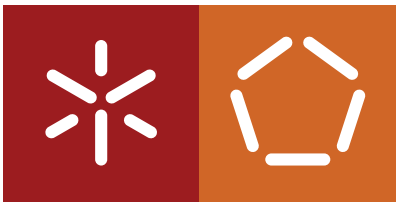
Universidade do Minho
Escola de Engenharia

Pedro Tiago Evangelista

Novel approaches for dynamic modelling of E. coli and their application in Metabolic Engineering

outubro de 2016

Novel approaches for dynamic modelling of E. coli and their application in Metabolic Engineering
Pedro Tiago Evangelista
Uminho | 2016



Universidade do Minho
Escola de Engenharia

Pedro Tiago Evangelista

Novel approaches for dynamic modelling of *E. coli* and their application in Metabolic Engineering

PhD thesis in Bioengineering

This work was realized under the supervision of:

Prof. Isabel Cristina de Almeida Pereira da Rocha

Prof. Bruce Tidor

Prof. Miguel Francisco de Almeida Pereira da Rocha

October 2016

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, October 5, 2016

Pedro Tiago Evangelista

ACKNOWLEDGEMENTS/AGRADECIMENTOS

Quero fazer um agradecimento muito especial aos professores Isabel Rocha e Miguel Rocha, que além de terem sido meus orientadores neste doutoramento,foram antes demais meus amigos e sempre me apoiaram quando precisei. Sem eles esta viagem não teria sido possível.

Agradeço a todos os que fazem parte da família que é a Silicolife, pela amizade, sentimento de irmandade e pelo apoio que sempre me deram (em especial aqueles que sofreram quando os obriguei a ouvir música africana).

À minha familia, em particular à minha mãe, ao meu pai, à minha avó, e à minha namorada, um grande muito obrigado.

A special thanks to Professor Bruce Tidor that guided me during this work. I also want to thank all the members of the Tidor Lab during my stay at MIT: Nermala Paudel, Yuanyuan Cui, Tina Toni, David Hagen, Ishan Patel, Brian Bonk and Raja R Srinivas.

Gostava de terminar agradecendo a todas aqueles que não nomeei aqui explicitamente, mas que sempre me incentivaram.

A todos o meu mais sincero agradecimento.

ABSTRACT

Novel approaches for dynamic modelling of *E. coli* and their application in Metabolic Engineering

One of the trends of modern societies is the replacement of chemical processes by biochemical ones, with new compounds being synthesized by engineered microorganisms, while some waste products are also being degraded by biotechnological means. Biotechnology holds the promise of creating a more profitable and environmental friendly industry, with a reduced number of waste products, when contrasted with the traditional chemical industry.

However, in an era in which genomes are sequenced at a faster pace than ever before, and with the advent omic measurements, this information is not directly translated into the targeted design of new microorganisms, or biological processes. These experimental data in isolation do not explain how the different cell constituents interact. Reductionist approaches that dominated science in the last century study cellular entities in isolation as separate chunks, without taking into consideration interactions with other molecules. This leads to an incomplete view of biological processes, which compromises the development of new knowledge.

To overcome these hurdles, a formal systems approach to Biology has been surging in the last thirty years. Systems biology can be defined as the conjugation of different fields (such as Mathematics, Computer Science, Biology),

to describe formally and non-ambiguously the behavior of the different cellular systems and their interactions, using to models and simulations. Metabolic Engineering takes advantage of these formal specifications, using mathematically based methods to derive strategies to optimize the microbial metabolism, in order to achieve a desired goal, such as the increase of the production of a relevant industrial compound. In this work, we develop a mechanistic dynamic model based on ordinary differential equations, comprised by elementary mass action descriptions of each reaction, from an existing model of Escherichia coli in the literature. We also explore different calibration processes for these reaction descriptions.

We also contribute to the field of strain design by utilizing evolutionary algorithms with a new representation scheme that allows to search for enzyme modulations, in continuous or discrete scales, as well as reaction knockouts, in existing dynamic metabolic models, aiming at the maximization of product yields.

In the bioprocess optimization field, we extended the Dynamic Flux Balance Analysis formulation to incorporate the possibility to simulate fed-batch bioprocesses. This formulation is also enhanced with methods that possess the capacity to design feed profiles to attain a specific goal, such as maximizing the bioprocess yield or productivity.

All the developed methods involved some form of sensitivity and identifiability analysis, to identify how model outputs are affected by their parameters.

