa2$ on Foxoa

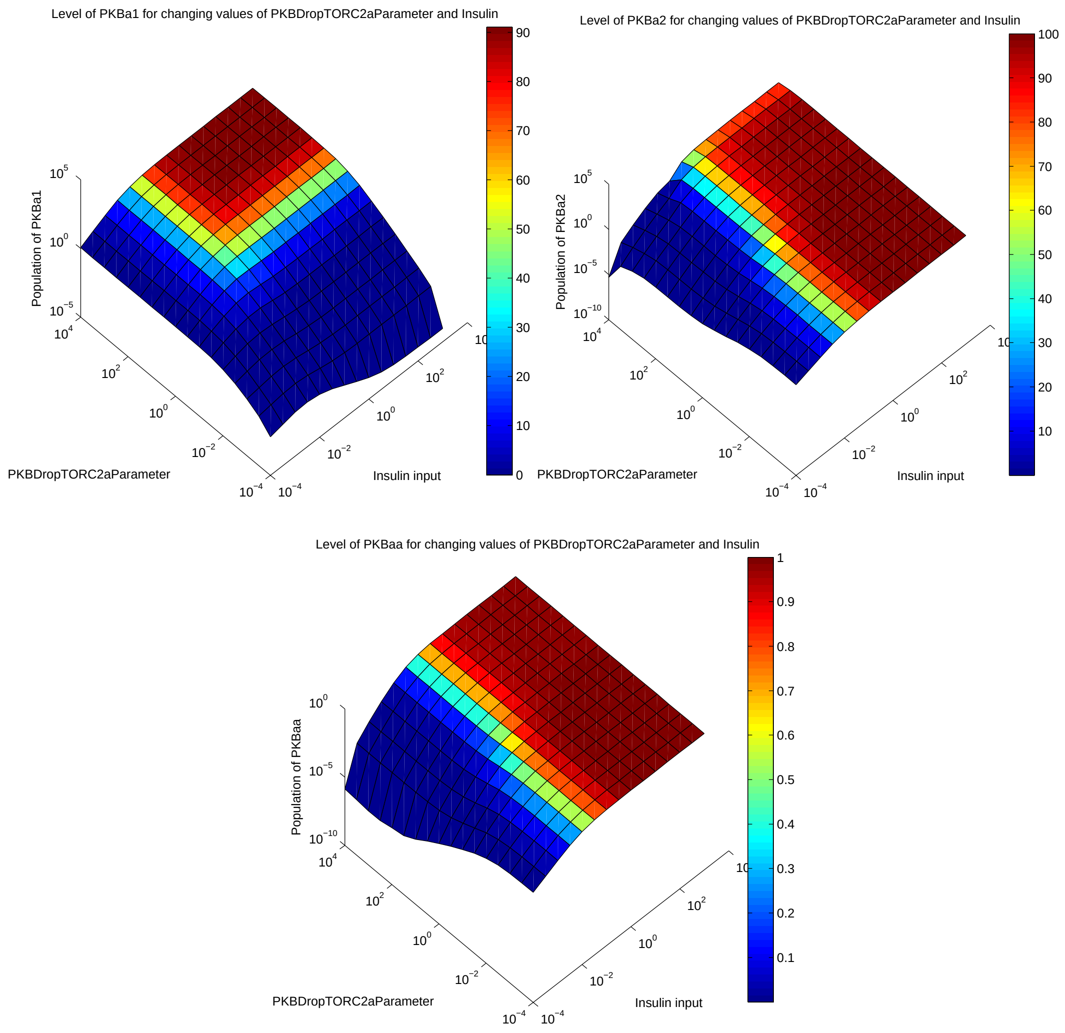


Figure 6.9.: The influence of the parameter $maxTORC2aParameter$ on the population of Foxoa

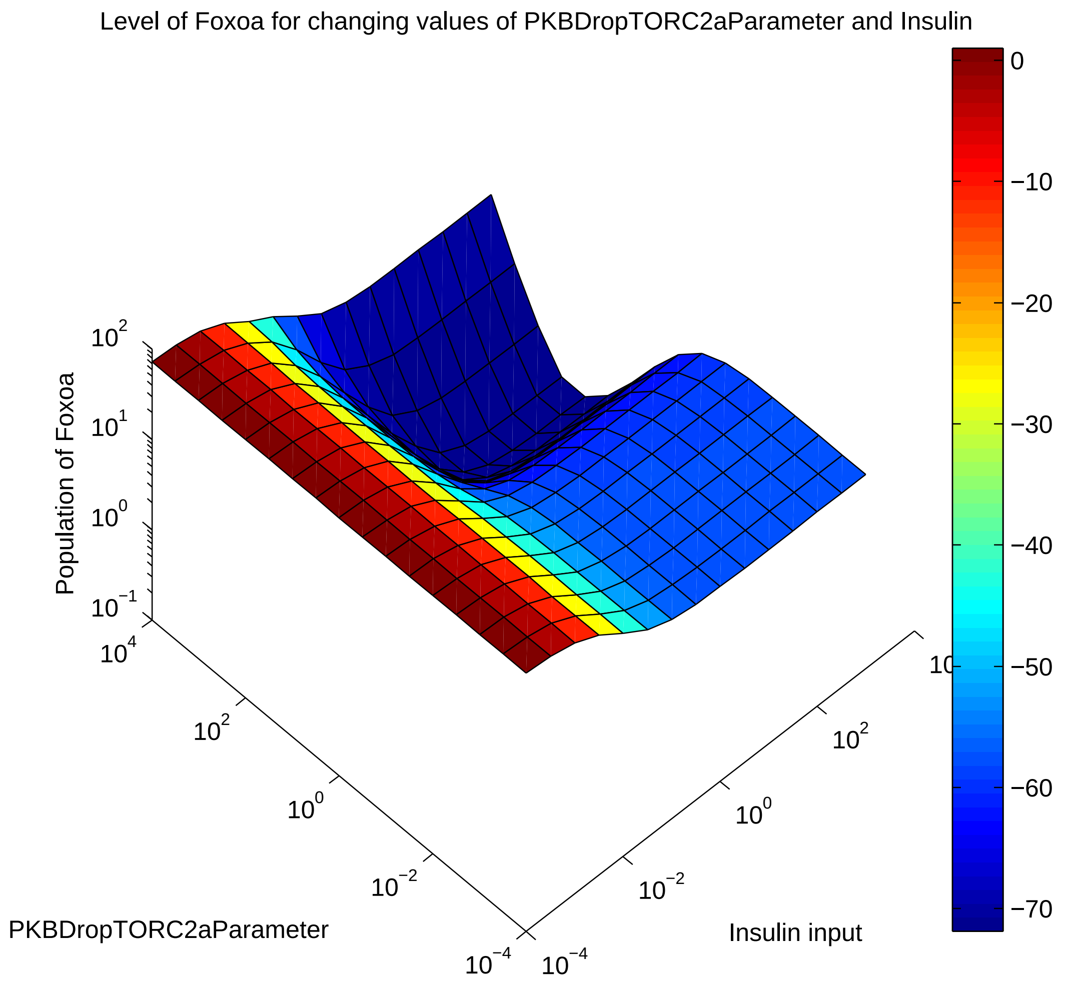


Figure 6.10.: The influence of the parameter $PKBDropTORC2aParameter1$ on the concentration of Foxoa

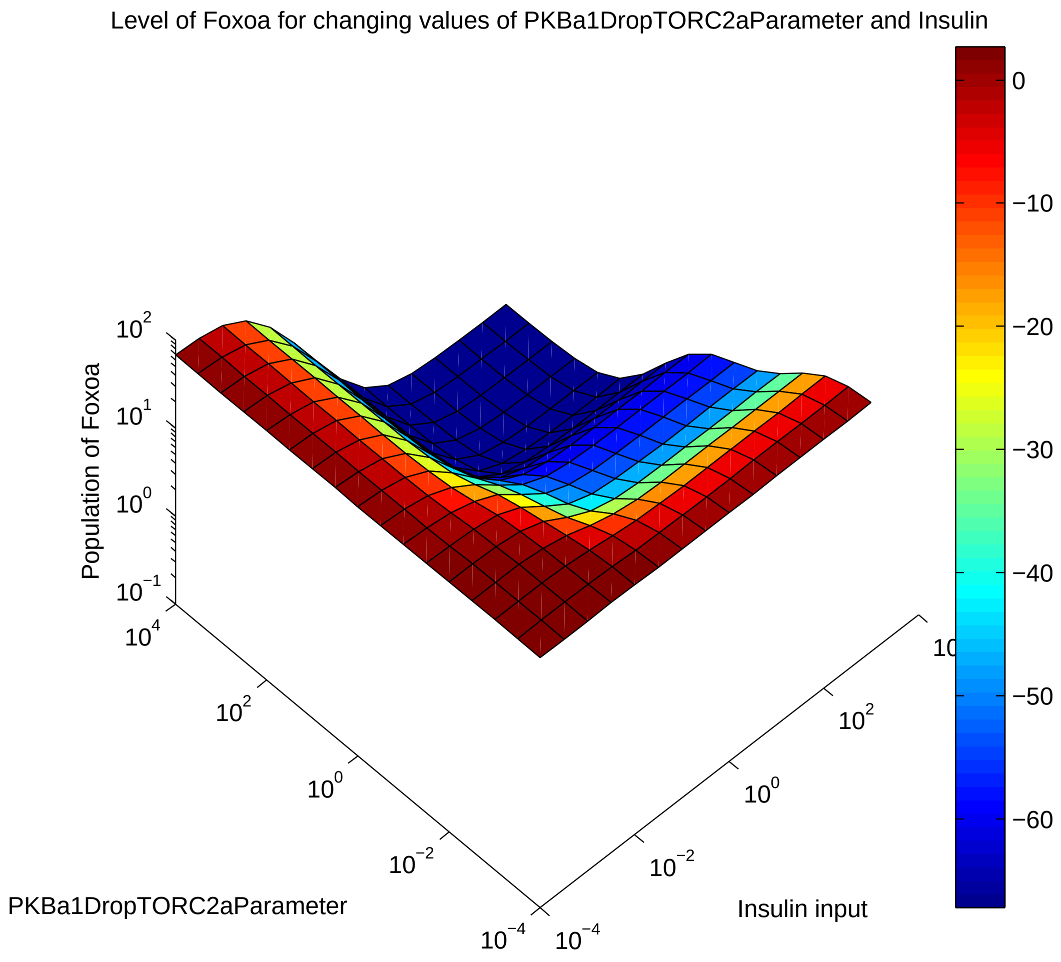


Figure 6.11.: The influence of the parameter $PKBa1DropTORC2aParameter$ on the population of Foxoa.

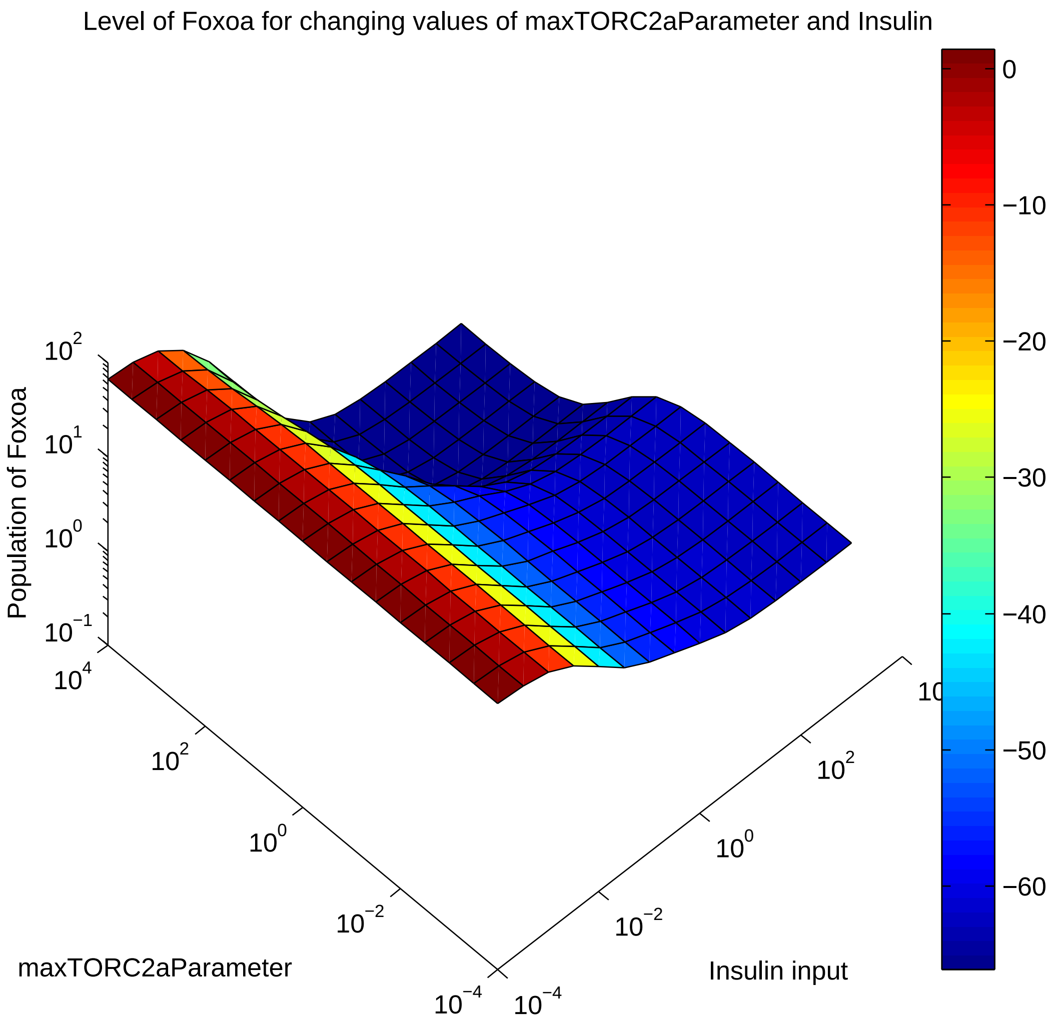


Figure 6.12.: The influence of the parameter *maxTORC2aParameter* on the populations of Foxoa.

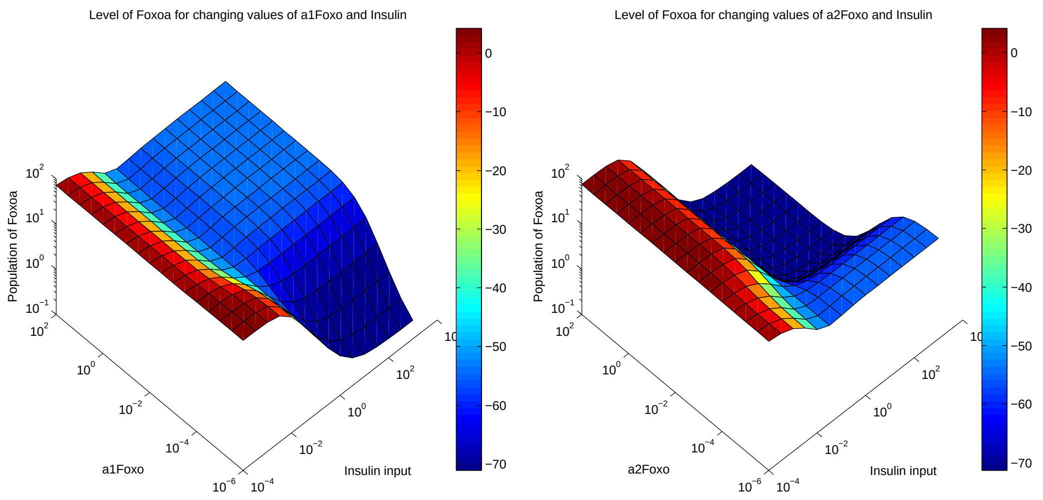


Figure 6.13.: The influence of the parameters $a1Foxo$ and $a2Foxo$ on the populations of Foxoa.

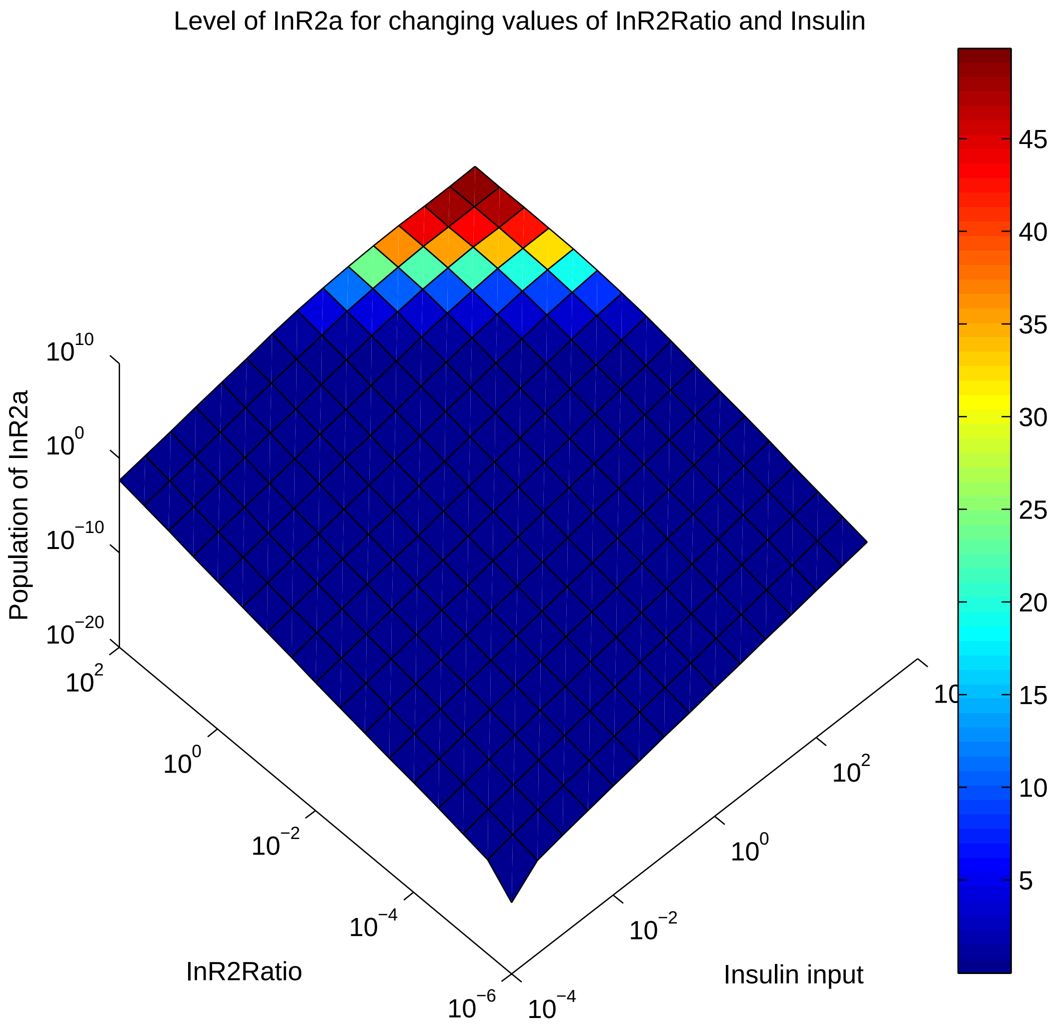


Figure 6.14.: The influence of the parameter *InR2Ratio* on the populations of InR2

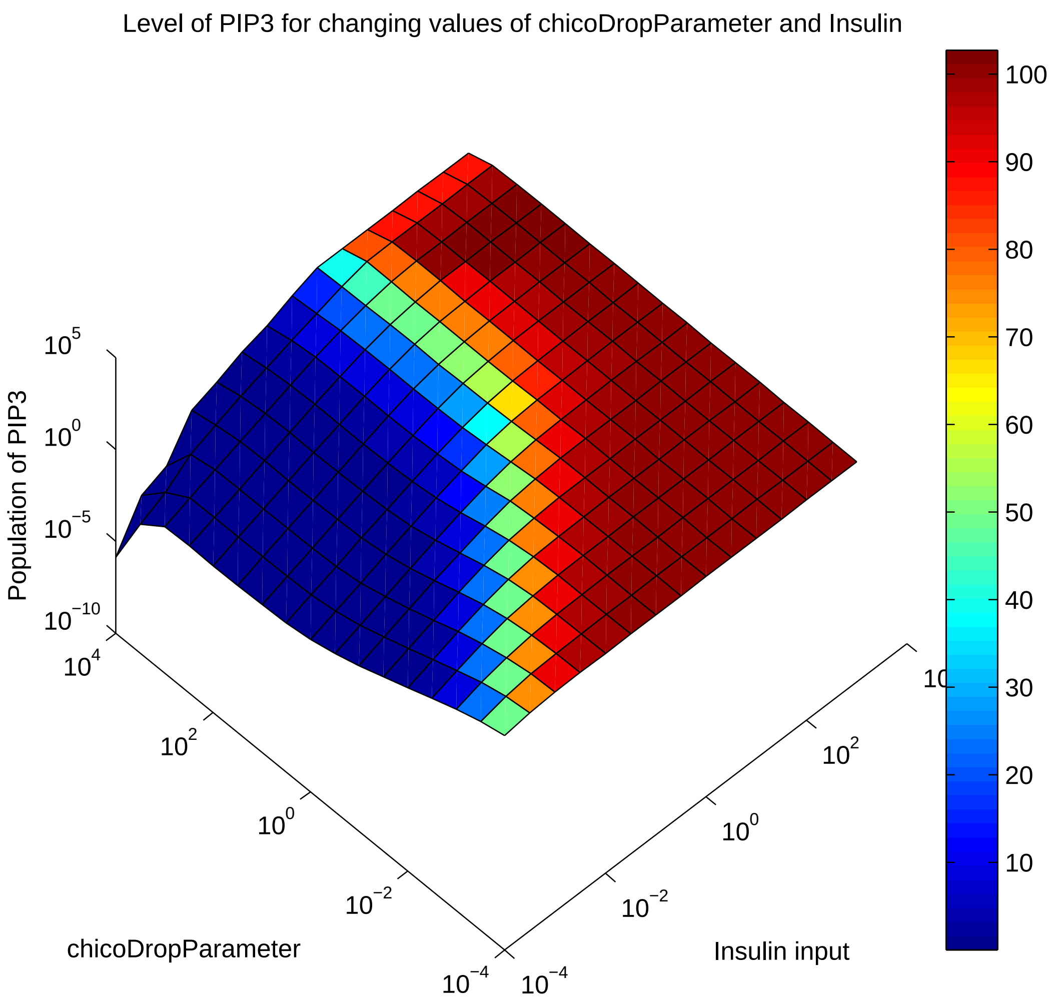


Figure 6.15.: The influence of the parameter *chicoDropParameter* on the populations of PIP3

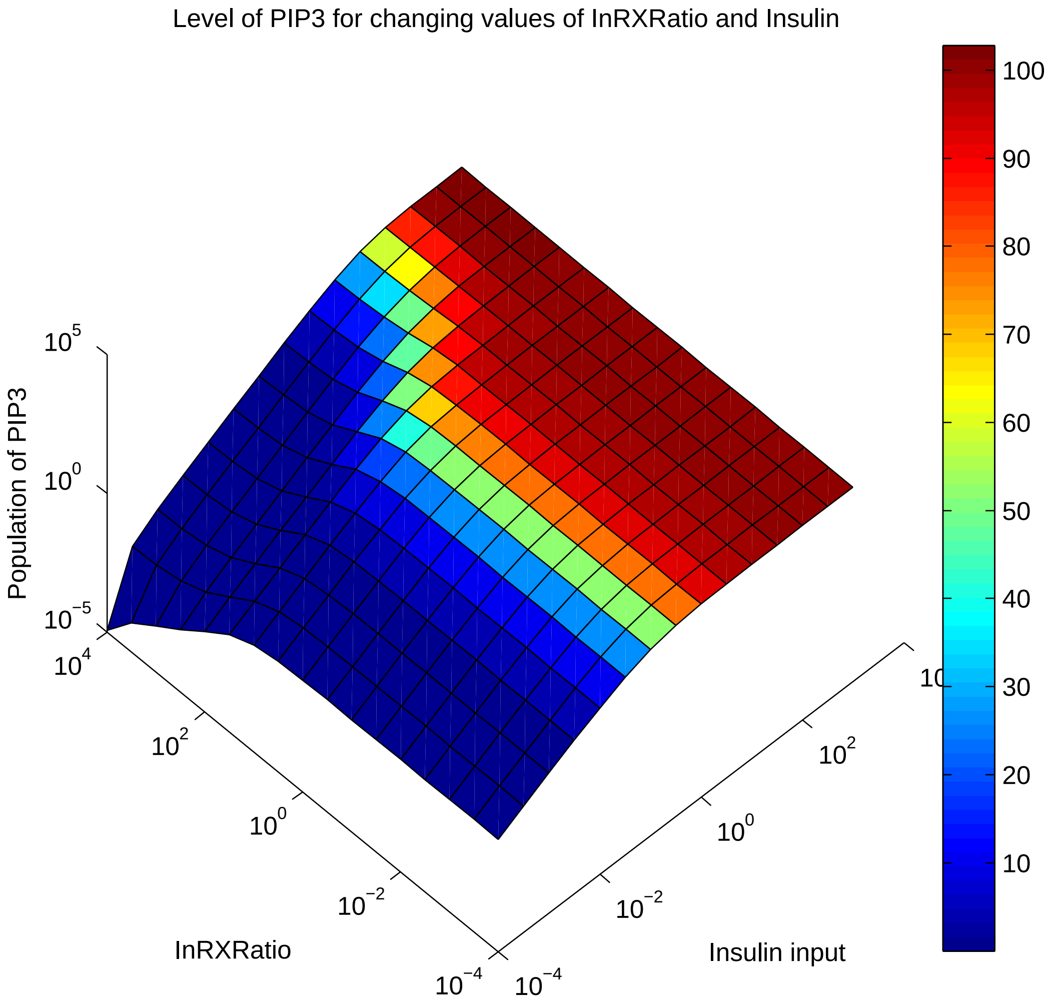


Figure 6.16.: The influence of the parameter $InRXRatio$ on the populations of PIP3

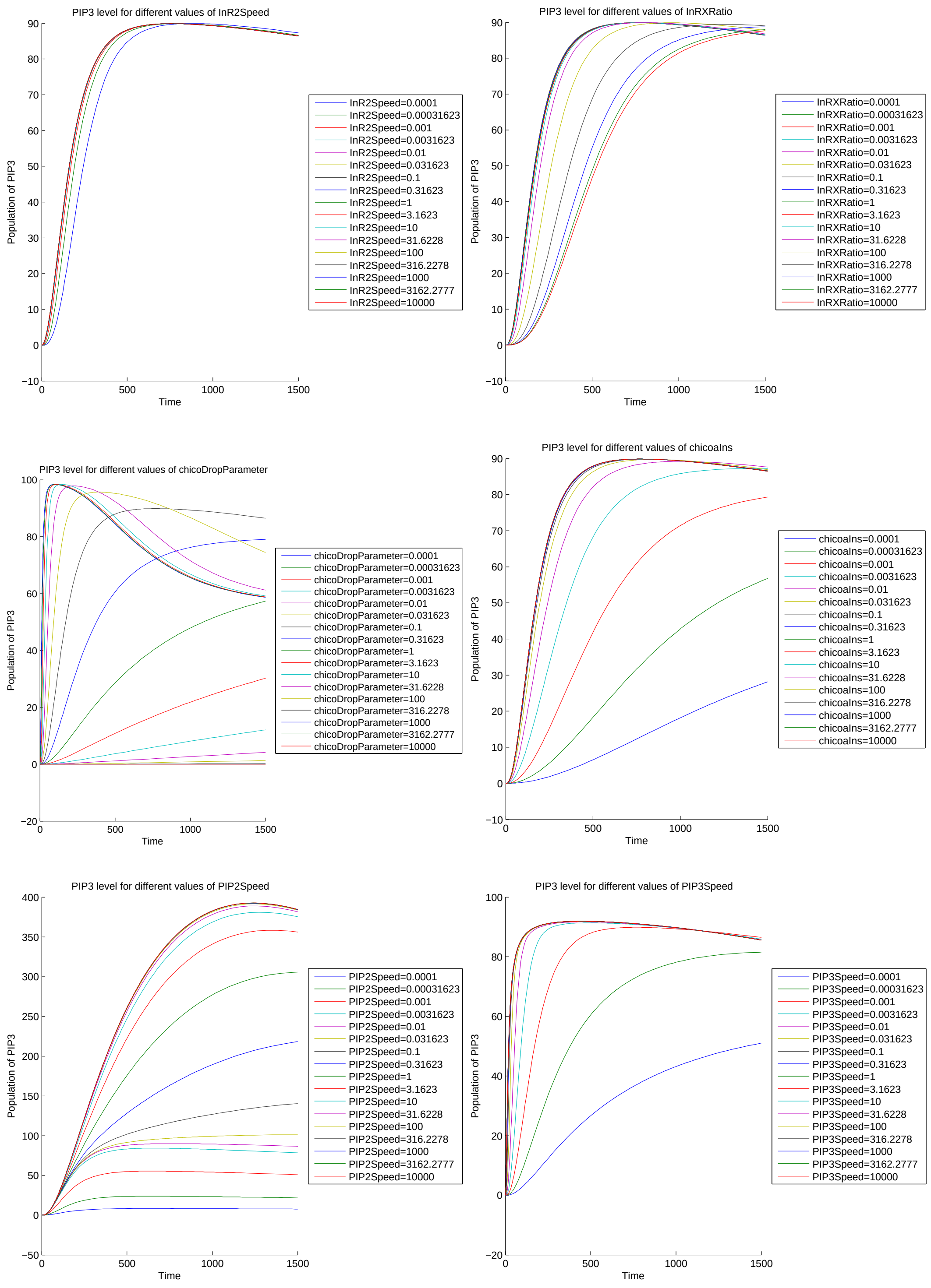


Figure 6.17.: Influence of different parameters on the speed of the input-subsystem

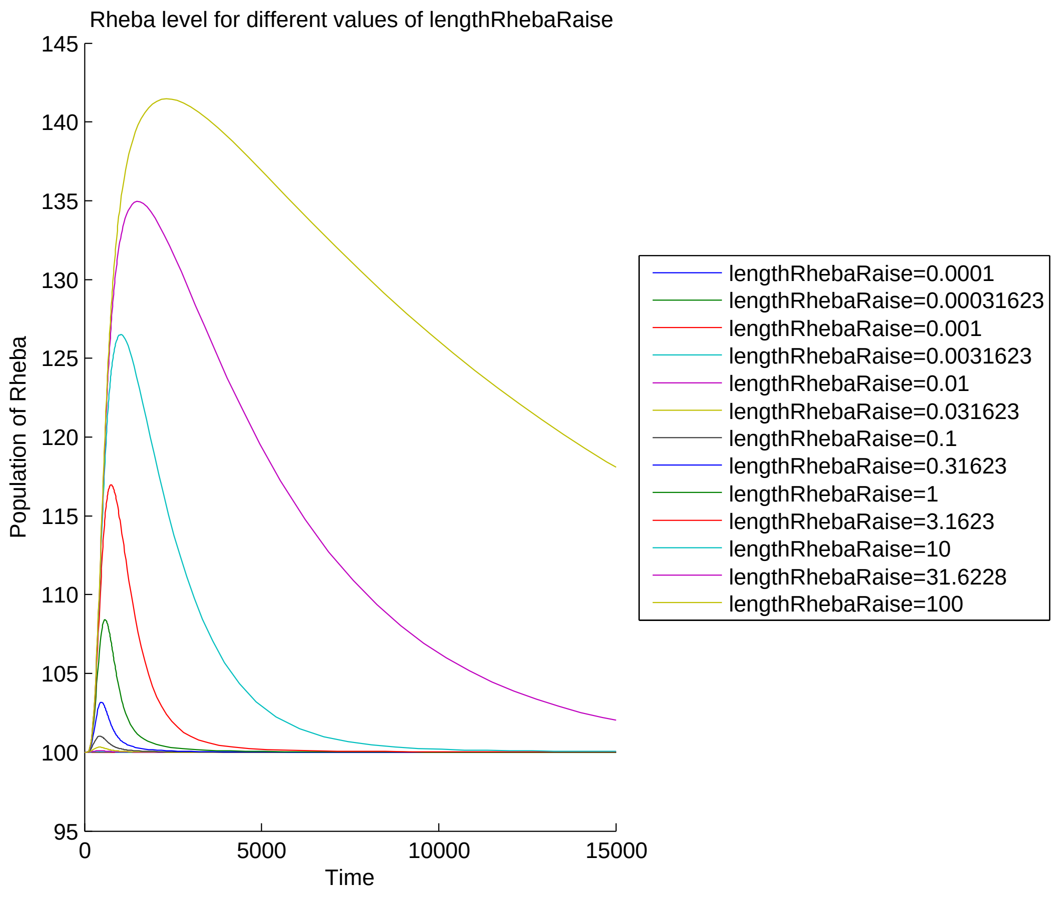


Figure 6.18.: The populations of Rheba for different parameter choices

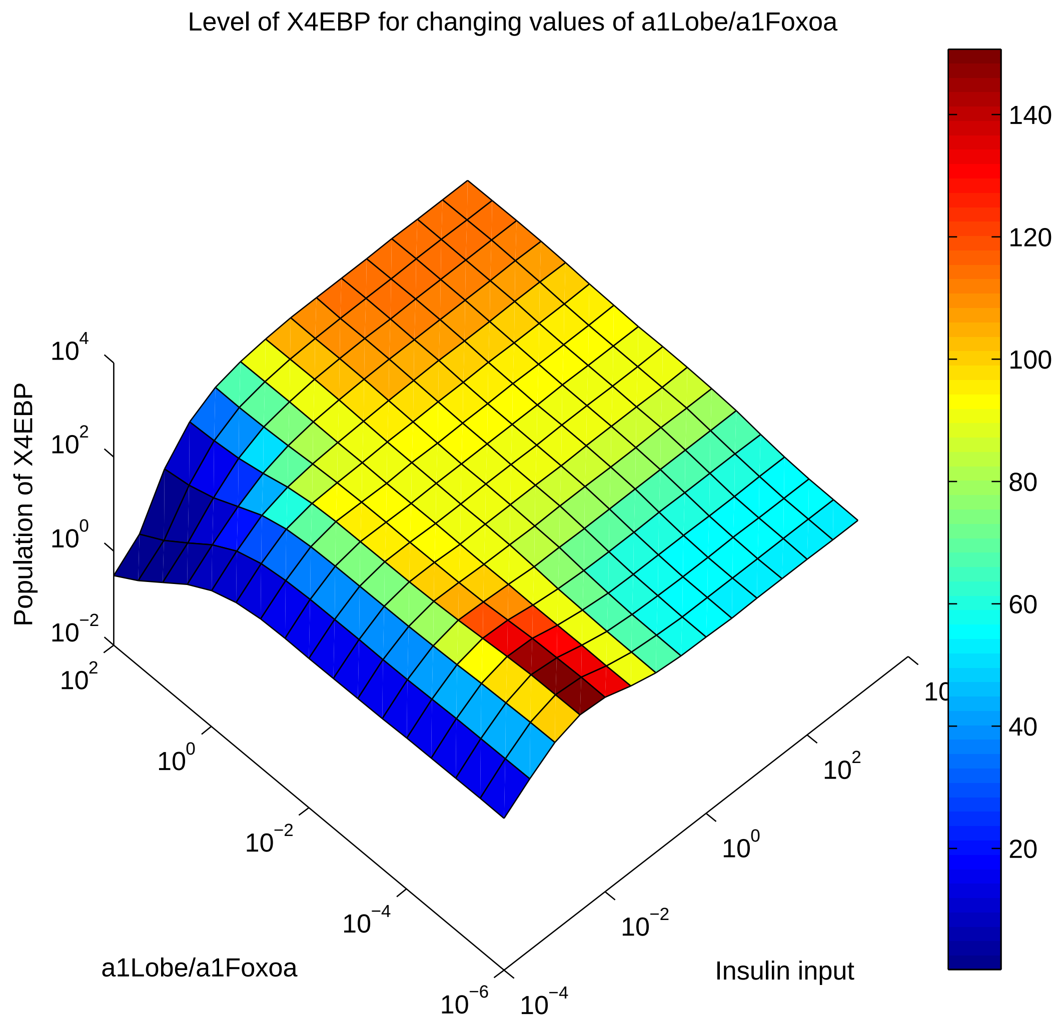


Figure 6.19.: The behaviour of the steady populations of 4EBP for different sensitivities of Foxoa and Lobea to a changing insulin inflow.

7. Simulation Using the Chemical Master Equation

7.1. The Chemical Master Equation

We can define the state of a chemical system as the populations $\vec{n} = [n_1, \dots, n_S]$ of S different species in the system. From the theory of chemical processes we know that *this state cannot evolve (...) with time as a purely deterministic process*[19]. Therefore the populations of the species for a given time must be viewed as random variable $n_{i,t}$ (the number of molecules of species i at time t).

The probability of a state $\vec{n}$ at time t given the initial state $\vec{n}_0$ at time t_0 can then be expressed as $P(\vec{n}, t|\vec{n}_0, t_0)$.

This probability distribution evolve over time. We can define the derivative as[19]

$$\frac{\partial}{\partial t} P(\vec{n}, t|\vec{n}_0, t_0) = \sum_{\mu=1}^{M} [c_\mu h_\mu(\vec{n} - \nu_\mu) P(\vec{n} - \nu_\mu, t|\vec{n}_0, t_0) - c_\mu h_\mu(\vec{n}) P(\vec{n}, t|\vec{n}_0, t_0)]$$

Whereas $\nu_{\mu,i}$ is the change of the population of species i when the reaction μ occurs (stoichiometry), c_μ is the probability that a random selected combination of the reactants of reactions μ reacts (per unit time) and $h_\mu(\vec{n})$ is the number of distinct combinations of the reactants of reaction μ.

This equation is also called the chemical master equation, and it is the exact way of describing a chemical process. The ODE model in chapter 5 ignores the non-deterministic and discrete (the population of each species must be a natural number) nature of the process.

7.2. Short Overview over Different Simulation Algorithms

7.2.1. Exact Algorithms

Solving the chemical master equation analytically is usually not possible.[19] So in order to find an estimate for the expected value of the random variables $N_{i,t}$ we could simulate the system multiple times and calculate the mean.

The simulation of this process in normally been done with a variant of the Stochastic Simulation Algorithm (SSA, also called Gillespie or Bortz-Kalos-Lebowitz Algorithm, described in [11]).

The main idea behind those algorithms is that we compute the *propensities* $a_\mu = c_\mu * h_\mu(\vec{n})$ for the current state $\vec{n}$. The propensity is the probability that the reaction μ happens per unit time and then compute which reaction happens when. The exact algorithm can be found in figure 7.1.

Since SSA is computationally expensive (for M reactions and N species you have to update $M + N$ values and compute 2 random variables per step, and the step size might be very small) there are a number of modified version that increases the computation speed like

1. at t=0 initialize $\vec{n} := \vec{n_0}$, $a_\mu := c_\mu * h_\mu(\vec{n}), a = \sum_\mu a_\mu$.
2. Find the reaction happening next: Sample a random number r_1 uniformed distributed between 0 and 1 and then find the smallest number $\hat{\mu}$ that satisfies
$$r_1 < \sum_{\mu=1}^{\hat{mu}} \frac{a_\mu}{a}$$
3. Find the time when this reactions happens: Sample a random number r_2 uniformed distributed between 0 and 1 and then compute
$$\tau = -a^{-1} ln(r_2)$$
4. Update $\vec{n} = \vec{n} + \vec{\nu_\mu}$, $t = t - \tau$, recompute all a_μ and a and continue at step 2 until you reach the end of the desired time frame.

Figure 7.1.: The SSA algorithm

the first reaction method[10], Gilson-Brucks next-reaction method[8] or the Partial Density Method[16]

7.2.2. Approximation Algorithms

Beside the exact, there are a number of approximation methods.

The τ-Leap method[9] uses a fixed time step size τ and then estimates how many of each reaction takes place in this step. Thus instead of recomputing the propensities after each reaction recomputed only after a given time frame. The approximations is good as long as the populations are big compared to the number of reactions that happen during one step. The drawback of this method is that we have to compute a random variable for every species in each time step, in order to compute how many reactions happened.

Another well known approximation method is the r-Leap Method[1]. There we do not fix the time, but the number of reaction that has to happen, before the propensities will be recomputed. Again one random variable per species will be needed to compute how many of a specific reaction happed.

7.3. Simulate the System using StochPy

StochPy[14] (Stochastic modelling in Python) is a python package that provides several stochastic simulation algorithms. I used it to simulate this system.

As claimed in chapter 3.1.2 the populations of the species PTEN, PDK1 and GEF are constant. Furthermore I assume the population of ATP to be constant (as claimed in chapter 3.1.4). I will also use the population of insulin instead of the insulin input flux as input for the system, so the population of insulin can also been assumed to be constant.

Those constant species and their in-/output reaction then do not have to be simulated and this increases the speed of the SSA simulation. I also removed the species Lobe, TSC and ADP, since they have no further use in the system.