All the work was constructed under a modular software framework (developed during this thesis), that permits the interaction of distinct algorithms and languages, being a flexible tool to utilize in a cluster environment. The framework is available as an open-source software package, and has appeal to systems biologists describing biological processes with ordinary differential equations.

RESUMO

Novel approaches for dynamic modelling of *E. coli* and their application in Metabolic Engineering

Uma das tendências na nossa sociedade actual é a substituição de processos químicos por processos bioquímicos, e a síntese de novos compostos por microrganismos, bem como a degradação de resíduos por meios biotecnológicos. A Biotecnologia tem, assim, a promessa de criar uma indústria mais rentavél e mais amiga do ambiente, com um número reduzido de resíduos, contrastando com a indústria química.

No entanto, numa era em que os genomas são sequenciados a um ritmo nunca visto, assim como as medições de dados ómicos, esta informação não é diretamente traduzida no desenho de estirpes microbianas ou processos biológicos. Estes dados experimentais em isolamento não explicam como os diferentes componentes celulares interagem. As abordagens reducionistas que dominaram a ciência no século passado, estudam os constituintes celulares em isolamento, como pedaços isolados, sem tomar em consideração as interacções com outras moléculas, o que traduz uma visão incompleta do mundo, que compromete o desenvolvimento de novo conhecimento.

Para superar estes obstáculos, uma nova abordagem à Biologia tem emergido nos últimos trinta anos. A Biologia de Sistemas pode ser definida como a conjugação de diferentes áreas (como a Matemática, Ciência da Computação, Bi-

ologia), para descrever formalmente e de forma não ambígua o comportamento dos diferentes sistemas celulares e as suas interações utilizando a modelação. A Engenharia Metabólica tira partido destas especificações formais, utilizando métodos matemáticos para derivar estratégias tendo em vista a optimização do metabolismo de microrganismos, de forma a atingir um objetivo definido como por exemplo o aumento da produção de um composto relevante a nível industrial.

Neste trabalho, desenvolvemos um modelo dinâmico mecanístico baseado em equações diferenciais ordinárias, composto por descrições ação de massas elementares para cada reacção, partindo de um modelo já existente da *Escherichia coli* na literatura.

Utilizamos também algoritmos evolucionários com um novo esquema de representação que permite pesquisar por modulações enzimáticas, numa escala contínua ou discreta, assim como eliminar reações em modelos metabólicos existentes de forma a maximizar o rendimento ou a produtividade.

Todos os métodos desenvolvidos envolveram alguma forma de análise de sensibilidade ou identifiabilidade, de forma a verificar como as saídas do modelo são afetados pelos parâmetros.

Todo o trabalho foi construído de acordo com uma plataforma de software modular (desenvolvida durante esta tese) que permite a interação de algoritmos e linguagens distintos, sendo uma ferramenta flexível para utilizar em ambientes de cluster. A plataforma encontra-se disponível como um pacote de software de código aberto e tem utilidade para biólogos de sistemas que pretendam descrever processos com equações diferencias ordinárias.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AOT one-factor-at-the-time