7.4. Simulation Results

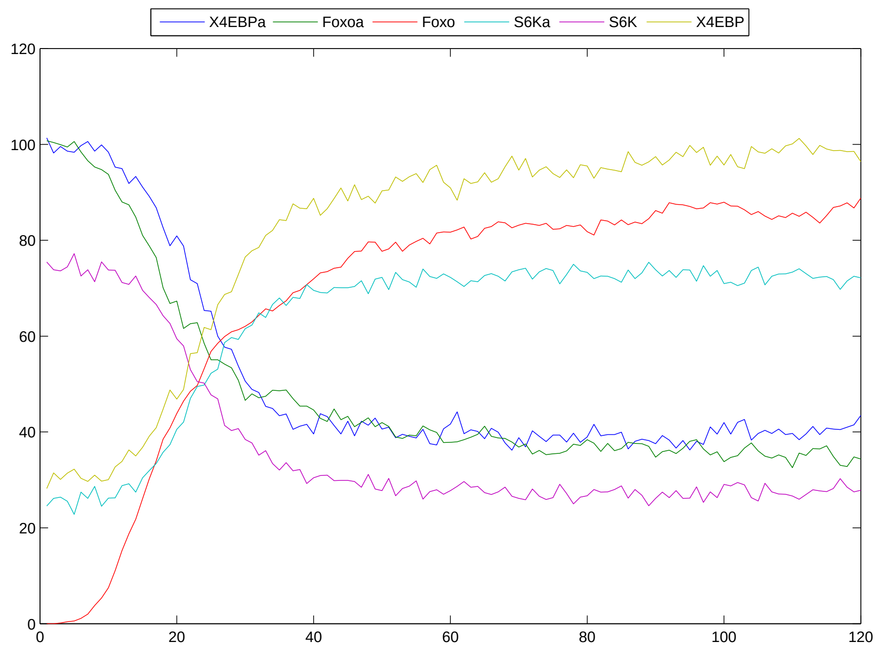


Figure 7.2.: The simulation result using SSA, using tau-leaping with $\epsilon = 0.01$ and 20 trajectories

The behaviour of the system can be seen in figure 7.2. As expected the system shows the same overall behaviour as the ODE simulation.

7.5. Limitation of the SSA-Model

The SSA model has three limitations

- As described in chapter 4.1 I use normal kinetics, even if for some reactions Michaelis-Menten Kinetics would be the more accurate choice.
- I use a fixed population of ATP (see chapter 5.3) and insulin even though those species are only nearly constant.
- The real scales are unknown, but since in a SSA simulation the population of each species have to be an integer value the scale matters.

8. Conclusions

8.1. Summary

In this thesis I presented a model of the insulin signaling pathway. I showed that this model is consistent with the preexisting knowledge and provides useful results. Then I simulated it using ODEs and the stochastic simulation algorithm for various parameter sets and showed the influence of those parameters for some important output species. Those effects are summarized in figure 8.2. In this chapter I come up with some guidelines to optimize the parameter choice under certain assumptions.

8.2. Limitations of the Model

The model described in this thesis has the following limitations:

- The model and the simulation techniques assumes that all the reaction take part in one homogeneous and well stirred reactor with constant conditions. Those assumptions might not be fulfilled.

- The transformation on the models done in chapter 2 in order to enforce mass conservation seems likely, but have to be verified.

- The kinetics are only approximative. As described in chapter 4.1 I used basic kinetics even where Michaelis-Menten kinetics would be suited better. This approximation allows it to describe the system with less free variables. In chapter 3.3.1 I show a minimized model for the system using more accurate kinetics and as can be seen there this model shows a similar basic behaviour as the basic kinetics approximation.

- All parameters have to be in a suitable range. I assumed all reaction rates to be in a range defined in [4].

8.3. System Design

Under the assumption that a biological system is optimal, we can optimize the parameters for some given data.

8.3.1. Insulin Input

First we have to decide on which insulin input range the system should be sensitive to. This is defined by the parameter *chicoDropParameter*. The influence of this parameter can be seen in figure 6.15. The higher we choose it, the more sensitive the system reacts on insulin input.

We can approximate the population of InR2a (see also chapter 3.3.1) as

$$PKBI = \frac{Ins * initInRX/2}{\frac{100}{InR2Ratio} + Ins}$$

and therefore if we want the system to show half of the maximal reaction on an insulin input of I_{half} we need to choose $InR2Ratio$ as

$$InR2Ratio = \frac{100}{I_{half}}$$

8.3.2. Smoothing out Fast Insulin Inputs

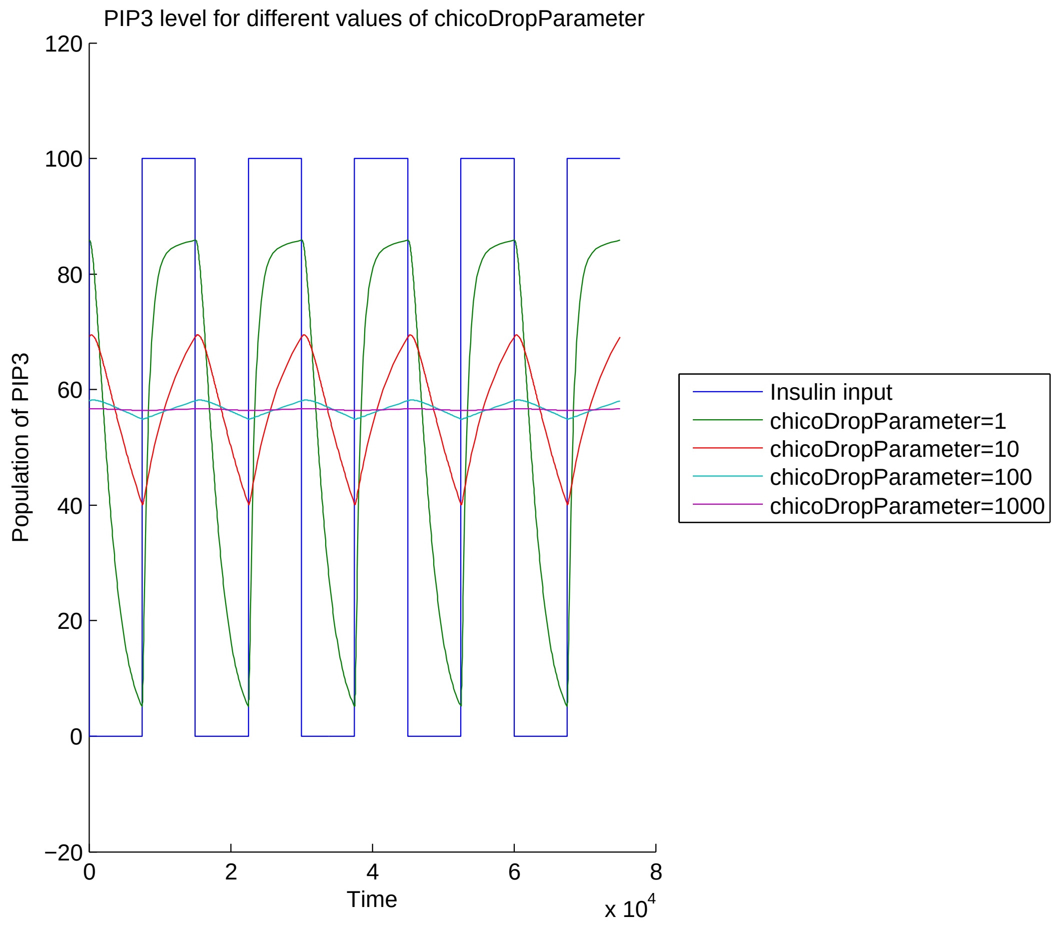


Figure 8.1.: The population of PIP3 for oscillating insulin input, smoothed with different values for parameter $chicoDropParameter$

We might want to have a certain amount of smoothing in the system. The smoothing capability of the input-subsystem can be steered with the parameter $chicoDropParameter$. As can be seen in figure 8.1 a high value for this parameter makes the system react slower and therefore smoother.

Besides *chicoDropParameter* there are also other parameter influencing this effect (like $PIP2Speed$, $PIP3Speed$ and *chicoaIns*), but their influence will be hidden by *chicoDropParameter*.

8.3.3. The PKB-Subsystem

Then we have to define whether we want to have a fast or a strong effect on the output species. This can be steered with the parameters $maxPKBa1$ (a high value results in a fast, but not that strong effect) and $maxPKBa2$ (reverse influence).

We can also steer this for the species Foxoa, Lobea and TSCa individually through the parameters $aXFoxo$, $aXLobe$ and $aXTSC$: A high value for the $a1$ parameter results in a fast effect, a higher value in $a2$ and aa results in a strong effect.

8.3.4. The S6K-Cycle

Then we have to define the total scale of S6K. As we know the total molecular population of S6K and S6Ka is constant. The total population is given in parameter $initS6K$. Thus with this parameter we can define the scale of S6K and S6Ka.

Beside the scale we have to define the behavior of those two species. We know from figure 1.1 that if no insulin is present most of the total population is stored in S6Ka, and if insulin becomes present this will change. Thus we can define two ratios

$$ratioS6KNo := \frac{initS6K}{S6K_{NoInsulin}} ratioS6KIns := \frac{initS6K}{S6K_{MaxTORC1a}}$$

The 2nd definition comes from the fact that a rising insulin input will increase the TORC1a level. As we can easily verify both ratios have to be between 0 and 1. The first ratio is given by the parameter $ratioS6KNoparameter$. To ensure that the drop is below one I had to apply a range transformation (see chapter 4.2.5)

$$ratioS6KNo := \frac{ratioS6KNoparameter}{ratioS6KNoparameter + 1}$$

The 2nd ratio cannot be chosen freely, since the maximal drop of S6K is defined by the maximal raise of TORC1a. Now this raise is again bounded by the maximal drop of Lobea. So lets start there.

The parameter *LobeaDrop* defines the maximal drop of Lobea (remember Lobea has its maximum when no insulin is present). The minimal positive of Lobea is then

$$minimalLobea = \frac{LobeaDrop}{LobeaDrop + 1} * maximumLobea$$

From this and the fact that the reaction rate of reactions $r50$ and $r51$ needs to be positive the maximum raise of TORC1a is limited. We have

$$\begin{aligned} \frac{LobeaDrop}{LobeaDrop+1} * maximumTORC1a &< minimumTORC1a & (8.1)\\ maximumTORC1a &> minimumTORC1a & (8.2) \end{aligned}$$

To fulfill this I introduced the parameter $TORC1aRangeParameter$. This parameter defines where in this range $minimumTORC1a$ lies

$$\begin{aligned} minimumTORC1a &= \frac{TORC1aRangeParameter}{TORCRangeParameter+1} * maximumTORC1a \\ &+ \left(1 - \frac{TORCRangeParameter}{TORCRangeParameter+1}\right) \\ &* \frac{LobeaDrop}{LobeaDrop+1} * maximumTORC1a \end{aligned} \tag{8.3}$$

Now due to the fact that the reaction rate of the reactions $r54$ and $r54r$ are again non-negative,this defines the 2nd S6K range

$$\begin{aligned} ratioS6KIns &= ratioS6KNo * minimumTORC1a \\ &/ \ (maximumTORC1a - ratioS6KNo \\ &* \ (maximumTORC1a - minimumTORC1a)) \end{aligned} \tag{8.4}$$

or using the parameters defined in this chapter

$$\begin{aligned} ratioS6KIns &= ((TORCRangeParameter + 1) \\ &* \ (LobeaDrop + 1) - 1) \\ &/ \ ((ratioS6KNoparameter + 1) \\ &* \ (TORCRangeParameter + 1) \\ &* \ (LobeaDrop + 1) - ratioS6KNoparameter) \end{aligned} \tag{8.5}$$

The exact computation can be found in appendix A.1.

8.3.5. Foxo and Foxoa

We also have to define the scale of Foxo and Foxoa. This can be done with the parameters $maxFoxo$ and $FoxoaNoP$ (the amount of Foxoa if no insulin is present).

8.3.6. 4EBP and 4EBPa

As for Foxoa we can define the scale of 4EBP and 4EBPa with the parameters $X4EBPaMax$ and $X4EBPMax$.

Furthermore the grow behaviour of 4EBP can be adjusted by changing the ratio between the parameters $a1Lobea$ and $a1Foxoa$ as showed in figure 6.19.

8.4. Outlook

8.4.1. Model Refinement

There are still some uncertainties with the model used

- The model uses normal kinetics. Does the behaviour of the system change if we would assume Michaelis-Menten kinetics?
- Are the transformations made in chapter 2 correct?
- The model is scale invariant. But as seen in chapter 7 this might no longer be the case if SSA is used to simulate it. How does the scale influences the outcome of the stochastic simulation?

- I assumed a homogeneous and well stirred reactor in which all the reactions take place. What changes if we drop this assumption?
- I assumed the concentration of ATP to be constant over time (see chapter 5.3). But how does the behaviour of the model change if this assumption is wrong and the concentration drops significantly for growing insulin input?

8.4.2. Model Generation from Experimental Data

In chapter 8.3 I came up with some guideline how to choose the parameters in order to fit the model to experimental data. So this data could now be used in order to refine the model.

8.4.3. Insulin Induces Cell Growth

In this thesis I showed how an insulin input influences the concentrations of S6K, 4EBP and Foxo.

- How do those species regulate the cell grow, what other factors play a role in cell grow and how do they interact with the insulin signaling pathway?
- If a cell does not get enough nutrients the cell growth will be paused. Is this effect caused by the insulin signaling pathway and if yes how does a lack of nutrient influences this system?
- How do the insulin like grow factors IGF-I and IGF-II influence the cell growth. Do they use the same pathway and if yes who do they interact with it?
- How does the insulin reaction changes if the cell grows? As we know cells stop growth with a certain size, but how does the cell size influence the insulin signaling pathway?

Parameter	Explanation	Effect on steady state	Effect on transitions	Effect on scaling
$initS6K$	Total molecular number of S6K and S6Ka	none	none	Sum of [S6Ka] + [S6K]
$foxoNoP$	Foxoa level if no insulin is present	none	none	Maximal level of Foxoa
$maxFoxo$	Foxo level if PKBa1, PKBa2 and PKBaa are maximal	none	none	Maximal level of Foxo
$X4EBPaMax$	Level of 4EBPa if Lobea is maximal	none	none	Maximal level of 4EBPa
$X4EBPMax$	Level of 4EBP if TORC1a and 4EBPa are maximal	none	none	Maximal level of 4EBP
$PKBDropTORC2a$-$Parameter$	Drop of PKB if TORC2a is maximal compared to if TORC2a is zero (for maximal insulin input)	Low values increase the influence of PKBa1, higher values the influence of PKBa2. Furthermore very high values limits the drop of Foxoa and Lobea	none	none
$maxPKBa1$	Level of PKBa1 if insulin is maximal and TORC2a is zero	Increases the influence of PKBa1. A high influence generally causes a fast but not strong reaction on all output species.	none	Maximal level of PKBa1
$maxPKBa2$	Level of PKBa2 if insulin and TORC2a are maximal	Increases the influence of PKBa2. A high influence generally causes a strong but not fast reaction of all output species.	none	Maximal level of PKBa2
$PKBaaT2a$	Level of PKBaa if insulin and TORC2a are maximal	Increases the influence of PKBaa. A high influence generally causes a strong but not fast reaction of all output species.	none	Maximal level of PKBaa
$maxTORC2aParameter$	Level of TORC2a if InR2a is maximal and S6Ka is minimal	Increases the influence of PKBa2 and PKBaa. A high influence generally causes a strong but not fast reaction of all output species.	none	Maximal level of TORC2a
$PKBa1DropTORC2a$-$Parameter$	Drop of PKBa1 if TORC2a is maximal compared to if TORC2a is zero (for maximal insulin input)	Increases the influence of PKBa2 and PKBaa.	none	none
$aXLobe$, $aXFoxo$, $aXTSC$	The reaction rates for the reactions $r27$, $r30$ and $r33$	The influence of the active variants of PKB on Lobea, Foxo and TSC. The difference $a1Lobea/a1Foxo$ influences the grow behaviour of $X4EBPa$	none	none
$InR2aRatio$	The ratio between the populations InR2a / InR2 for a insulin level of 100	Defines the sensitive range of insulin input	none	none
$InRXRatio$	The reaction between the populations InRX / InR2 for a insulin level of 100	Defines the sensitive range of insulin input	Speed of the input-system (how fast does PIP3 reaches steady state)	
$InR2Speed$, $PIP2Speed$, $PIP3Speed$	The reaction rates for the reactions $r2P$, $routPIP2$, $r9$	none	Speed of the input system (how fast does PIP3 reaches steady state)	none
$chicoaIns$, $chicoDropParameter$,	Maximal level of Chicoa and the drop of Chico if insulin is maximal compared to if there is no insulin	The reaction rates for the reactions $r2P$, $routPIP2$ and $r9$	Speed of the input system (how fast does PIP3 reaches steady state)	Maximum level of Chico
$lengthrhebaraise$	The inverse of the reaction rate of routRheba	none	How long and how strong does Rheba raises above steady level	

Figure 8.2.: The influence of selected parameters on the model

A. Code

A.1. Parameter Grouping

../maple/output.txt

```
    |\^/|     Maple 13 (X86 64 WINDOWS)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2009
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
# This file sets up a dimensionless grouping of the insulin pathway.
# It exports this grouping and some default values for all
# parameters to Matlab.
>
>
#Part 1: The grouping.
# The first dimension that will be removed is the length.
# This is done
# implicitly, by choosing the dimension to 1,
# and then use the total mass
# instead of the concentrations...
# The time and mass dimensions will be removed at the end...
#
# Besides the dimensionless we also requires that the parameters are
# valid for all positive values. This will require some variable
# transformation
> allSize := 1e-6:
>
>
#Part 1.1: ADP/ATP-Circle
# We assume a fixed level of ATP, enforced by the routATP reaction.
> assign(solve({
>         routATPP * maxATP = routATPPRev
>     },{routATPP})):
>
#Part 1.2: InRX-Circle
# There are the two reactions r1 and r2. For maximal insulin
# input we
# define the rations InRXratio and InR2Ratio
> assign(solve({
>         maxIns * routP + maxIns * InR2max * r2P = routPRev,
>         InR2max * maxIns * r2P = InR2amax  * r2rP,
>         InR1max^2 * r1rPRev = InR2max * r1rP,
>         r1rPRev = InRXSpeed,
>         r2P = InR2Speed,
>         InR2amax / InR2max = InR2Ratio,
>         InR1max / InR2max = InRXRatio,
>         InR1max + 2 * (InR2max + InR2amax) = initInRX
>     },{r2P,r2rP,r1rP,r1rPRev,routPRev, InR2max, InR2amax, InR1max})):
>
#Part 1.4: Chico-Flow
# Chico if insulin is present
> chicoIns := chicoDropParameter / (chicoDropParameter + 1) * chicoNo:
```

```
>
# If InR2a is maximal this should induce the maximal flow
# through Chico
> assign(solve({
>         routChicoP * chicoNo = routChicoPRev,
>         chicoIns * (routChicoP + r5P * InR2amax * maxATP)
>             = routChicoPRev
>     },{r5P, routChicoP, routChicoPRev}));
>
# The maximal PI3K flow is given by the maximal flow through Chico.
> maxPI3KFlow := chicoIns * InR2amax * r5P * maxATP:
>
#Part 1.5: PI3K-Flow
# r7rP, r6rPRev, routp60P, routp110P have to be chosen so that the
# flux through this reaction is bigger than maxPKBIFlow.
> r7rP := (1 + r7rPParameter) * maxPI3KFlow / (chicoaIns * PI3KIns):
> r6rPRev := (1 + r6rPRevParameter) * maxPI3KFlow / (p110Ins * p60Ins):
> routp60P := (1 + routp60PParameter) * maxPI3KFlow / p60No:
> routp110P := (1 + routp110PParameter) * maxPI3KFlow / p110No:
>
> assign(solve({
>         # p60 and p110 with and without insulin
>         p60No  * routp60P  = routp60PRev,
>         p110No * routp110P = routp110PRev,
>         p60Ins  * routp60P  + maxPI3KFlow = routp60PRev,
>         p110Ins * routp110P + maxPI3KFlow = routp110PRev,
>
>         # PI3K with and without insulin
>         PI3KNo * r6rP = p60No * p110No * r6rPRev,
>         p110Ins * p60Ins * r6rPRev = PI3KIns * r6rP + maxPI3KFlow,
>
>         # PI3Ka with max insulin (without this is zero)
>         maxPI3KFlow = maxPI3Ka * routPI3KaP,
>
>         # Chicoa with max insulin (without this is zero)
>         chicoaIns * r7rP * PI3KIns = maxPI3KFlow + maxPI3Ka * r7rPRev
>
>     },{PI3KIns, p110Ins, p60Ins, r6rP, r7rPRev, maxPI3Ka,
>     routp110PRev, routp60PRev})):
>
#Part 1.6: PIP2/PIP3-Circle
# The speed of this circle is given by its input reaction and r9
> routPIP2PRev := PIP2Speed:
> r9P := PIP3Speed;
                              r9P := PIP3Speed

>
> assign(solve({
>         #PIP2 level is steady
>         PIP2s * routPIP2P = routPIP2PRev,
>
>         #PTEN level is steady
>         PTENs * routPTENP = routPTENPRev,
>
>         #PIP3 with maximal insulin input (with no input its zero)
>         PIP2s * maxPI3Ka * r9P = maxPIP3 * PTENs * r11P
>     }, {r11P, routPIP2P, routPTENP})):
>
#Part 1.7: The PKBI-System
# PKBIIns defined over a range parameter
> PKBIIns := PKBIDropParameter / (PKBIDropParameter + 1) * PKBINo:
```

```
>
# PKB minimal (if Ins is max and TORC2a is max) defined over a range
# parameter
> minPKB := maxPKB * PKBDropTORC2aParameter
>      / (1 + PKBDropTORC2aParameter):
>
# The PKBa1 drop has to be smaller than the PKB drop..
> minPKBa1 := maxPKBa1 * PKBa1DropTORC2aParameter
>      / (1 + PKBa1DropTORC2aParameter)
>      * PKBDropTORC2aParameter / (PKBDropTORC2aParameter + 1):
>
> assign(solve({
>         #Defining the state of PKBI
>         PKBINo *  routPKBIP = routPKBIPRev,
>         PKBIIns * routPKBIP = routPKBIPRev-maxPKBIFlow,
>         PKBIIns * maxPIP3 * r12P = maxPKBIFlow,
>
>         #Defining the state of PDK1s
>         PDK1s * routPDK1P = routPDK1PRev,
>
>         #Defining the system if no TORC2a is present, thus
>         #PKB, PKBa1 is max,
>         #PKBa2 is zero
>         maxPKBIFlow = maxPKB * r15P * PDK1s * maxATP,
>         maxPKBa1 * routPKBa1P = maxPKB * r15P * PDK1s * maxATP,
>
>         #Defining the system if full TORC2a is present. Then
>         #PKB,PKBa1 is minimal
>         #(but not zero), PKBa2 is maximal and PKBa2 is in
>         #PKBa2T2a (we dont
>         #know)
>         maxPKBIFlow = minPKB * maxATP
>             * (r15P * PDK1s + r18P * maxTORC2a),
>         minPKB * r15P * PDK1s * maxATP
>             = minPKBa1 * (routPKBa1P + r24P * maxATP * maxTORC2a),
>         maxPKBa2 * r21P * PDK1s * maxATP = minPKB * r18P
>             * maxATP * maxTORC2a,
>         PKBaaT2a * routPKBaaP
>             = (minPKBa1 * r24P * maxTORC2a
>             + maxPKBa2 * PDK1s * r21P) * maxATP
>
>     },{r12P, r15P, r18P, r21P, r24P, routPDK1P, routPKBIP,
>         routPKBIPRev, routPKBa1P, routPKBaaP}));
>
# Now we need to know the maximal level of PKBaa.
# This is hard to say...
# But we can give a upper bound
> assign(solve({
>         maxPKBaa * routPKBaaP
>             = maxPKBa1 * r24P * maxATP * maxTORC2a
>             + maxPKBa2 * maxATP * PDK1s * r21P
>     },{maxPKBaa}));
>
#Part 1.8: Flows
> r27P := a1Foxo:
> r30P := a2Foxo:
> r33P := aaFoxo:
> r35P := a1Lobe:
> r37P := a2Lobe:
> r39P := aaLobe:
> r41P := a1TSC:
```

```
> r43P := a2TSC:
> r45P := aaTSC:
>
#foxoaNo has to be at least 1 (there has to be foxoa...)
> foxoaNo := foxoaNoP:
>
# Compute the drops of TSCa, Foxoa and Lobea
> minTSCa := TSCaNo * TSCaDrop / (TSCaDrop +1):
> minLobea := LobeaNo * LobeaDrop / (LobeaDrop +1):
> minFoxoa := foxoaNo * FoxoaDrop / (FoxoaDrop + 1):
>
> assign(solve({
>         #The flows for maximal PKBax, thus the minimal TSCa,
>         #Foxoa, Lobea
>         #Note that those value will not be reached, since PKBa1,
>         #PKBa2, PKBaa
>         #cannot be maximal at the same time (and maxPKBaa is
>         #only a weak
>         #upper bound. So min????a are only a (weak) lower bound...
>         (maxATP * (r41P * maxPKBa1 + r43P * maxPKBa2
>             + r45P * maxPKBaa)
>             * minTSCa + minTSCa * routTSCaP) = routTSCaPRev,
>         (maxATP * (r35P * maxPKBa1 + r37P * maxPKBa2
>             + r39P * maxPKBaa)
>             * minLobea + minLobea * routLobeaP) = routLobeaPRev,
>         (maxATP * (r27P * maxPKBa1 + r30P * maxPKBa2
>             + r33P * maxPKBaa)
>             * minFoxoa + minFoxoa * routFoxoaP) = routFoxoaPRev,
>
>         #The maxmial values of TSC,Foxo, Lobe (same note as as above)
>         (maxATP * (r41P * maxPKBa1 + r43P * maxPKBa2
>             + r45P * maxPKBaa))
>             * minTSCa = maxTSC * routTSCP,
>         (maxATP * (r35P * maxPKBa1 + r37P * maxPKBa2
>             + r39P * maxPKBaa))
>             * minLobea = maxLobe * routLobeP,
>         (maxATP * (r27P * maxPKBa1 + r30P * maxPKBa2
>             + r33P * maxPKBaa))
>             * minFoxoa = maxFoxo* routFoxoP,
>
>         #The maximal values of TSCa,Foxoa, Lobea
>         TSCaNo * routTSCaP = routTSCaPRev,
>         LobeaNo * routLobeaP = routLobeaPRev,
>         foxoaNo * routFoxoaP = routFoxoaPRev
>
>     },{routFoxoP, routFoxoaP, routFoxoaPRev, routLobeP,
>     routLobeaP, routLobeaPRev, routTSCP, routTSCaP, routTSCaPRev})):
>
#Part 1.9: Rheba-Circle
# The reaction r49P defines the strength of the rheba raise
> r49P := strengthRhebaRaise:
>
# The length of the Rheba raise is defined by routRhebaPRev
> routRhebaPRev := 1/lengthRhebaRaise:
>
> assign(solve({
>         #The level of GEF is steay
>         GEFs * routGEFP = routGEFPRev,
>
>         #The level of Rheba is steady
>         Rhebas * routRhebaP = routRhebaPRev,
>
>           #Rheb if no insulin is presnet
>           TSCaNo * r47P * Rhebas = GEFs * maxRheb * r49P
>      },{r47P, routGEFP, routRhebaP})):
>
>
#Part 1.10: TORC1-Circle
# The steady TORC1a level is given by the function
# (note Rheba is steady
# to and therefore does not influnce the steady level of TORC1a...)
> fTORC1a := lobea -> TORC1s*(r50rPRev*Rhebas+r51rP*lobea)
>      /(r50rP*Rhebas+r51rPRev*lobea):
>
> assign(solve({
>           #The level of TORC1 is steady
>           TORC1s * routTORC1P = routTORC1PRev,
>
>           #TORC1a if Lobea is zero
>           #(note: This will never be the case...)
>           fTORC1a(0) = TORC1aZero,
>
>           #TORC1a id Lobea is maximal
>           fTORC1a(LobeaNo) = TORC1aMin,
>
>           #TORC1a if Lobea is minimal
>           fTORC1a(minLobea) = TORC1aMax
>
>      }, {r51rPRev, r51rP, r50rPRev, routTORC1P})):
>
#Part 1.11: Prequisites for the S6K circle
# I order to compute the parameters for the S6Ka circle we first
# have to
# define how much TORC1a raises at maximal. We know that the reaction
# rates have to be positive thus
> sol := solve({r51rPRev > 0, r51rP > 0, r50rPRev > 0}, TORC1aZero)
>      assuming (LobeaNo > minLobea, minLobea > 0, r50rP > 0,
>      TORC1s > 0, Rhebas > 0, TORC1aZero > TORC1aMax,
>      TORC1aMax > TORC1aMin, TORC1aMin > 0,
>      LobeaDrop < 1, LobeaDrop > 0);
          {
 sol := {
          {
```

$$[\{-\frac{\text{TORC1aMin TORC1aMax}}{-\text{LobeaDrop TORC1aMin} + \text{TORC1aMax LobeaDrop} - \text{TORC1aMin}} < \text{TORC1aZero}\}] ,\quad \text{LobeaDrop} < \frac{\text{TORC1aMin}}{-\text{TORC1aMin} + \text{TORC1aMax}}$$

$$[\,] ,\quad \text{LobeaDrop} = \frac{\text{TORC1aMin}}{-\text{TORC1aMin} + \text{TORC1aMax}}$$

$$[\,] ,\quad \text{otherwise}$$

```
>
# In order for this to have a solutions we need the condition to be
# fulfilled
```

```
> solve({LobeaDrop < TORC1aMin /(TORC1aMax - TORC1aMin)},{TORC1aMin})
>     assuming (TORC1aMax > TORC1aMin, TORC1aMin > 0, LobeaDrop < 1,
>     LobeaDrop > 0);
```

$$\{TORC1aMin < TORC1aMax, \frac{LobeaDrop\ TORC1aMax}{LobeaDrop + 1} < TORC1aMin\}$$

```
>
# To fulfill this inequality let us introduce the
# TORCRangeParameter>0
# that shows us where in between lower and upper bound
# TORC1aMin is...
> TORC1aMin :=
>         TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax +
>         (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
>         LobeaDrop * TORC1aMax / (LobeaDrop+1):
>
# If we now insert this to the solution form above lower bound for
# TORC1aZero and introduce the (positive) parameter
# TORC1aZeroParameter and get
> TORC1aZero := (1 + TORC1aZeroParameter) * TORC1aMin * TORC1aMax
>     / (LobeaDrop * TORC1aMin - TORC1aMax * LobeaDrop + TORC1aMin):
>
>
#Part 1.12: S6K-Circle
# We define the ratio of S6K if no insulin ins present with an
# parameter
# (since this should be smaller than 1)
> ratioS6Kno := ratioS6KNoparameter / (1 + ratioS6KNoparameter):
>
# The speed of this circle is give by the reaction r54P
> r54P := S6KaSpeed:
>
> assign(solve({
>         #S6K without insulin
>         ratioS6Kno * initS6K * TORC1aMin * maxATP * r54P
>             = (1 - ratioS6Kno) * initS6K * r54rP,
>
>         #S6K when TORC1a is maximal
>         ratioS6Kins * initS6K * TORC1aMax * maxATP * r54P
>             = (1 - ratioS6Kins) * initS6K * r54rP
>     }, {r54rP, ratioS6Kins})):
>
>
#Lets see what we have for ratioS6Kins
> ratioS6Kins;
ratioS6KNoparameter

    (TORCRangeParameter LobeaDrop + TORCRangeParameter + LobeaDrop)/(

    ratioS6KNoparameter TORCRangeParameter LobeaDrop

     + ratioS6KNoparameter TORCRangeParameter + ratioS6KNoparameter LobeaDrop

     + TORCRangeParameter LobeaDrop + TORCRangeParameter + LobeaDrop + 1)

>
#Part 1.13: TORC2-Circle
# The speed of the TORC2a circle is given by the reaction rTORC2aP
> rTORC2aP := TORC2speed:
>
```

```
> assign(solve({
>         #TORC2a if S6K and InR2a are maximal
>         InR2amax * (initTORC2 - TORC2amaxInsmaxS6Ka) * rTORC2aP
>             = TORC2amaxInsmaxS6Ka * (1 - ratioS6Kins) * initS6K
>             * r61P * maxATP,
>         #TORC2a is maximal if InR2a is max, but S6Ka is minimal
>         InR2amax * (initTORC2 - maxTORC2a) * rTORC2aP
>             = maxTORC2a * (1 - ratioS6Kno) * initS6K * r61P * maxATP
>     }, {r61P, TORC2amaxInsmaxS6Ka})):
>
# The maximal amount of TORC2a is a part of the total TORC2a
> maxTORC2a := initTORC2 * maxTORC2aParameter
>     / (1 + maxTORC2aParameter):
>
#Part 1.14: 4EBP-Circle
# The speed of this circle is given by r56P
> r56P := X4EBPSpeed:
>
# Note: For any steady state the flow though r56 and r63r is equal...
> assign(solve({
>         #X4EBPa whith and without insulin
>         X4EBPaMax * routX4EBPaP = routX4EBPaPRev + r62P * foxoaNo,
>         X4EBPaMin * routX4EBPaP = routX4EBPaPRev + r62P * minFoxoa,
>
>         #X4EBP with and without insulin
>         X4EBPaMax * TORC1aMin * r56P * maxATP = X4EBPMin * r62rP,
>         X4EBPaMin * TORC1aMax * r56P * maxATP = X4EBPMax * r62rP
>     }, {r62P, r62rP, X4EBPMin, routX4EBPaP})):
>
# Now we want routX4EBPaP to be positive
> solve({routX4EBPaP > 0},{X4EBPaMin})
>     assuming (routX4EBPaPRev > 0, FoxoaDrop > 0);
                            FoxoaDrop X4EBPaMax
                          {------------------- < X4EBPaMin}
                               FoxoaDrop + 1

>
# But also X4EBPaMin < X4EBPaMax... Thus we can assign to X4EBPaMin
> X4EBPaMin := X4EBPaMax
>     * FoxoaDrop / (FoxoaDrop + 1)
>     * (((FoxoaDrop + 1) / FoxoaDrop -1)
>     * X4EBPaDropParameter / (X4EBPaDropParameter + 1) + 1):
>
#Part 2: Make the grouping dimensionless
# To make this grouping dimensionless we have to examine the
# dimensions of each parameter...
#                                         T        M
# routATPPRev                   [        -1        1   ]
# routADPP                      [        -1        0   ]
# routP                         [        -1       -1   ]
# maxIns                        [                  1   ]
# initInRX                      [                  1   ]
# InR2Ratio                     [                  0   ]
# InR2Speed                     [        -1       -1   ]
# InRXRatio                     [                  0   ]
# InRXSpeed                     [        -1       -1   ]
# chicoDropParameter            [                  0   ]
# chicoNo                       [                  1   ]
# r6rPRevParameter              [                  0   ]
# chicoaIns                     [                  1   ]
# PI3KNo                        [                  1   ]
# routChicoP                     [          1 ]
# p60No                          [          1 ]
# p110No                         [          1 ]
# routp60PParameter              [          0 ]
# routp110PParameter             [          0 ]
# r7rPParameter                  [          0 ]
# routPI3KaP                     [   −1     0 ]
# PTENs                          [          1 ]
# PIP2s                          [          1 ]
# PIP2Speed                      [   −1     1 ]
# routPTENPRev                   [   −1     1 ]
# maxPIP3                        [          1 ]
# PIP3Speed                      [   −1    −1 ]
# PKBIDropParameter              [          0 ]
# PKBINo                         [          1 ]
# maxPKB                         [          1 ]
# PDK1s                          [          1 ]
# maxPKBa1                       [          1 ]
# PKBa1DropTORC2aParameter       [          0 ]
# maxPKBIFlow                    [   −1     1 ]
# PKBDropTORC2aParameter         [          0 ]
# PKBaaT2a                       [          1 ]
# maxPKBa2                       [          1 ]
# routPDK1PRev                   [   −1     1 ]
# FoxoaDrop                      [          0 ]
# LobeaDrop                      [          0 ]
# TSCaDrop                       [          0 ]
# TSCaNo                         [          1 ]
# LobeaNo                        [          1 ]
# foxoaNoP                       [          1 ]
# a1Foxo                         [   −1    −2 ]
# a2Foxo                         [   −1    −2 ]
# aaFoxo                         [   −1    −2 ]
# a1TSC                          [   −1    −2 ]
# a2TSC                          [   −1    −2 ]
# aaTSC                          [   −1    −2 ]
# a1Lobe                         [   −1    −2 ]
# a2Lobe                         [   −1    −2 ]
# aaLobe                         [   −1    −2 ]
# GEFs                           [          1 ]
# Rhebas                         [          1 ]
# maxRheb                        [          1 ]
# strengthRhebaRaise             [   −1    −1 ]
# lengthRhebaRaise               [    1    −1 ]
# routGEFPRev                    [   −1     1 ]
# TORC1aMax                      [          1 ]
# TORC1aZeroParameter            [          0 ]
# TORCRangeParameter             [          0 ]
# r50rP                          [   −1    −1 ]
# TORC1s                         [          1 ]
# routTORC1PRev                  [   −1     1 ]
# X4EBPaMax                      [          1 ]
# X4EBPaDropParameter            [          0 ]
# X4EBPMax                       [          1 ]
# routX4EBPaPRev                 [   −1     1 ]
# initS6K                        [          1 ]
# ratioS6KNoparameter            [          0 ]
# X4EBPSpeed                     [   −1    −2 ]
# initTORC2                      [          1 ]
# TORC2speed                     [   −1    −1 ]
# S6KaSpeed                      [   −1    −2 ]
# maxTORC2aParameter              [                  0   ]
# maxATP                          [                  1   ]
# maxFoxo                         [                  1   ]
# maxLobe                         [                  1   ]
# maxTSC                          [                  1   ]
>
# Now I will use E.S. Taylors Method to remove dimensions. First lets
# remove the time using lengthRhebaRaise (thus simply multiply all
# variables that have a -1 in time by lengthRhebaRaise.
# Then remove the
# time by removing maxIns. Thus will lead in to a dimensionless
# grouping:
> grouping := [
> routATPPRev / maxIns ^2 / lengthRhebaRaise,
> routADPP / maxIns / lengthRhebaRaise,
> routP * maxIns^2 / lengthRhebaRaise,
> initInRX / maxIns,
> InR2Ratio,
> InR2Speed / lengthRhebaRaise,
> InRXRatio,
> InRXSpeed  / lengthRhebaRaise,
> chicoDropParameter,
> chicoNo / maxIns,
> r6rPRevParameter,
> chicoaIns / maxIns,
> PI3KNo / maxIns,
> routChicoP / maxIns,
> p60No / maxIns,
> p110No / maxIns,
> routp60PParameter,
> routp110PParameter,
> r7rPParameter,
> routPI3KaP / maxIns / lengthRhebaRaise,
> PTENs / maxIns,
> PIP2s / maxIns,
> PIP2Speed / maxIns^2 / lengthRhebaRaise,
> routPTENPRev / maxIns^2 / lengthRhebaRaise,
> maxPIP3 / maxIns,
> PIP3Speed / lengthRhebaRaise,
> PKBIDropParameter,
> PKBINo / maxIns,
> maxPKB / maxIns,
> PDK1s / maxIns,
> maxPKBa1 / maxIns,
> PKBa1DropTORC2aParameter,
> maxPKBIFlow / maxIns^2 / lengthRhebaRaise,
> PKBDropTORC2aParameter,
> PKBaaT2a / maxIns,
> maxPKBa2 / maxIns,
> routPDK1PRev / maxIns^2 / lengthRhebaRaise,
> FoxoaDrop,
> LobeaDrop,
> TSCaDrop,
> TSCaNo / maxIns,
> LobeaNo / maxIns,
> foxoaNoP / maxIns,
> a1Foxo * maxIns / lengthRhebaRaise,
> a2Foxo * maxIns / lengthRhebaRaise,
> aaFoxo * maxIns / lengthRhebaRaise,
> a1TSC * maxIns / lengthRhebaRaise,
> a2TSC * maxIns / lengthRhebaRaise,
> aaTSC * maxIns / lengthRhebaRaise ,
> a1Lobe * maxIns / lengthRhebaRaise ,
> a2Lobe * maxIns / lengthRhebaRaise ,
> aaLobe * maxIns / lengthRhebaRaise ,
> GEFs / maxIns ,
> Rhebas / maxIns ,
> maxRheb / maxIns ,
> strengthRhebaRaise / lengthRhebaRaise ,
> routGEFPRev / maxIns^2 / lengthRhebaRaise ,
> TORC1aMax / maxIns ,
> TORC1aZeroParameter ,
> TORCRangeParameter ,
> r50rP ,
> TORC1s / maxIns ,
> routTORC1PRev / maxIns^2 / lengthRhebaRaise ,
> X4EBPaMax / maxIns ,
> X4EBPaDropParameter ,
> X4EBPMax / maxIns ,
> routX4EBPaPRev / maxIns^2 / lengthRhebaRaise ,
> initS6K / maxIns ,
> ratioS6KNoparameter ,
> X4EBPSpeed * maxIns / lengthRhebaRaise ,
> initTORC2 / maxIns ,
> TORC2speed / lengthRhebaRaise ,
> S6KaSpeed * maxIns / lengthRhebaRaise ,
> maxTORC2aParameter ,
> maxATP / maxIns ,
> maxFoxo / maxIns ,
> maxLobe / maxIns ,
> maxTSC / maxIns ]:
>
# But to evaluate the influence of a specific parameter it sometimes
# easier to use the non-dimensionless grouping...
>
#Part 3: Export the grouping to Matlab
# Note: Those groupings are not dimensionless. For such groupings see
# part 2
>
> with(CodeGeneration):
> fffwith(linalg):
>
# Export formulas to compute the reaction rates from the parameters
> reactionRates := Matrix([r11P, r12P, r15P, r18P, r1rP, r1rPRev, r2P,
>     r21P, r24P, r27P, r2rP, r30P, r33P, r35P, r37P, r39P, r41P, r43P,
>     r45P, r47P, r49P, r5P, r50rP, r50rPRev, r51rP, r51rPRev, r54P,
>     r54rP, r56P, r61P, r62P, r62rP, r6rP, r6rPRev, r7rP, r7rPRev, r9P,
>     routP, routPRev, routATPP, routATPPRev, routChicoP,
>     routChicoPRev, routFoxoaP, routFoxoaPRev, routGEFP, routGEFPRev,
>     routLobeaP, routLobeaPRev, routPDK1P, routPDK1PRev, routPIP2P,
>     routPIP2PRev, routPKBIP, routPKBIPRev, routPTENP, routPTENPRev,
>     routRhebaP, routRhebaPRev, routTORC1P, routTORC1PRev, routTSCaP,
>     routTSCaPRev, routX4EBPaP, routX4EBPaPRev, routp110P,
>     routp110PRev, routp60P, routp60PRev, routFoxoP, routLobeP,
>     routADPP, routTSCP, routPKBaaP, routPKBa1P, routPI3KaP,
>     rTORC2aP, allSize]):
> Matlab(reactionRates);
cg = [PIP2s * routChicoP * chicoNo * PIP3Speed / maxPIP3 / PTENs / routPI3KaP
    / (chicoDropParameter + 1) maxPKBIFlow * (PKBIDropParameter + 1) /
    PKBIDropParameter / PKBINo / maxPIP3 maxPKBIFlow / maxPKB / PDK1s /
    maxATP maxPKBIFlow / maxPKB / PKBDropTORC2aParameter / maxATP / initTORC2
    / maxTORC2aParameter * (1 + maxTORC2aParameter) InRXRatio ^ 2 * initInRX
/ (InRXRatio + 2 + 2 * InR2Ratio) * InRXSpeed InRXSpeed InR2Speed
maxPKBIFlow / maxPKBa2 / PDK1s / maxATP / (1 + PKBDropTORC2aParameter)
maxPKBIFlow / maxPKBa1 / PKBa1DropTORC2aParameter / maxATP / initTORC2 /
maxTORC2aParameter * (1 + maxTORC2aParameter) a1Foxo maxIns * InR2Speed /
InR2Ratio a2Foxo aaFoxo a1Lobe a2Lobe aaLobe a1TSC a2TSC aaTSC GEFs *
maxRheb * strengthRhebaRaise / TSCaNo / Rhebas strengthRhebaRaise
(InRXRatio + 2 + 2 * InR2Ratio) * routChicoP / chicoDropParameter /
InR2Ratio / initInRX / maxATP r50rP (1 + TORC1aZeroParameter) *
(TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax + (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1)) * TORC1aMax / (LobeaDrop * (TORCRangeParameter /
(TORCRangeParameter + 1) * TORC1aMax + (1 - TORCRangeParameter /
(TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax / (LobeaDrop + 1)) -
TORC1aMax * LobeaDrop + TORCRangeParameter / (TORCRangeParameter + 1) *
TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) * r50rP / TORC1s -r50rP * Rhebas
* (-(1 + TORC1aZeroParameter) * (TORCRangeParameter / (TORCRangeParameter
+ 1) * TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) ^ 2 * TORC1aMax / (LobeaDrop *
(TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax + (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1)) - TORC1aMax * LobeaDrop + TORCRangeParameter /
(TORCRangeParameter + 1) * TORC1aMax + (1 - TORCRangeParameter /
(TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax / (LobeaDrop + 1)) *
LobeaDrop + (1 + TORC1aZeroParameter) * (TORCRangeParameter /
(TORCRangeParameter + 1) * TORC1aMax + (1 - TORCRangeParameter /
(TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax / (LobeaDrop + 1)) *
TORC1aMax ^ 2 / (LobeaDrop * (TORCRangeParameter / (TORCRangeParameter +
1) * TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) - TORC1aMax * LobeaDrop +
TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax + (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1)) * LobeaDrop - (TORCRangeParameter / (TORCRangeParameter
+ 1) * TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) ^ 2 * (1 + TORC1aZeroParameter)
* TORC1aMax / (LobeaDrop * (TORCRangeParameter / (TORCRangeParameter + 1)
* TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) - TORC1aMax * LobeaDrop +
TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax + (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1)) + (TORCRangeParameter / (TORCRangeParameter + 1) *
TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) * TORC1aMax) / LobeaDrop /
(-TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax - (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1) + TORC1aMax) / TORC1s / LobeaNo -r50rP * Rhebas *
(-LobeaDrop * (TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax
+ (1 - TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop *
TORC1aMax / (LobeaDrop + 1)) - (1 + TORC1aZeroParameter) *
(TORCRangeParameter / (TORCRangeParameter + 1) * TORC1aMax + (1 -
TORCRangeParameter / (TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax /
(LobeaDrop + 1)) * TORC1aMax / (LobeaDrop * (TORCRangeParameter /
(TORCRangeParameter + 1) * TORC1aMax + (1 - TORCRangeParameter /
(TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax / (LobeaDrop + 1)) -
TORC1aMax * LobeaDrop + TORCRangeParameter / (TORCRangeParameter + 1) *
TORC1aMax + (1 - TORCRangeParameter / (TORCRangeParameter + 1)) *
LobeaDrop * TORC1aMax / (LobeaDrop + 1)) + TORC1aMax * LobeaDrop +
TORC1aMax) / LobeaNo / LobeaDrop / (-TORCRangeParameter /
(TORCRangeParameter + 1) * TORC1aMax - (1 - TORCRangeParameter /
(TORCRangeParameter + 1)) * LobeaDrop * TORC1aMax / (LobeaDrop + 1) +
TORC1aMax) S6KaSpeed ratioS6KNoparameter * TORC1aMax *
```

```
(TORCRangeParameter * LobeaDrop + TORCRangeParameter + LobeaDrop) *
maxATP * S6KaSpeed / (TORCRangeParameter * LobeaDrop + TORCRangeParameter
+ LobeaDrop + 1) X4EBPSpeed InR2Ratio * initInRX * TORC2speed *
(initTORC2 + initTORC2 * ratioS6KNoparameter − initTORC2 *
maxTORC2aParameter / (1 + maxTORC2aParameter) − initTORC2 *
maxTORC2aParameter / (1 + maxTORC2aParameter) * ratioS6KNoparameter) /
initTORC2 / maxTORC2aParameter * (1 + maxTORC2aParameter) / initS6K /
maxATP / (InRXRatio + 2 + 2 * InR2Ratio) −routX4EBPaPRev * (X4EBPaMax +
FoxoaDrop * X4EBPaMax − X4EBPaMax * FoxoaDrop ^ 2 / (FoxoaDrop + 1) *
(((FoxoaDrop + 1) / FoxoaDrop − 1) * X4EBPaDropParameter /
(X4EBPaDropParameter + 1) + 1) − X4EBPaMax * FoxoaDrop / (FoxoaDrop + 1)
* (((FoxoaDrop + 1) / FoxoaDrop − 1) * X4EBPaDropParameter /
(X4EBPaDropParameter + 1) + 1)) / (FoxoaDrop * X4EBPaMax − X4EBPaMax *
FoxoaDrop ^ 2 / (FoxoaDrop + 1) * (((FoxoaDrop + 1) / FoxoaDrop − 1) *
X4EBPaDropParameter / (X4EBPaDropParameter + 1) + 1) − X4EBPaMax *
FoxoaDrop / (FoxoaDrop + 1) * (((FoxoaDrop + 1) / FoxoaDrop − 1) *
X4EBPaDropParameter / (X4EBPaDropParameter + 1) + 1)) / foxoaNoP
X4EBPaMax * FoxoaDrop / (FoxoaDrop + 1) * (((FoxoaDrop + 1) / FoxoaDrop −
1) * X4EBPaDropParameter / (X4EBPaDropParameter + 1) + 1) * TORC1aMax *
X4EBPSpeed * maxATP / X4EBPMax routChicoP * chicoNo * (1 +
routp110PParameter + routp60PParameter + routp60PParameter *
routp110PParameter + r6rPRevParameter + r6rPRevParameter *
routp110PParameter + r6rPRevParameter * routp60PParameter +
r6rPRevParameter * routp60PParameter * routp110PParameter) / PI3KNo /
routp60PParameter / routp110PParameter / (chicoDropParameter + 1) (1 +
r6rPRevParameter) / (chicoDropParameter + 1) * chicoNo * routChicoP /
p110No / routp110PParameter * (1 + routp110PParameter) / p60No /
routp60PParameter * (1 + routp60PParameter) (1 + r7rPParameter) /
(chicoDropParameter + 1) * chicoNo * routChicoP / chicoaIns /
r6rPRevParameter / PI3KNo / routp60PParameter / routp110PParameter * (1 +
routp110PParameter + routp60PParameter + routp60PParameter *
routp110PParameter + r6rPRevParameter + r6rPRevParameter *
routp110PParameter + r6rPRevParameter * routp60PParameter +
r6rPRevParameter * routp60PParameter * routp110PParameter) r7rPParameter
* routPI3KaP PIP3Speed routP maxIns * (routP * InRXRatio + 2 * routP + 2
* routP * InR2Ratio + initInRX * InR2Speed) / (InRXRatio + 2 + 2 *
InR2Ratio) routATPPRev / maxATP routATPPRev routChicoP routChicoP *
chicoNo maxATP * (a1Foxo * maxPKBa1 * PKBa1DropTORC2aParameter + a2Foxo *
maxPKBa2 * PKBa1DropTORC2aParameter + aaFoxo * PKBaaT2a + aaFoxo *
PKBaaT2a * PKBa1DropTORC2aParameter) * FoxoaDrop /
PKBa1DropTORC2aParameter foxoaNoP * maxATP * (a1Foxo * maxPKBa1 *
PKBa1DropTORC2aParameter + a2Foxo * maxPKBa2 * PKBa1DropTORC2aParameter +
aaFoxo * PKBaaT2a + aaFoxo * PKBaaT2a * PKBa1DropTORC2aParameter) *
FoxoaDrop / PKBa1DropTORC2aParameter routGEFPRev / GEFs routGEFPRev
maxATP * (a1Lobe * maxPKBa1 * PKBa1DropTORC2aParameter + a2Lobe *
maxPKBa2 * PKBa1DropTORC2aParameter + aaLobe * PKBaaT2a + aaLobe *
PKBaaT2a * PKBa1DropTORC2aParameter) * LobeaDrop /
PKBa1DropTORC2aParameter LobeaNo * maxATP * (a1Lobe * maxPKBa1 *
PKBa1DropTORC2aParameter + a2Lobe * maxPKBa2 * PKBa1DropTORC2aParameter +
aaLobe * PKBaaT2a + aaLobe * PKBaaT2a * PKBa1DropTORC2aParameter) *
LobeaDrop / PKBa1DropTORC2aParameter routPDK1PRev / PDK1s routPDK1PRev
PIP2Speed / PIP2s PIP2Speed maxPKBIFlow * (PKBIDropParameter + 1) /
PKBINo maxPKBIFlow * (PKBIDropParameter + 1) routPTENPRev / PTENs
routPTENPRev 1 / Rhebas / lengthRhebaRaise 1 / lengthRhebaRaise
routTORC1PRev / TORC1s routTORC1PRev maxATP * (a1TSC * maxPKBa1 *
PKBa1DropTORC2aParameter + a2TSC * maxPKBa2 * PKBa1DropTORC2aParameter +
aaTSC * PKBaaT2a + aaTSC * PKBaaT2a * PKBa1DropTORC2aParameter) *
TSCaDrop / PKBa1DropTORC2aParameter TSCaNo * maxATP * (a1TSC * maxPKBa1 *
PKBa1DropTORC2aParameter + a2TSC * maxPKBa2 * PKBa1DropTORC2aParameter +
aaTSC * PKBaaT2a + aaTSC * PKBaaT2a * PKBa1DropTORC2aParameter) *
TSCaDrop / PKBa1DropTORC2aParameter −routX4EBPaPRev / (FoxoaDrop *
    X4EBPaMax − X4EBPaMax ∗ FoxoaDrop ˆ 2 / (FoxoaDrop + 1) ∗ (((FoxoaDrop +
    1) / FoxoaDrop − 1) ∗ X4EBPaDropParameter / (X4EBPaDropParameter + 1) +
    1) − X4EBPaMax ∗ FoxoaDrop / (FoxoaDrop + 1) ∗ (((FoxoaDrop + 1) /
    FoxoaDrop − 1) ∗ X4EBPaDropParameter / (X4EBPaDropParameter + 1) + 1))
    routX4EBPaPRev (1 + routp110PParameter) / (chicoDropParameter + 1) ∗
    chicoNo ∗ routChicoP / p110No (1 + routp110PParameter) /
    (chicoDropParameter + 1) ∗ chicoNo ∗ routChicoP (1 + routp60PParameter) /
    (chicoDropParameter + 1) ∗ chicoNo ∗ routChicoP / p60No (1 +
    routp60PParameter) / (chicoDropParameter + 1) ∗ chicoNo ∗ routChicoP
    maxATP ∗ (a1Foxo ∗ maxPKBa1 ∗ PKBa1DropTORC2aParameter + a2Foxo ∗
    maxPKBa2 ∗ PKBa1DropTORC2aParameter + aaFoxo ∗ PKBaaT2a + aaFoxo ∗
    PKBaaT2a ∗ PKBa1DropTORC2aParameter) ∗ foxoaNoP ∗ FoxoaDrop / maxFoxo /
    PKBa1DropTORC2aParameter / (FoxoaDrop + 1) maxATP ∗ (a1Lobe ∗ maxPKBa1 ∗
    PKBa1DropTORC2aParameter + a2Lobe ∗ maxPKBa2 ∗ PKBa1DropTORC2aParameter +
    aaLobe ∗ PKBaaT2a + aaLobe ∗ PKBaaT2a ∗ PKBa1DropTORC2aParameter) ∗
    LobeaNo ∗ LobeaDrop / maxLobe / PKBa1DropTORC2aParameter / (LobeaDrop +
    1) routADPP maxATP ∗ (a1TSC ∗ maxPKBa1 ∗ PKBa1DropTORC2aParameter + a2TSC
    ∗ maxPKBa2 ∗ PKBa1DropTORC2aParameter + aaTSC ∗ PKBaaT2a + aaTSC ∗
    PKBaaT2a ∗ PKBa1DropTORC2aParameter) ∗ TSCaNo ∗ TSCaDrop / maxTSC /
    PKBa1DropTORC2aParameter / (TSCaDrop + 1) maxPKBIFlow ∗
    (PKBDropTORC2aParameter + 1 + PKBa1DropTORC2aParameter) / PKBaaT2a / (1 +
    PKBDropTORC2aParameter + PKBa1DropTORC2aParameter +
    PKBa1DropTORC2aParameter ∗ PKBDropTORC2aParameter) maxPKBIFlow / maxPKBa1
    routPI3KaP TORC2speed 0.1e−5;];
>
# Export all parameter as variables
> params := [ routATPPRev, routADPP, routP, maxIns, initInRX,
>     InR2Ratio, InR2Speed, InRXRatio, InRXSpeed, chicoDropParameter,
>     chicoNo, r6rPRevParameter, chicoaIns, PI3KNo, routChicoP, p60No,
>     p110No, routp60PParameter, routp110PParameter, r7rPParameter,
>     routPI3KaP, PTENs, PIP2s, PIP2Speed, routPTENPRev, maxPIP3,
>     PIP3Speed, PKBIDropParameter, PKBINo, maxPKB, PDK1s, maxPKBa1,
>     PKBa1DropTORC2aParameter, maxPKBIFlow, PKBDropTORC2aParameter,
>     PKBaaT2a, maxPKBa2, routPDK1PRev, FoxoaDrop, LobeaDrop,
>     TSCaDrop, TSCaNo, LobeaNo, foxoaNoP, a1Foxo, a2Foxo, aaFoxo,
>     a1TSC, a2TSC, aaTSC, a1Lobe, a2Lobe, aaLobe, GEFs, Rhebas,
>     maxRheb, strengthRhebaRaise, lengthRhebaRaise, routGEFPRev,
>     TORC1aMax, TORC1aZeroParameter, TORCRangeParameter, r50rP,
>     TORC1s, routTORC1PRev, X4EBPaMax, X4EBPaDropParameter,
>     X4EBPMax, routX4EBPaPRev, initS6K, ratioS6KNoparameter,
>     X4EBPSpeed, initTORC2, TORC2speed ,S6KaSpeed,
>     maxTORC2aParameter, maxATP, maxFoxo, maxLobe, maxTSC]:
> Matlab(params);
cg0 = [routATPPRev routADPP routP maxIns initInRX InR2Ratio InR2Speed
    InRXRatio InRXSpeed chicoDropParameter chicoNo r6rPRevParameter chicoaIns
    PI3KNo routChicoP p60No p110No routp60PParameter routp110PParameter
    r7rPParameter routPI3KaP PTENs PIP2s PIP2Speed routPTENPRev maxPIP3
    PIP3Speed PKBIDropParameter PKBINo maxPKB PDK1s maxPKBa1
    PKBa1DropTORC2aParameter maxPKBIFlow PKBDropTORC2aParameter PKBaaT2a
    maxPKBa2 routPDK1PRev FoxoaDrop LobeaDrop TSCaDrop TSCaNo LobeaNo
    foxoaNoP a1Foxo a2Foxo aaFoxo a1TSC a2TSC aaTSC a1Lobe a2Lobe aaLobe GEFs
    Rhebas maxRheb strengthRhebaRaise lengthRhebaRaise routGEFPRev TORC1aMax
    TORC1aZeroParameter TORCRangeParameter r50rP TORC1s routTORC1PRev
    X4EBPaMax X4EBPaDropParameter X4EBPMax routX4EBPaPRev initS6K
    ratioS6KNoparameter X4EBPSpeed initTORC2 TORC2speed S6KaSpeed
    maxTORC2aParameter maxATP maxFoxo maxLobe maxTSC];
>
# Export the name of the parameters
> names := map(convert,params,'string'):
> Matlab(names);
```

```
cg1 = ['routATPPRev' 'routADPP' 'routP' 'maxIns' 'initInRX' 'InR2Ratio'
    'InR2Speed' 'InRXRatio' 'InRXSpeed' 'chicoDropParameter' 'chicoNo'
    'r6rPRevParameter' 'chicoaIns' 'PI3KNo' 'routChicoP' 'p60No' 'p110No'
    'routp60PParameter' 'routp110PParameter' 'r7rPParameter' 'routPI3KaP'
    'PTENs' 'PIP2s' 'PIP2Speed' 'routPTENPRev' 'maxPIP3' 'PIP3Speed'
    'PKBIDropParameter' 'PKBINo' 'maxPKB' 'PDK1s' 'maxPKBa1'
    'PKBa1DropTORC2aParameter' 'maxPKBIFlow' 'PKBDropTORC2aParameter'
    'PKBaaT2a' 'maxPKBa2' 'routPDK1PRev' 'FoxoaDrop' 'LobeaDrop' 'TSCaDrop'
    'TSCaNo' 'LobeaNo' 'foxoaNoP' 'a1Foxo' 'a2Foxo' 'aaFoxo' 'a1TSC' 'a2TSC'
    'aaTSC' 'a1Lobe' 'a2Lobe' 'aaLobe' 'GEFs' 'Rhebas' 'maxRheb'
    'strengthRhebaRaise' 'lengthRhebaRaise' 'routGEFPRev' 'TORC1aMax'
    'TORC1aZeroParameter' 'TORCRangeParameter' 'r50rP' 'TORC1s'
    'routTORC1PRev' 'X4EBPaMax' 'X4EBPaDropParameter' 'X4EBPMax'
    'routX4EBPaPRev' 'initS6K' 'ratioS6KNoparameter' 'X4EBPSpeed' 'initTORC2'
    'TORC2speed' 'S6KaSpeed' 'maxTORC2aParameter' 'maxATP' 'maxFoxo'
    'maxLobe' 'maxTSC'];
>
# Now as least we have to compute the range in which
# those parameters can
# lie... For reaction rates we know them and we have
# to transform this to
# those parameters. As a size of the containment I
# assume 1/1000000 M^-1,
# in order to get reaction rates around zero. How this is done is
# described in the thesis...
> minimal := [
>     1e-4, 1e-4, 1e-4, 1e-6, 1e-6, 1e-8, 1e-3,
>     1e-8, 1e-4, 1e-10, 1e-6, 1e-10, 1e-6,
>     1e-6, 1e-4, 1e-6, 1e-6, 1e-10, 1e-10,
>     1e-10, 1e-4, 1e-6, 1e-6, 1e-4, 1e-4, 1e-6,
>     1e-3,1e-10,1e-6,1e-6,1e-6,1e-6,
>     1e-10,1e-10,1e-10,1e-6,
>     1e-6,1e-4,1e-10,1e-10,1e-10,1e-6,1e-6,
>     1e-10,1e-4,1e-4,1e-4,1e-4,1e-4,1e-4,1e-4,1e-4,
>     1e-6,1e-6,1e-6,1e-3,1e2,1e-4,
>     1e-6,1e-10,1e-10,1e-3,1e-6,
>     1e-4,1e-6,1e-10,1e-6,1e-4,
>     1e-6,1e-10,1e-4,1e-6,1e-4,1e-4,
>     1e-10,1e-6,1e-6,1e-6,1e-6]:
>
> maximal := [
>     1e-2, 1e-2, 1e-2, 1e+2, 1e+2, 1e+8, 1e-0,
>     1e+8, 1e-2, 1e+10, 1e+2, 1e+10, 1e+2,
>     1e+2, 1e-2, 1e+2, 1e+2, 1e+10, 1e+10,
>     1e+10, 1e-2, 1e+2, 1e+2, 1e-2, 1e-2, 1e+2,
>     1e+0,1e+10,1e+2,1e+2,1e+2,1e+2,
>     1e+10,1e+10,1e+10,1e+2,
>     1e+2,1e-2,1e+10,1e+10,1e+10,1e+2,1e+2,
>     1e+10,1e-0,1e-0,1e-0,1e-0,1e-0,1e-0,1e-0,1e-0,
>     1e+2,1e+2,1e+2,1e-0,1e4,1e-2,
>     1e+2,1e+10,1e+10,1e-0,1e+2,
>     1e-2,1e+2,1e+10,1e+2,1e-2,
>     1e+2,1e+10,1e-0,1e+2,1e-0,1e-0,
>     1e+10,1e+2,1e+2,1e+2,1e+2]:
>
> Matlab(minimal);
cg2 = [0.1e-3 0.1e-3 0.1e-3 0.1e-5 0.1e-5 0.1e-7 0.1e-2 0.1e-7 0.1e-3 0.1e-9
    0.1e-5 0.1e-9 0.1e-5 0.1e-5 0.1e-3 0.1e-5 0.1e-5 0.1e-9 0.1e-9 0.1e-9
    0.1e-3 0.1e-5 0.1e-5 0.1e-3 0.1e-3 0.1e-5 0.1e-2 0.1e-9 0.1e-5 0.1e-5
    0.1e-5 0.1e-5 0.1e-9 0.1e-9 0.1e-9 0.1e-5 0.1e-5 0.1e-3 0.1e-9 0.1e-9
    0.1e-9 0.1e-5 0.1e-5 0.1e-9 0.1e-3 0.1e-3 0.1e-3 0.1e-3 0.1e-3 0.1e-3
    0.1e-3 0.1e-3 0.1e-3 0.1e-5 0.1e-5 0.1e-5 0.1e-2 0.1e3 0.1e-3 0.1e-5
    0.1e-9 0.1e-9 0.1e-2 0.1e-5 0.1e-3 0.1e-5 0.1e-9 0.1e-5 0.1e-3 0.1e-5
    0.1e-9 0.1e-3 0.1e-5 0.1e-3 0.1e-3 0.1e-9 0.1e-5 0.1e-5 0.1e-5 0.1e-5];
> Matlab(maximal);
cg3 = [0.1e-1 0.1e-1 0.1e-1 0.1e3 0.1e3 0.1e9 0.1e1 0.1e9 0.1e-1 0.1e11 0.1e3
    0.1e11 0.1e3 0.1e3 0.1e-1 0.1e3 0.1e3 0.1e11 0.1e11 0.1e11 0.1e-1 0.1e3
    0.1e3 0.1e-1 0.1e-1 0.1e3 0.1e1 0.1e11 0.1e3 0.1e3 0.1e3 0.1e3 0.1e11
    0.1e11 0.1e11 0.1e3 0.1e3 0.1e-1 0.1e11 0.1e11 0.1e11 0.1e3 0.1e3 0.1e11
    0.1e1 0.1e1 0.1e1 0.1e1 0.1e1 0.1e1 0.1e1 0.1e1 0.1e1 0.1e3 0.1e3 0.1e3
    0.1e1 0.1e5 0.1e-1 0.1e3 0.1e11 0.1e11 0.1e1 0.1e3 0.1e-1 0.1e3 0.1e11
    0.1e3 0.1e-1 0.1e3 0.1e11 0.1e1 0.1e3 0.1e1 0.1e1 0.1e11 0.1e3 0.1e3
    0.1e3 0.1e3];
>
>
#Part 4: Parameters
# Now we set some default values to all the parameters...
# Normally such
# that the maximal value is 100 for every species...
>
> maxATP := 1:
> routATPPRev := 1000000:
> routADPP := 0.01:
> routP := 1000:
> maxIns := 100:
> initInRX := 100:
> InR2Ratio := 100:
> InR2Speed := 0.1e-2:
> InRXRatio := 1:
> InRXSpeed := 0.1e-2:
> chicoDropParameter :=0.1:
> routChicoP := 0.0001:
> chicoNo := 100:
> r6rPRevParameter := 1:
> chicoaIns := 60:
> PI3KNo := 200:
> p60No := 100:
> p110No := 100:
> r7rPParameter := 1:
> routPI3KaP := 0.01:
> PTENs := 100:
> PIP2s := 100:
> PIP2Speed := 0.01:
> routPTENPRev := 0.01:
> maxPIP3 := 100:
> PIP3Speed := 0.1e-2:
> routp60PParameter := 0.1:
> routp110PParameter:= 0.1:
> PKBIDropParameter := 0.1:
> PKBINo := 100:
> maxPKB := 100:
> PDK1s := 100:
> maxPKBa1 := 100:
> PKBa1DropTORC2aParameter := 10:
> maxPKBIFlow := 0.1:
> PKBDropTORC2aParameter := 1:
> PKBaaT2a := 1:
> maxPKBa2 := 100:
> routPDK1PRev := 0.01:
> FoxoaDrop := 1/3:
> LobeaDrop := 1/99:
> TSCaDrop := 1/3:
```

```
> TSCaNo := 100:
> LobeaNo := 100:
> foxoaNoP := 100:
> a1Foxo := 0.0001:
> a2Foxo := 0.0001:
> aaFoxo := 0.0001:
> a1TSC := 0.0001:
> a2TSC := 0.0001:
> aaTSC := 0.0001:
> a1Lobe := 0.0001:
> a2Lobe := 0.0001:
> aaLobe := 0.0001:
> maxFoxo := 100:
> maxTSC := 100:
> maxLobe := 100:
> GEFs := 100:
> Rhebas := 100:
> maxRheb := 100:
> strengthRhebaRaise := .1:
> lengthRhebaRaise := 100:
> routGEFPRev := 0.01:
> TORC1aMax := 30:
> TORC1aZeroParameter := 0.1:
> TORCRangeParameter := 0.1:
> r50rP := 0.01:
> TORC1s := 100:
> routTORC1PRev := 0.01:
> X4EBPaMax := 99:
> X4EBPaDropParameter := 10/89:
> X4EBPMax := 99:
> routX4EBPaPRev := 0.01:
> initS6K := 100:
> ratioS6KNoparameter := 3:
> X4EBPSpeed := 0.1:
> initTORC2 := 100:
> TORC2speed := 0.1:
> S6KaSpeed := 0.1:
> maxTORC2aParameter := 90:
>
#Part 5: Export values to Matlab
# Now we want also to export the values computed
# But first print out all reaction rates to see if they are valid...
> Digits := 5:
> evalf([r1rP, r1rPRev, r2P, r2rP, r5P, r6rP, r6rPRev, r7rP, r7rPRev]);
[0.00049261, 0.001, 0.001, 0.0010000, 0.000020300, 0.011000, 0.00022000,

    0.00036666, 0.01]

> evalf([r9P, r11P, r12P, r15P, r18P, r21P, r24P]);
                      -5                                                          -5
[0.001, 0.90909 10   , 0.00011000, 0.000010000, 0.000010111, 0.50000 10   ,

                 -5
    0.10111 10   ]

> evalf([r27P, r30P, r33P, r35P, r37P, r39P, r41P, r43P, r45P]);
    [0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001]

> evalf([r47P, r49P, r50rP, r50rPRev, r51rP, r51rPRev]);
              [0.10000, 0.1, 0.01, 0.0036300, 0.0033000, 0.22099]
```

```
> evalf([r54P, r54rP, r56P, r61P, r62P, r62rP, rTORC2aP]);
            [0.1, 0.90000, 0.1, 0.0021894, 0.00089000, 0.97727, 0.1]

> evalf([routP,routPRev, routADPP, routATPP, routATPPRev, routChicoP]);
                                                  7            7
            [1000., 100000., 0.01, 0.10000 10 , 0.10000 10 , 0.0001]

> evalf([routChicoPRev,routFoxoP,routFoxoaP, routFoxoaPRev, routGEFP]);
              [0.0100, 0.0050275, 0.0067033, 0.67033, 0.00010000]

> evalf([routGEFPRev,routLobeP,routLobeaP, routLobeaPRev, routPDK1P]);
              [0.01, 0.00020110, 0.00020313, 0.020313, 0.00010000]

> evalf([routPDK1PRev,routPI3KaP, routPIP2P, routPIP2PRev, routPKBIP]);
                   [0.01, 0.01, 0.00010000, 0.01, 0.0011000]

> evalf([routPKBIPRev,routPKBa1P,routPKBaaP, routPTENP, routPTENPRev]);
                  [0.11, 0.0010000, 0.054545, 0.00010000, 0.01]

> evalf([routRhebaP,routRhebaPRev,routTORC1P,routTORC1PRev, routTSCP]);
              [0.00010000, 0.010000, 0.00010000, 0.01, 0.0050275]

> evalf([routTSCaP,routTSCaPRev,routX4EBPaP,routX4EBPaPRev, routp60P]);
               [0.0067033, 0.67033, 0.0010000, 0.01, 0.00010000]

> evalf([routp60PRev, routp110P, routp110PRev]);
                        [0.010000, 0.00010000, 0.010000]

>
# Export parameter values
> Matlab(params);
cg4 = [1000000 0.1e-1 1000 100 100 100 0.1e-2 1 0.1e-2 0.1e0 100 1 60 200
    0.1e-3 100 100 0.1e0 0.1e0 1 0.1e-1 100 100 0.1e-1 0.1e-1 100 0.1e-2
    0.1e0 100 100 100 100 10 0.1e0 1 1 100 0.1e-1 0.1e1 / 0.3e1 0.1e1 /
    0.99e2 0.1e1 / 0.3e1 100 100 100 0.1e-3 0.1e-3 0.1e-3 0.1e-3 0.1e-3
    0.1e-3 0.1e-3 0.1e-3 0.1e-3 100 100 100 0.1e0 100 0.1e-1 30 0.1e0 0.1e0
    0.1e-1 100 0.1e-1 99 0.10e2 / 0.89e2 99 0.1e-1 100 3 0.1e0 100 0.1e0
    0.1e0 90 1 100 100 100];
>
#Export reactionrates )to file and to stdout).
> exportVec := evalf([r11P, r12P, r15P, r18P, r1rP, r1rPRev, r2P,
>     r21P,  r24P, r27P, r2rP, r30P, r33P, r35P, r37P, r39P, r41P,
>     r43P, r45P, r47P, r49P, r5P, r50rP, r50rPRev, r51rP, r51rPRev,
>     r54P, r54rP, r56P, r61P, r62P, r62rP, r6rP, r6rPRev, r7rP,
>     r7rPRev, r9P, routP, routPRev, routATPP, routATPPRev,
>     routChicoP, routChicoPRev, routFoxoaP, routFoxoaPRev, routGEFP,
>     routGEFPRev, routLobeaP, routLobeaPRev, routPDK1P, routPDK1PRev,
>     routPIP2P, routPIP2PRev, routPKBIP, routPKBIPRev, routPTENP,
>     routPTENPRev, routRhebaP, routRhebaPRev, routTORC1P,
>     routTORC1PRev, routTSCaP, routTSCaPRev, routX4EBPaP,
>     routX4EBPaPRev, routp110P, routp110PRev, routp60P, routp60PRev,
>     routFoxoP, routLobeP, routADPP, routTSCP, routPKBaaP,
>     routPKBa1P, routPI3KaP, rTORC2aP, allSize]):
>
>
> Matlab(exportVec);
cg5 = [0.90909e-5 0.11000e-3 0.10000e-4 0.10111e-4 0.49261e-3 0.1e-2 0.1e-2
    0.50000e-5 0.10111e-5 0.1e-3 0.10000e-2 0.1e-3 0.1e-3 0.1e-3 0.1e-3
    0.1e-3 0.1e-3 0.1e-3 0.1e-3 0.10000e0 0.1e0 0.20300e-4 0.1e-1 0.36300e-2
    0.33000e-2 0.22099e0 0.1e0 0.90000e0 0.1e0 0.21894e-2 0.89000e-3
    0.97727e0 0.11000e-1 0.22000e-3 0.36666e-3 0.1e-1 0.1e-2 0.1000e4
    0.10000e6 0.10000e7 0.10000e7 0.1e-3 0.100e-1 0.67033e-2 0.67033e0
    0.10000e-3 0.1e-1 0.20313e-3 0.20313e-1 0.10000e-3 0.1e-1 0.10000e-3
    0.1e-1 0.11000e-2 0.11e0 0.10000e-3 0.1e-1 0.10000e-3 0.10000e-1
    0.10000e-3 0.1e-1 0.67033e-2 0.67033e0 0.10000e-2 0.1e-1 0.10000e-3
    0.10000e-1 0.10000e-3 0.10000e-1 0.50275e-2 0.20110e-3 0.1e-1 0.50275e-2
    0.54545e-1 0.10000e-2 0.1e-1 0.1e0 0.1e-5];
>
> for i from 1 to nops(params) do
>     if (params[i] < minimal[i]) then
>         printf("Value for param %s is %g, this is lower than %g\n",
>         names[i],evalf(params[i]),evalf(minimal[i]));
>     end;
>
>     if (params[i] > maximal[i]) then
>         printf("Value for param %s is %g, this is higher than %g\n",
>         names[i],evalf(params[i]),evalf(maximal[i]));
>     end;
> end;
Value for param routATPPRev is 1.0000e+06, this is higher than 0.01
Value for param routP is 1000, this is higher than 0.01
Value for param PI3KNo is 200, this is higher than 100
>
> fffwith(linalg):
> exportVec := Matrix(1,nops(exportVec), exportVec):
> ExportMatrix("../simulation/reactionRates.mat", exportVec,
>     target = Matlab):
Error, (in ExportMatrix) argument 'target␣=␣CodeGeneration:-Matlab' invalid:
    rhs
should be of type {identical(Matlab), identical(MatrixMarket),
identical(delimited)}
>
>
#Part 6: Check if the system has only one solution
# Therefore set up the equations for all reactions
>
> unassign('routP');
>
> r11 := PTEN *PIP3 *r11P:
> r12 := PIP3 *PKBI *r12P:
> r15 := PDK1 *ATP *PKB *r15P:
> r18 := TORC2a *ATP *PKB *r18P:
> r1r := (InR2 *r1rP-InR1 ^2*r1rPRev):
> r2 := InR2 *Ins *r2P:
> r21 := PDK1 *ATP *PKBa2 *r21P:
> r24 := TORC2a *ATP *PKBa1 *r24P:
> r27 := PKBa1 *ATP *Foxoa *r27P:
> r2r := InR2a *r2rP:
> r30 := PKBa2 *ATP *Foxoa *r30P:
> r33 := PKBaa *ATP *Foxoa *r33P:
> r35 := PKBa1 *ATP *Lobea *r35P:
> r37 := PKBa2 *ATP *Lobea *r37P:
> r39 := PKBaa *ATP *Lobea *r39P:
> r41 := PKBa1 *ATP *TSCa *r41P:
> r43 := PKBa2 *ATP *TSCa *r43P:
> r45 := PKBaa *ATP *TSCa *r45P:
> r47 := TSCa *Rheba *r47P:
> r49 := GEF *Rheb *r49P:
> r5 := InR2a *ATP *Chico *r5P:
> r50r := Rheba *(TORC1a *r50rP-TORC1 *r50rPRev):
> r51r := Lobea *(TORC1 *r51rP-TORC1a *r51rPRev):
> r54 := TORC1a *ATP *S6K *r54P:
```

```
> r54r := S6Ka *r54rP:
> r56 := TORC1a *ATP *X4EBPa *r56P:
> r61 := S6Ka *ATP *TORC2a *r61P:
> r62 := Foxoa *r62P:
> r62r := X4EBP *r62rP:
> r6r := (PI3K *r6rP-p60 *p110 *r6rPRev):
> r7r := (PI3Ka *r7rP-Chicoa *PI3K *r7rPRev):
> r9 := PI3Ka *PIP2 *r9P:
> rout := (Ins *routP-routPRev):
> routATP := (ATP *routATPP-routATPPRev):
> routChico := (Chico *routChicoP-routChicoPRev):
> routFoxoa := (Foxoa *routFoxoaP-routFoxoaPRev):
> routGEF := (GEF *routGEFP-routGEFPRev):
> routLobea := (Lobea *routLobeaP-routLobeaPRev):
> routPDK1 := (PDK1 *routPDK1P-routPDK1PRev):
> routPIP2 := (PIP2 *routPIP2P-routPIP2PRev):
> routPKBI := (PKBI *routPKBIP-routPKBIPRev):
> routPTEN := (PTEN *routPTENP-routPTENPRev):
> routRheba := (Rheba *routRhebaP-routRhebaPRev):
> routTORC1 := (TORC1 *routTORC1P-routTORC1PRev):
> routTSCa := (TSCa *routTSCaP-routTSCaPRev):
> routX4EBPa := (X4EBPa *routX4EBPaP-routX4EBPaPRev):
> routp110 := (p110 *routp110P-routp110PRev):
> routp60 := (p60 *routp60P-routp60PRev):
> routFoxo := Foxo *routFoxoP:
> routLobe := Lobe *routLobeP:
> routADP := ADP *routADPP:
> routTSC := TSC *routTSCP:
> routPKBaa := PKBaa *routPKBaaP:
> routPKBa1 := PKBa1 *routPKBa1P:
> routPI3Ka := PI3Ka *routPI3KaP:
> rTORC2aNEW := InR2a *TORC2 *rTORC2aP:
>
# and the equations for all the substances
> D_ADP  := r15 +r18 +r21 +r24 +r27 +r30 +r33 +r35 +r37 +r39 +r41
>      +r43 +r45 + r5 +r54 +r56 +r61 -routADP :
> D_ATP  := -r15 -r18 -r21 -r24 -r27 -r30 -r33 -r35 -r37 -r39 -r41
>      -r43 - r45 -r5 -r54 -r56 -r61 -routATP :
> D_Chico  := -r5 -routChico :
> D_Chicoa  := r5 +r7r :
> D_Foxo  := r27 +r30 +r33 -routFoxo :
> D_Foxoa  := -r27 -r30 -r33 -routFoxoa :
> D_GEF  := -routGEF :
> D_InR1  := 2*r1r :
> D_InR2  := -r1r -r2 +r2r :
> D_InR2a  := r2 -r2r :
> D_Ins  := -r2 -rout :
> D_Lobe  := r35 +r37 +r39 -routLobe :
> D_Lobea  := -r35 -r37 -r39 -routLobea :
> D_PDK1  := -routPDK1 :
> D_PI3K  := -r6r +r7r :
> D_PI3Ka  := -r7r -routPI3Ka :
> D_PIP2  := r11 -r9 -routPIP2 :
> D_PIP3  := -r11 +r9 :
> D_PKB  := r12 -r15 -r18 :
> D_PKBI  := -r12 -routPKBI :
> D_PKBa1  := r15 -r24 -routPKBa1 :
> D_PKBa2  := r18 -r21 :
> D_PTEN  := -routPTEN :
> D_Rheb  := r47 -r49 :
> D_Rheba  := -r47 +r49 -routRheba :
```

```
> D_S6K   := -r54 +r54r  :
> D_S6Ka   := r54 -r54r  :
> D_TORC1   := r50r -r51r -routTORC1  :
> D_TORC1a   := -r50r +r51r  :
> D_TORC2   := r61 -rTORC2aNEW  :
> D_TORC2a   := -r61 +rTORC2aNEW  :
> D_TSC   := r41 +r43 +r45 -routTSC  :
> D_TSCa   := -r41 -r43 -r45 -routTSCa  :
> D_X4EBP   := r56 -r62r  :
> D_X4EBPa   := -r56 +r62 +r62r -routX4EBPa  :
> D_p110   := r6r -routp110  :
> D_p60   := r6r -routp60  :
> D_PKBaa   := r21 +r24 -routPKBaa  :
>
# Then solve this system
# Again we assume that the ATP level stays on 100 as above
>
> sol := solve(
>     {D_GEF = 0, D_Chico = 0, D_Chicoa = 0, D_Foxo = 0, D_Foxoa = 0,
>         D_InR2 = 0, D_InR2a = 0, D_Ins = 0, D_Lobe = 0, D_Lobea = 0,
>         D_PDK1 = 0, D_PI3K = 0, D_PI3Ka = 0, D_PIP2 = 0, D_PIP3 = 0,
>         D_PKB = 0, D_PKBI = 0, D_PKBa1 = 0, D_PKBa2 = 0,
>         D_PKBaa = 0, D_PTEN = 0, D_Rheb = 0, D_Rheba = 0,
>         D_TORC1 = 0,  D_TORC1a = 0, D_TORC2a = 0, D_TSC = 0,
>         D_TSCa = 0, D_X4EBP = 0,  D_X4EBPa = 0, D_p110 = 0,
>         D_p60 = 0, D_S6Ka = 0, D_ADP = 0,
>         InR1 + (InR2 + InR2a) * 2 = initInRX ,  initS6K = S6K + S6Ka,
>         TORC2 = initTORC2 - TORC2a, ATP = 100},
>     {GEF, Chico , Chicoa , Foxo , Foxoa , InR2 , InR2a , Ins , Lobe , Lobea , PI3K ,
>     PI3Ka , PDK1, PIP2 , PIP3 , PKB, PKBI, PKBa1 , PKBa2, PKBaa, PTEN, Rheb ,
>     Rheba , TORC1, TORC1a, TORC2a, X4EBP, X4EBPa, TSC, TSCa, p110 , p60 , S6Ka ,
>     ADP, TORC2, InR1 , S6K, ATP}):
>
# We are only interested in the number of solution , not in the
# solution itself. If there is one steady level per species
# there should be 38
# solutions:
> nops(sol);
                                    38

>
>
>
> quit
```

A.2. Symbolic Analyze and Simulation of the Reduces Steady State Model

../maple/output_simple.txt

```
    |\^/|      Maple 13 (X86 64 WINDOWS)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2009
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
# First compute the functions. See thesis for details how to choose the
# speces
# compute InR2a(Ins)
>
> sol := solve({
```

```
>       InR1 * InR1 * r1P = InR2 * r1PRev,
>       InR2 * Ins * r2P = InR2a * r2rP,
>       InR1 + 2 * (InR2a + InR2) = initInRX
>       },{InR1, InR2, InR2a}, Explicit = true):
>
> sol[1,3];
InR2a = (4 r1P initInRX r2rP + r2rP r1PRev + 4 r1P Ins r2P initInRX + (

                              2                 2        2
    8 r1P initInRX r2rP  r1PRev + r2rP  r1PRev

                                                1/2            /
     + 8 r2rP r1PRev r1P Ins r2P initInRX)   ) Ins r2P  /  (8 r1P
                                                             /

          2        2     2
    (r2rP  + Ins  r2P  + 2 Ins r2P r2rP))

> sol[2,3];
InR2a = (4 r1P initInRX r2rP + r2rP r1PRev + 4 r1P Ins r2P initInRX - (

                              2                 2        2
    8 r1P initInRX r2rP  r1PRev + r2rP  r1PRev

                                                1/2            /
     + 8 r2rP r1PRev r1P Ins r2P initInRX)   ) Ins r2P  /  (8 r1P
                                                             /

          2        2     2
    (r2rP  + Ins  r2P  + 2 Ins r2P r2rP))

>
# This can be reformed to
#
#   Ins * initInRX/2       r2P * Ins * (r2rP +/- sqrt(...))
#  ------------------- +  -----------------------------------
#   (r2rP/r2P + Ins)        8r1P * (r2rP + Ins * r2P)^2
#
# Thus a Michaelis Menten reaction plus and a higher order correction term.
# Lets approximate this without the h.o.t
>
# Now lets compute PKBI(InR2a)
>
>
> r5Flux := InR2a * r5P * maxATP * Chico / (r5P2 + InR2a):
> r7Flux := Chicoa * PI3K * r7P - PI3K * r7PRev:
> r6Flux := p110 * p60 * r6P - PI3K * r6rP:
> r9Flux := PIP2 * PI3Ka * r9P / (r9P2 + PI3Ka):
> r12Flux := PKBI * PIP3 * r12P / (r12P2 + PIP3):
>
> sol := solve({
>       Chico * routChicoP + r5Flux = routChicoPRev,
>       r5Flux = r7Flux,
>       routp60P * p60 + r6Flux = routp60PRev,
>       routp110P * p110 + r6Flux = routp110PRev,
>       r6Flux = r7Flux,
>       r7Flux = PI3Ka * routPI3KaP,
>       r9Flux = PIP3 * r11P * PTENs,
>       routPIP2 * PIP2 = routPIP2PRev,
>       PKBI * routPKBIP + r12Flux = routPKBIPRev
>       },{Chico, Chicoa, p60, p110, PI3K, PI3Ka, PIP3, PIP2, PKBI}):
```

```
>
> sol[7];
PKBI = routPKBIPRev (r12P2 r11P PTENs routPIP2 InR2a r5P maxATP routChicoPRev

     + r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP routChicoP r5P2

     + r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP routChicoP InR2a

     + r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP InR2a r5P maxATP

     + routPIP2PRev r9P InR2a r5P maxATP routChicoPRev)/(

    routPIP2PRev r9P InR2a r5P maxATP routChicoPRev routPKBIP

     + routPIP2PRev r9P InR2a r5P maxATP routChicoPRev r12P

     + routPKBIP r12P2 r11P PTENs routPIP2 InR2a r5P maxATP routChicoPRev

     + routPKBIP r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP routChicoP r5P2

     + routPKBIP r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP routChicoP InR2a

     + routPKBIP r12P2 r11P PTENs routPIP2 r9P2 routPI3KaP InR2a r5P maxATP)

>
#This can be written as
#         A * InR2a + routPKBIPRev
# PKBI = ----------------------------
#            B * InR2a + routPKBIP
#
>
#Now if we know PKBI we know the flux into PKB. From this I try to
#compute the levels of Lobea, TSCa, Foxoa
>
> r12Flux := routPKBIPRev - PKBI * routPKBIP:
> r18Flux := PKB   * TORC2a * r18P * maxATP / (TORC2a + r18P2):
> r24Flux := PKBa1 * TORC2a * r24P * maxATP / (TORC2a + r24P2):
> r15Flux := PKB   * maxATP * r15P:
> r21Flux := PKBa2 * maxATP * r21P:
> sol := solve({
>     r12Flux = r18Flux + r15Flux,
>     r15Flux = r24Flux + PKBa1 * routPKBa1P,
>     r18Flux = r21Flux,
>     r24Flux + r21Flux = PKBaa * routPKBaaP
>     },{PKB,PKBa1,PKBa2,PKBaa}):
>
>     sol[2];
PKBa1 = - (-routPKBIPRev TORC2a + PKBI routPKBIP TORC2a + PKBI routPKBIP r18P2

                                                              2
     - routPKBIPRev r18P2) r15P (TORC2a + r24P2)/(TORC2a  r18P r24P maxATP

              2
     + TORC2a  r18P routPKBa1P + TORC2a r18P routPKBa1P r24P2

              2                               2
     + TORC2a  r24P maxATP r15P + r15P TORC2a  routPKBa1P

     + r15P TORC2a routPKBa1P r24P2 + r24P maxATP r15P r18P2 TORC2a

     + r15P r18P2 TORC2a routPKBa1P + r15P r18P2 routPKBa1P r24P2)
```

```
>       sol[3];
                            TORC2a r18P (-routPKBIPRev + PKBI routPKBIP)
          PKBa2 = - ---------------------------------------------------------------
                    maxATP r21P (TORC2a r18P + r15P TORC2a + r15P r18P2)

>       sol[4];
PKBaa = - TORC2a (r18P TORC2a r24P maxATP PKBI routPKBIP

     - r18P TORC2a r24P maxATP routPKBIPRev

     + r18P TORC2a routPKBa1P PKBI routPKBIP

     + r18P routPKBa1P r24P2 PKBI routPKBIP

     - r18P routPKBa1P r24P2 routPKBIPRev - r24P maxATP r15P r18P2
         routPKBIPRev

     - r18P TORC2a routPKBa1P routPKBIPRev

     + r24P maxATP r15P r18P2 PKBI routPKBIP

     + TORC2a r24P maxATP r15P PKBI routPKBIP

     - TORC2a r24P maxATP r15P routPKBIPRev)/(routPKBaaP (

           2                           2
    TORC2a  r18P r24P maxATP + TORC2a  r18P routPKBa1P

                                                 2
     + TORC2a r18P routPKBa1P r24P2 + TORC2a  r24P maxATP r15P

                      2
     + r15P TORC2a  routPKBa1P + r15P TORC2a routPKBa1P r24P2

     + r24P maxATP r15P r18P2 TORC2a + r15P r18P2 TORC2a routPKBa1P

     + r15P r18P2 routPKBa1P r24P2))

>
# As can be seen from above the levels of all active variants of PKB can
# be expressed as a broken rational function of degree 2 (TORC2a) and 1
# (PKBI). For PKBa2 deg(TORC2a) is even 1... Thus also a weighted sum of
# those (and the deactivations flux for Lobea, TSCa and Foxoa is nothing
# else...( is a broken rational function of degree 2 (resp. 1). Thus the
# level of Foxoa etc. can be expressed as
#
# Foxoa = routFoxoaPRev / (routFoxoaP + f(TORC2a, PKBI));
#
>
# Now lets check TORC1a(Lobea)
>
> r51Flux := TORC1s * Lobea * r51P / (Lobea + r51P2)
>     - TORC1a * Lobea * r51PRev / (Lobea + r51PRev2):
> r50Flux := TORC1s * Rhebas * r50P - TORC1a * Rhebas * r50PRev:
> sol := solve({
>     r51Flux + r50Flux = 0
>     },{TORC1a}):
> sol[1];
TORC1a = TORC1s (Lobea r51P + Rhebas r50P Lobea + Rhebas r50P r51P2)
```

```
        (Lobea + r51PRev2)/((Lobea + r51P2)

        (Lobea r51PRev + Rhebas r50PRev Lobea + Rhebas r50PRev r51PRev2))

>
#As can be seen this is a broken rational function of degree 2 in Lobea
#
# Lets check S6Ka(TORC1a)
>
> sol := solve({
>     S6K * TORC1a * r54P / (r54P2 + TORC1a) = S6Ka *r54rP,
>     S6K + S6Ka = initS6K
>     },{S6K, S6Ka}):
> sol[1];
                                 initS6K r54rP (r54P2 + TORC1a)
                      S6K = ------------------------------------------
                            TORC1a r54P + r54rP r54P2 + r54rP TORC1a

> sol[2];
                                      TORC1a r54P initS6K
                     S6Ka = ------------------------------------------
                            TORC1a r54P + r54rP r54P2 + r54rP TORC1a

>
# Therefore this can be written as
#
#               TORC1a + B
# S6Ka =   ------------------
#           C * TORC1a + D
>
#Lets do X4EBPa
>
> sol := solve({
>     X4EBPa * routX4EBPaP = routX4EBPaPRev + r62P * Foxoa / (Foxoa + r62P2)
>     },{X4EBPa}):
> sol[1];
                  routX4EBPaPRev Foxoa + routX4EBPaPRev r62P2 + r62P Foxoa
          X4EBPa = ----------------------------------------------------------
                              (Foxoa + r62P2) routX4EBPaP

#
# Therefore this can be written as
#
#              routX4EBPaPRev              Foxoa
# X4EBPa = ------------------- + ------------------
#              routX4EBPaP            A * Foxoa + B
#
>
#Next is X4EBP
>
> solve(X4EBPa * TORC1a * r56P / (TORC1a + r56P2) = X4EBP * r62rP, X4EBP);
                                      X4EBPa TORC1a r56P
                                   -------------------------
                                   (TORC1a + r56P2) r62rP

#
# Therefore this can be written as
#
#             X4EBP * TORC1a * A
# X4EBP = ---------------------------
#                    TORC1a + B
```

```
#
>
#Last is TORC2a
>
> sol := solve({
>     TORC2a * S6Ka * r61P / (S6Ka + r61P2) = TORC2 * InR2a * rTP
>          / (InR2a + rTP2),
>     TORC2a + TORC2 = initTORC2
>     },{TORC2a,TORC2}):
>
> sol[2];
                          InR2a rTP initTORC2 (S6Ka + r61P2)
  TORC2a = ------------------------------------------------------------------------
           S6Ka r61P InR2a + InR2a rTP S6Ka + InR2a rTP r61P2 + S6Ka r61P rTP2

#
# Therefore this can be written as
#
#                            InR2a * (S6Ka + A)
#  TORC2a =
      ------------------------------------------------------------------
#           B * S6Ka * InR2a + C * InR2a + D * S6Ka
#
>
>
# Some Helper functions
#   PKBFlowT:   The induced flow through the PKB system by different TORC1a
#               level.
#   PKBFlow:    The flow induced by the PKB system.
>
> PKBFlowT := (x,START, END, LINEAR1, LINEAR2, CONST)
>            ->(END * x^2 + LINEAR1 * x + START * CONST)
>            / (x ^2 + LINEAR2 * x + CONST):
> PKBFlow  := (x, y, START, END, LINEAR1, LINEAR2, CONST, SPEED, OUT, IN)
>            -> IN / ( y / (y + SPEED)
>            * PKBFlowT(x,START, END, LINEAR1, LINEAR2, CONST) + OUT):
>
# The level of different proteins given the input and some parameters
> InR2a  := (Ins)           -> initInRX / 2 * Ins / (Ins + InR2aSpeed):
> PKBI   := (InR2a)         -> (p1 * InR2a + routPKBIPRev)
>                               / (p2 * InR2a + routPKBIP):
>
> Lobea  := (PKBI, TORC2a)  -> PKBFlow(TORC2a, (PKBINoIns - PKBI), l1, l2, l3,
>                               l4, l5, l6,routLobeaP, routLobeaPRev):
> TSCa   := (PKBI, TORC2a)  -> PKBFlow(TORC2a, (PKBINoIns - PKBI), t1, t2, t3,
>                               t4, t5, t6,routTSCaP, routTSCaPRev):
> Foxoa  := (PKBI, TORC2a)  -> PKBFlow(TORC2a, (PKBINoIns - PKBI), f1, f2, f3,
>                                f4, f5, f6, routFoxoaP, routFoxoaPRev):
> TORC1a := (Lobea)         -> (TORC1aInfLobea * Lobea^2 + t11 * Lobea
>                                + TORC1aNoLobea * t12)
>                                / (Lobea^2 + t13 * Lobea + t12):
> S6Ka   := (TORC1a)        -> (s1 + TORC1a) / (s2 * TORC1a + s3):
> X4EBPa := (Foxoa)         -> routX4EBPaPRev / routX4EBPaP + x1 * Foxoa
>                                / (Foxoa + x2):
> X4EBP  := (TORC1a, X4EBPa)-> X4EBPa * TORC1a * x3 / (TORC1a + x4):
> TORC2a := (S6Ka, InR2a)   -> InR2a * (S6K + t21)
>                               / (t22 * S6Ka * InR2a + t23 * S6Ka + t24 *
  InR2a):
>
>
>
```

```
>
>
>
# Compute scale if no ins
> PKBINoIns     := PKBIMaxIns   * PKBIInsFac:
> FoxoaNoIns    := FoxoaMaxIns  * FoxoaInsFac:
> LobeaNoIns    := LobeaMaxIns  * LobeaInsFac:
> TSCaNoIns     := TSCaMaxIns   * TSCaInsFac:
> TORC1aNoIns   := TORC1aMaxIns * TORC1aInsFac:
> X4EBPaNoIns   := X4EBPaMaxIns * X4EBPaInsFac:
> X4EBPNoIns    := X4EBPMaxIns  * X4EBPInsFac:
> S6KaNoIns     := S6KaMaxIns   * S6KaInsFac:
> TORC2aNoIns   := TORC2aMaxIns * TORC2aInsFac:
>
#Parameters if maximal insulin is present
> initInRX        := 100:
> PKBIMaxIns      := 90:
> FoxoaMaxIns     :=   0.5:
> LobeaMaxIns     :=  10:
> TSCaMaxIns      :=  10:
> TORC1aMaxIns    := 100:
> X4EBPMaxIns     := 101:
> X4EBPaMaxIns    :=  10:
> S6KaMaxIns      := 100:
> TORC2aMaxIns    :=  40:
>
# Give the factor if no insulin is present
> PKBIInsFac    :=   1.111:
> FoxoaInsFac   := 200:
> LobeaInsFac   :=  10:
> TSCaInsFac    :=  10:
> TORC1aInsFac :=    0.01:
> S6KaInsFac    :=    0.02:
> TORC2aInsFac :=    0:
>
#Set up IN/Out reactions
> routLobeaPRev  := 100:
> routFoxoaPRev  := 100:
> routTSCaPRev   := 100:
> routX4EBPaPRev := 100:
> routPKBIPRev   := 100:
>
#Parameters that regulate the variability of the system for
#different insulin inputs
> InR2aSpeed  := 100:
> p2          := 0.01:
> t11         := 1:
> t12         := 100:
> t13         := 100:
> x2          := 1:
> t22         := 20:
> t23         := 1:
> t24         := 20:
> s3          := 1:
>
#Parameters for the PKBI System
> l2  := 0.5 * l1:
> l3  := 0.2:
> l4  := 10:
> l5  := 1:
> l6 := 10:
```

```
> t2 := 1.001 * t1:
> t3 := 0.5:
> t4 := 10:
> t5 := 1:
> t6 := 10:
> f2 := 1.001 * f1:
> f3 := 0.5:
> f4 := 10:
> f5 := 1:
> f6 := 10:
>
#Set up parameters to fulfill scale
> assign(solve({
>         Lobea(PKBIMaxIns, TORC2aMaxIns) = LobeaMaxIns,
>         Lobea(PKBINoIns, TORC2aNoIns) = LobeaNoIns},
>     {l1, routLobeaP})):
> assign(solve({
>         Foxoa(PKBIMaxIns, TORC2aMaxIns) = FoxoaMaxIns,
>         Foxoa(PKBINoIns, TORC2aNoIns) = FoxoaNoIns},
>     {f1, routFoxoaP})):
> assign(solve({
>         TSCa(PKBIMaxIns, TORC2aMaxIns) = TSCaMaxIns,
>         TSCa(PKBINoIns, TORC2aNoIns) = TSCaNoIns},
>     {t1, routTSCaP})):
> assign(solve({
>         PKBI(0) = PKBINoIns,
>         PKBI(initInRX/2) = PKBIMaxIns},
>     {routPKBIP,p1})):
> assign(solve({
>         TORC1a(LobeaMaxIns) = TORC1aMaxIns,
>         TORC1a(LobeaNoIns) = TORC1aNoIns},
>     {TORC1aInfLobea,TORC1aNoLobea}));
> assign(solve({
>         S6Ka(TORC1aMaxIns) = S6KaMaxIns,
>         S6Ka(TORC1aNoIns) = S6KaNoIns},
>     {s1,s2}));
> assign(solve({
>         TORC2a(S6KaMaxIns, initInRX/2) = TORC2aMaxIns},
>     {t21}));
>
#Enforce Constraints
> assign(solve({
>         X4EBPa(FoxoaMaxIns) = X4EBPaMaxIns,
>         X4EBPa(FoxoaNoIns) = X4EBPaNoIns},
>     {x1,routX4EBPaP}));
> assign(solve({
>         X4EBP(TORC1aMaxIns, X4EBPaMaxIns) = X4EBPMaxIns,
>         X4EBP(TORC1aNoIns, X4EBPaNoIns) = X4EBPNoIns},
>     {x3,x4}));
>
> sol := solve({
>         routX4EBPaP>0,x1 > 0},
>     {X4EBPaInsFac})
>     assuming (routX4EBPaPRev > 0, FoxoaInsFac > 1, x2>0,
>         X4EBPaMaxIns > 0, FoxoaMaxIns >0):
> X4EBPaInsFac := 0.99 * rhs(sol[2]):
>
> sol := solve({x3 > 0, x4> 0},{X4EBPInsFac}):
> X4EBPInsFac := 4 * lhs(sol[1]):
>
>
```

```
>
>
> getVals := proc(InsLevel)::list;
>     local sol, InR2aLevel, PKBILevel;
>     description "NONE";
>     InR2aLevel := InR2a (InsLevel);
>     PKBILevel := PKBI(InR2aLevel);
>     sol := solve({
>         LobeaLevel  = Lobea  (PKBILevel, TORC2aLevel),
>         TSCaLevel   = TSCa   (PKBILevel, TORC2aLevel),
>         FoxoaLevel  = Foxoa  (PKBILevel, TORC2aLevel),
>         TORC1aLevel = TORC1a (LobeaLevel),
>         S6KaLevel   = S6Ka   (TORC1aLevel),
>         X4EBPaLevel = X4EBPa (FoxoaLevel),
>         X4EBPLevel  = X4EBP  (TORC1aLevel, X4EBPaLevel),
>         TORC2aLevel = TORC2a (S6KaLevel, InR2aLevel),
>         FoxoaLevel > 0
>         },
>         {FoxoaLevel, LobeaLevel, S6KaLevel,
>         TORC1aLevel, TORC2aLevel, TSCaLevel, X4EBPLevel, X4EBPaLevel});
>         return [InsLevel, rhs(sol[1]), rhs(sol[2]), rhs(sol[3]),
>         rhs(sol[4]), rhs(sol[5]), rhs(sol[6]), rhs(sol[7]),
>         rhs(sol[8])]:
> end proc;
getVals := proc(InsLevel)::list;
local sol, InR2aLevel, PKBILevel;
description "NONE";
    InR2aLevel := InR2a(InsLevel);
    PKBILevel := PKBI(InR2aLevel);
    sol := solve({FoxoaLevel = Foxoa(PKBILevel, TORC2aLevel),
        LobeaLevel = Lobea(PKBILevel, TORC2aLevel),
        S6KaLevel = S6Ka(TORC1aLevel), TORC1aLevel = TORC1a(LobeaLevel),
        TORC2aLevel = TORC2a(S6KaLevel, InR2aLevel),
        TSCaLevel = TSCa(PKBILevel, TORC2aLevel),
        X4EBPLevel = X4EBP(TORC1aLevel, X4EBPaLevel),
        X4EBPaLevel = X4EBPa(FoxoaLevel), 0 < FoxoaLevel}, {FoxoaLevel,
        LobeaLevel, S6KaLevel, TORC1aLevel, TORC2aLevel, TSCaLevel,
        X4EBPLevel, X4EBPaLevel});
    return [InsLevel, rhs(sol[1]), rhs(sol[2]), rhs(sol[3]), rhs(sol[4]),
        rhs(sol[5]), rhs(sol[6]), rhs(sol[7]), rhs(sol[8])]
end proc

>
> fffwith(linalg):
> minimalEvaluationPower := -2:
> maximalEvaluationPower := +5:
> evaluationPowerStep := 0.25:
> evalueationPoints :=
>     [$(minimalEvaluationPower/evaluationPowerStep
>         ..maximalEvaluationPower/evaluationPowerStep)]
>     * evaluationPowerStep:
> evaluationPoints := [0, evalueationPoints[]]:
> results := []:
> for i in evalueationPoints do
>     10^i;
>     results := [op(results), op(getVals(10^i))]:
> end:
memory used=91.7MB, alloc=72.2MB, time=1.62
memory used=168.0MB, alloc=72.2MB, time=3.31
memory used=244.3MB, alloc=72.2MB, time=5.02
>
```

```
> res := Matrix(nops(evalueationPoints),9,results):
> ExportMatrix("../simulation/resultsSimple.mat", res, target = Matlab):
>
> quit
memory used=257.1MB, alloc=72.2MB, time=5.26
```

B. The Model

../TmpModel.psc

```
# Generated by PySCeS 0.8.0 (2012-03-01 05:37)

# Keywords
Description:
Modelname: test
Output_In_Conc: True
Species_In_Conc: True

# GlobalUnitDefinitions
UnitVolume: litre, 1.0, 0, 1
UnitArea: metre, 1.0, 0, 2
UnitLength: metre, 1.0, 0, 1
UnitSubstance: mole, 1.0, 0, 1
UnitTime: second, 1.0, 0, 1

# Compartments
Compartment: all, 1.0, 3

# Reactions
rinRheba@all:
    $pool > Rheba
    rinRheba_rinRhebaP

routPI3Ka@all:
    PI3Ka > $pool
    PI3Ka*routPI3Ka_routPI3KaP

r1r@all:
    InR2 > {2.0}InR1
    InR2*r1r_r1rP

r9@all:
    PIP2 > PIP3
    PI3Ka*PIP2*r9_r9P

r45@all:
    TSCa > TSC
    PKBaa*maxATP*TSCa*r45_r45P

r62@all:
    $pool > X4EBPa
    Foxoa*r62_r62P

r47@all:
    Rheba > Rheb
    Rheba*TSCa*r47_r47P

r41@all:
    TSCa > TSC
    PKBa1*maxATP*TSCa*r41_r41P
```

```
r43@all:
    TSCa > TSC
    PKBa2*maxATP*TSCa*r43_r43P

r27@all:
    Foxoa > Foxo
    PKBa1*maxATP*Foxoa*r27_r27P

routFoxoa@all:
    Foxoa > $pool
    Foxoa*routFoxoa_routFoxoaP

routRheba@all:
    Rheba > $pool
    Rheba*routRheba_routRhebaP

r24@all:
    PKBa1 > PKBaa
    TORC2a*maxATP*PKBa1*r24_r24P

r49@all:
    Rheb > Rheba
    Rheb*maxGEF*r49_r49P

rinp110@all:
    $pool > p110
    rinp110_rinp110P

r21@all:
    PKBa2 > PKBaa
    maxATP*maxPDK1*PKBa2*r21_r21P

r54r@all:
    S6Ka > S6K
    S6Ka*r54r_r54rP

r51@all:
    TORC1a > TORC1
    Lobea*TORC1a*r51_r51P

routLobea@all:
    Lobea > $pool
    Lobea*routLobea_routLobeaP

rinFoxoa@all:
    $pool > Foxoa
    rinFoxoa_rinFoxoaP

rinPIP2@all:
    $pool > PIP2
    rinPIP2_rinPIP2P

r56@all:
    X4EBPa > X4EBP
    TORC1a*maxATP*X4EBPa*r56_r56P

rinTORC1@all:
    $pool > TORC1
    rinTORC1_rinTORC1P

rinLobea@all:
        $pool > Lobea
        rinLobea_rinLobeaP

    rinp60@all:
        $pool > p60
        rinp60_rinp60P

    routPIP2@all:
        PIP2 > $pool
        PIP2*routPIP2_routPIP2P

    r7r@all:
        PI3Ka > PI3K + Chicoa
        PI3Ka*r7r_r7rP

    routTSCa@all:
        TSCa > $pool
        TSCa*routTSCa_routTSCaP

    routX4EBPa@all:
        X4EBPa > $pool
        X4EBPa*routX4EBPa_routX4EBPaP

    routPKBaa@all:
        PKBaa > $pool
        PKBaa*routPKBaa_routPKBaaP

    routTSC@all:
        TSC > $pool
        TSC*routTSC_routTSCP

    r50r@all:
        TORC1a > TORC1
        Rheba*TORC1a*r50r_r50rP

    routChico@all:
        Chico > $pool
        Chico*routChico_routChicoP

    r62r@all:
        X4EBP > X4EBPa
        X4EBP*r62r_r62rP

    r15@all:
        PKB > PKBa1
        maxATP*maxPDK1*PKB*r15_r15P

    r12@all:
        PKBI > PKB
        PKBI*PIP3*r12_r12P

    r11@all:
        PIP3 > PIP2
        maxPTEN*PIP3*r11_r11P

    rTORC2a@all:
        TORC2 > TORC2a
        InR2a*TORC2*rTORC2a_rTORC2aP

    routLobe@all:
        Lobe > $pool
    Lobe*routLobe_routLobeP

r18@all:
    PKB > PKBa2
    TORC2a*maxATP*PKB*r18_r18P

r6r@all:
    PI3K > p60 + p110
    PI3K*r6r_r6rP

r39@all:
    Lobea > Lobe
    PKBaa*maxATP*Lobea*r39_r39P

r35@all:
    Lobea > Lobe
    PKBa1*maxATP*Lobea*r35_r35P

r50@all:
    TORC1 > TORC1a
    TORC1*Rheba*r50_r50P

r37@all:
    Lobea > Lobe
    PKBa2*maxATP*Lobea*r37_r37P

r30@all:
    Foxoa > Foxo
    PKBa2*maxATP*Foxoa*r30_r30P

r2r@all:
    InR2a > InR2
    InR2a*r2r_r2rP

r54@all:
    S6K > S6Ka
    TORC1a*maxATP*S6K*r54_r54P

r33@all:
    Foxoa > Foxo
    PKBaa*maxATP*Foxoa*r33_r33P

r5@all:
    Chico > Chicoa
    InR2a*maxATP*Chico*r5_r5P

r6@all:
    p60 + p110 > PI3K
    p60*p110*r6_r6P

r7@all:
    PI3K + Chicoa > PI3Ka
    Chicoa*PI3K*r7_r7P

r1@all:
    {2.0}InR1 > InR2
    InR1*InR1*r1_r1P

r2@all:
    InR2 > InR2a
    InR2*maxIns*r2_r2P
```

```

rinChico@all:
    $pool > Chico
    rinChico_rinChicoP

rinX4EBPa@all:
    $pool > X4EBPa
    rinX4EBPa_rinX4EBPaP

routFoxo@all:
    Foxo > $pool
    Foxo*routFoxo_routFoxoP

r61@all:
    TORC2a > TORC2
    S6Ka*maxATP*TORC2a*r61_r61P

rinTSCa@all:
    $pool > TSCa
    rinTSCa_rinTSCaP

routTORC1@all:
    TORC1 > $pool
    TORC1*routTORC1_routTORC1P

routPKBI@all:
    PKBI > $pool
    PKBI*routPKBI_routPKBIP

rinPKBI@all:
    $pool > PKBI
    rinPKBI_rinPKBIP

routp60@all:
    p60 > $pool
    p60*routp60_routp60P

routp110@all:
    p110 > $pool
    p110*routp110_routp110P

routPKBa1@all:
    PKBa1 > $pool
    PKBa1*routPKBa1_routPKBa1P

r51r@all:
    TORC1 > TORC1a
    Lobea*TORC1*r51r_r51rP

# Fixed species

# Variable species
Rheba@all = 145
PI3Ka@all = 3
InR2@all = 1
InR1@all = 0
PIP2@all = 53
PIP3@all = 75
TSCa@all = 28
TSC@all = 85
X4EBPa@all = 60
```

```
Rheb@all = 38
Foxoa@all = 40
Foxo@all = 108
PKBa1@all = 51
PKBaa@all = 1
p110@all = 74
PKBa2@all = 99
S6Ka@all = 53
S6K@all = 47
TORC1a@all = 14
TORC1@all = 85
Lobea@all = 1
X4EBP@all = 82
p60@all = 75
PI3K@all = 115
Chicoa@all = 0
Chico@all = 8
PKB@all = 47
PKBI@all = 11
TORC2@all = 3
TORC2a@all = 97
Lobe@all = 102
InR2a@all = 54

# Parameters
rinRheba_rinRhebaP = 1.0
routPI3Ka_routPI3KaP = 1.0
r1r_r1rP = 0.049261
r9_r9P = 0.1
r45_r45P = 0.01
maxATP = 1.0
r62_r62P = 0.088119
r47_r47P = 10.0
r41_r41P = 0.01
r43_r43P = 0.01
r27_r27P = 0.01
routFoxoa_routFoxoaP = 0.67033
routRheba_routRhebaP = 0.01
r24_r24P = 0.00010111
r49_r49P = 10.0
maxGEF = 100.0
rinp110_rinp110P = 1.0
r21_r21P = 0.0005
maxPDK1 = 100.0
r54r_r54rP = 90.0
r51_r51P = 22.099
routLobea_routLobeaP = 0.020313
rinFoxoa_rinFoxoaP = 67.703
rinPIP2_rinPIP2P = 1.0
r56_r56P = 10.0
rinTORC1_rinTORC1P = 1.0
rinLobea_rinLobeaP = 2.0313
rinp60_rinp60P = 1.0
routPIP2_routPIP2P = 0.01
r7r_r7rP = 0.036666
routTSCa_routTSCaP = 0.67033
routX4EBPa_routX4EBPaP = 0.1
routPKBaa_routPKBaaP = 5.4545
routTSC_routTSCP = 0.50275
r50r_r50rP = 1.0
routChico_routChicoP = 0.01
```

```
r62r_r62rP = 97.727
r15_r15P = 0.001
r12_r12P = 0.011
r11_r11P = 0.00090909
maxPTEN = 100.0
rTORC2a_rTORC2aP = 10.0
routLobe_routLobeP = 0.02011
r18_r18P = 0.0010111
r6r_r6rP = 1.1
r39_r39P = 0.01
r35_r35P = 0.01
r50_r50P = 0.363
r37_r37P = 0.01
r30_r30P = 0.01
r2r_r2rP = 0.1
r54_r54P = 10.0
r33_r33P = 0.01
r5_r5P = 0.00203
r6_r6P = 0.022
r7_r7P = 1.0
r1_r1P = 0.1
r2_r2P = 0.1
maxIns = 100.0
rinChico_rinChicoP = 1.0
rinX4EBPa_rinX4EBPaP = 1.0
routFoxo_routFoxoP = 0.50777
r61_r61P = 0.21894
rinTSCa_rinTSCaP = 67.033
routTORC1_routTORC1P = 0.01
routPKBI_routPKBIP = 0.11
rinPKBI_rinPKBIP = 11.0
routp60_routp60P = 0.01
routp110_routp110P = 0.01
routPKBa1_routPKBa1P = 0.1
r51r_r51rP = 0.33
```

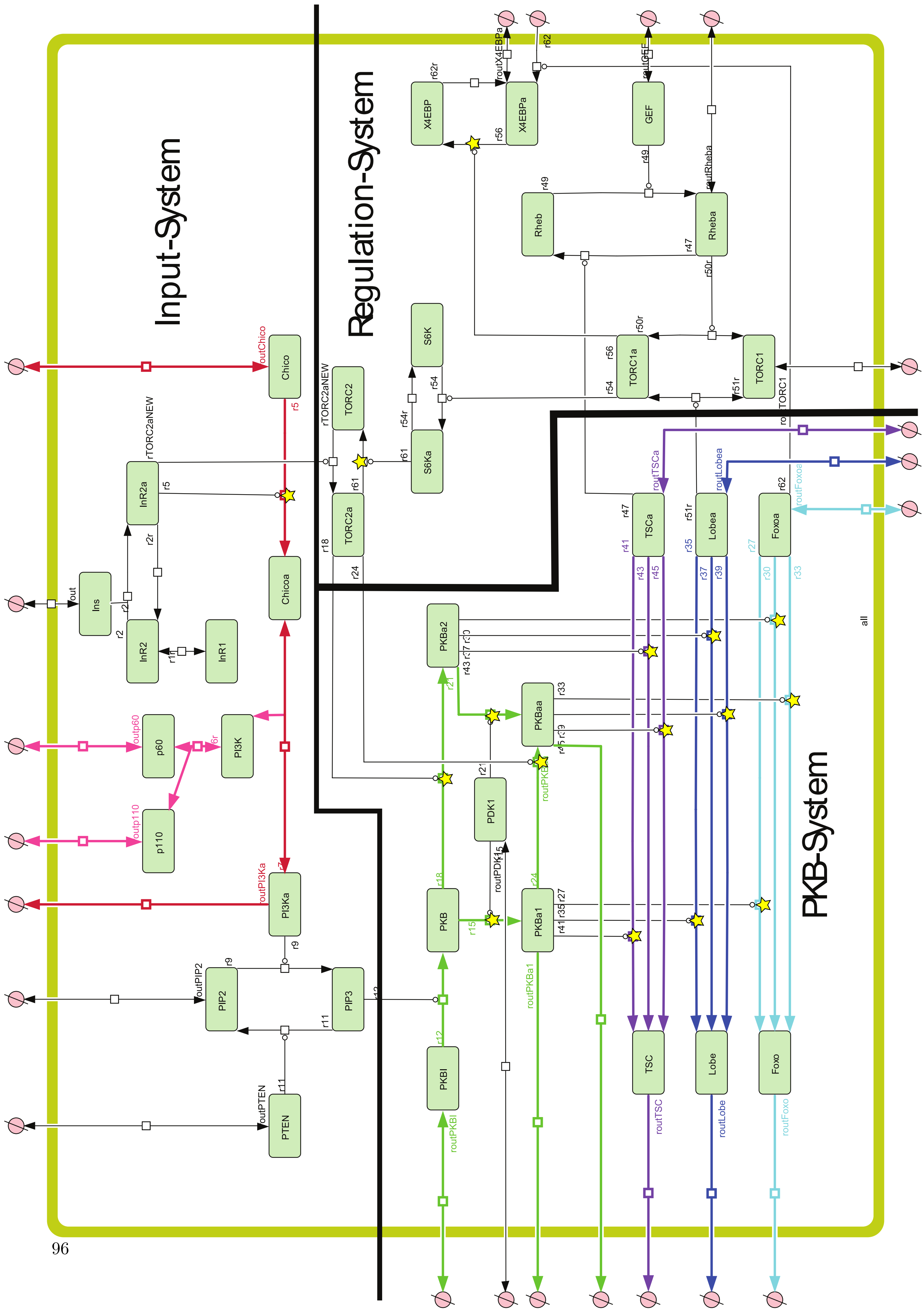


Figure B.1.: The model without ATP and ADP. A yellow star means that this reaction reduces ATP to ADP

List of Figures

Bibliography

[1] A. Auger, P. Chatelain, and P. Koumoutsakos. R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps. *Journal of Chemical Physics*, 125(8), 2006. Cited By (since 1996): 27.

[2] Julie Baker, Jeh-Ping Liu, Elizabeth J. Robertson, and Argiris Efstratiadis. Role of insulin-like growth factors in embryonic and postnatal growth. *Cell*, 75(1):73 – 82, 1993.

[3] Gabriel Ciobanu, Mario J. Pérez-Jiménez, and Gheorghe Paun, editors. *Applications of Membrane Computing*. Natural Computing Series. Springer, 2006.

[4] Alan Fersht. *Structure and Mechanism in Protein Science: A Guide to Enzyme Catalysis and Protein Folding*. W. H. Freeman, 1st edition, September 1998.

[5] Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Hiroaki Kitano, and Norihiro Kikuchi. CellDesigner: a modeling tool for biochemical networks. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1707–1712. Winter Simulation Conference, 2006.

[6] Xinsheng Gao and Duojia Pan. Tsc1 and tsc2 tumor suppressors antagonize insulin signaling in cell growth.

[7] Attila Garami, Fried J.T Zwartkruis, Takahiro Nobukuni, Manel Joaquin, Marta Roccio, Hugo Stocker, Sara C Kozma, Ernst Hafen, Johannes L Bos, and George Thomas. Insulin activation of rheb, a mediator of mtor/s6k/4e-bp signaling, is inhibited by tsc1 and 2. *Molecular Cell*, 11(6):1457 – 1466, 2003.

[8] M. A. Gibson and J. Bruck. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000.

[9] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001. Cited By (since 1996): 443.

[10] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

[11] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[12] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novre, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson,

P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. B. M. L. Forum. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, March 2003.

[13] Kenneth A. Johnson and Roger S. Goody. The original michaelis constant: Translation of the 1913 michaelismenten paper. *Biochemistry*, 50(39):8264–8269, 2011.

[14] Timo Maarleveld, Brett Olivier, and Frank Bruggeman. Stochpy. http://stompy.sourceforge.net.

[15] Rajesh Ramaswamy. Insulin pathway modeling, 2011.

[16] Rajesh Ramaswamy, Nélido G. Segredo, and Ivo F. Sbalzarini. A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. *The Journal of Chemical Physics*, 130(24):244104+, 2009.

[17] Ivo F. Sbalzarini. Spatiotemporal modeling and simulation, 2009.

[18] Henning Schmidt and Mats Jirstrand. Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics*, 22(4):514–515, February 2006.

[19] Daniel T. and Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404 – 425, 1992.

[20] D.C. Wilcox. *Basic Fluid Mechanics*. DCW Industries, Incorporated, 2010.