ARL Aggregated rate law

AST Abstract syntax tree

BNF Backus Naur form

DE Differential Evolution

DFBA Dynamic Flux Balance Analysis

DSL Domain specific language

EA Evolutionary Algorithm

EC Evolutionary Computation

EE Elementary Effect

FAST Fourier amplitude sensitivity test

FBA Flux Balance Analysis

FIM Fisher Information Matrix

fPCA functional Principal Component Analysis

GA Genetic Algorithm

GE Grammatical Evolution

GMA Generalized mass-action

GSA Global Sensitivity Analysis

HDE Hybrid Differential Evolution

HDMR High-dimensional model representation

KA King-Altman

LHCS Latin Hyper Cube Sampling

MA Mass-action

MARL Mass-Action rate law

MC Monte Carlo

MCA Metabolic Control Analysis

ME Metabolic Engineering

MFA Metabolic Flux Analysis

MILP Mixed Integer Linear Programming

MLE Maximum Likelihood estimation

MINLP Mixed Integer Non-Linear Progamming

MOMA Minimization of Metabolic Adjustment

MPSA Multi-Parametric Sensitivity Analysis

MSAE Mean square average error

MWC Monod-Wyman-Changeux

NT non-terminal symbol set

ODE Ordinary Differential Equation

PAM Polynomial Approximation Method

PC Principal component

PCA Pricinpal Component Analysis

PDE Partial Differential Equation

PFBA Parsimonious Flux Balance Analysis

QRS-HDMR Quasi Random Sampling High-dimensional model representation

ROOM Regulatory On/Off Minimization

RS Random Sampling

SA Sensitivity Analysis

SB Systems Biology

SBML Systems Biology Modeling Language

SOA Static optimization algorithm

T Terminal symbol set

UA Uncertainty analysis

WAR Weighted Average of Relative Sensitivities

CHAPTER 1

INTRODUCTION

Dynamic metabolic models play an increasing role, in the crafting of engineering strategies to redirect microorganism metabolic necessities, for the production of pertinent industrial compounds. Nonetheless, there is the need to construct finer grain mechanistic representations of metabolism, able to capture a wide range of microorganisms physiological behavior.

This thesis, is focused on the development of a mechanistic model of central carbon metabolism of *Escherichia coli* and the respective calibration and identifiability procedures, having as basis mass-action elementary reaction descriptions for all enzymatic mechanisms. Within the same context, the design of Metabolic Engineering strategies utilizing Evolutionary Algorithms, is also explored. Extensions to other existing formalisms, like Dynamic Flux Balance Analysis, that integrate the bioprocess models with microorganism stoichiometric models, are also formulated. The software developed during this work, is made available as an open-source package for the Systems Biology community.

The chapter provides an overview of the work, as well as its motivation, goals and structure of this thesis.

1.1 Context

One of the main goals of Bioengineering has been the engineering of biological systems at microbial level to construct cell factories to produce relevant compounds, or to degrade undesirable products, as other man engineered complex systems, such as cars, planes, buildings or industrial facilities. In this context, Biotechnology can be defined as the utilization of living systems (often microorganisms) or their parts to attain specific goals. This field has a wide range of applications and to help categorize the main areas of operation, a color scheme was conceived (becoming a *de facto* standard in the community) [1]:

- White biotechnology, also known as industrial biotechnology, involves all the bioprocesses utilizing microorganisms to produce relevant compounds that are not possible to produce by chemical routes alone or replacing chemical synthesis. This type of biotechnology also incorporates industrial processes that utilize enzymes to catalyze a reaction as part of the production process.

- Green biotechnology is connected with the use of bioengineering in agricultural processes;

- Red biotechnology also known as medical related biotechnology utilizes biological systems to produce or process pharmaceutical products.

On the other hand, Systems Biology (SB) advocates the comprehension of biological systems, recurring to mathematical modeling, to elucidate the function and interactions of distinct cellular entities. Only with precise and formal descriptions of a system inner workings, it is possible to craft engineering strategies to attain specific goals [2]. In industrial biotechnology players, one major

question is how to improve production yields of relevant compounds in an efficient engineering way. In this context, efficient can be defined as utilizing existing knowledge about microbes and the bioprocess to devise a mathematical formulation that is prone to the creation of rational metabolic and bioprocess engineering strategies. These methods should enable the conjugation of cell physiological needs with the devised goals, like the synthesis of new compounds or the overproduction of existing ones.

Mathematical models representing microoganisms try to capture the behavior of cell constituents and the complexity of their interactions (many times in a non-linear fashion). It is important to bear in mind that different types of models assemble information differently and are usually focused in a partial subsystem of the cell. Models possess distinct granularities and depth concerning the scenario being studied. In this work we are interested in modeling the cell metabolism at meso scale for devising ME strategies.

These types of metabolic models can be described recurring only to metabolic topological information (ignoring regulatory and kinetic information), assuming that the system is at steady-state or, if more detailed information about enzymatic kinetics is available, a dynamic version of the system can be constructed. Steady-state models provide useful information concerning the possible states of a biochemical network at steady-state but, if the system is under-determined, it is not possible to identify the true cell state without making strong assumptions (such as the maximization of a biomass flux) that can compromise the result. On the other hand, dynamic metabolic models are commonly modeled as differential equation systems assuming cells behave like small bioreactors due to homogeneity in the cytoplasm and high number of interacting molecules. If these assumptions do not hold, then another formalism has to be applied instead. These models possess a smaller number of steady-states than steady-state under-

determined models and do not require the assumption of an underlying cellular objective.

Nonetheless, the main factors affecting negatively the construction of large scale dynamic models are the lack of mechanistic information concerning biochemical reactions and experimental hindrances such as the incapacity to measure all important metabolites for the parameter calibration. Besides the topological description of the reaction network, initial system concentrations, experimental data regarding fluxes or concentrations, as well as partial knowledge of the kinetics are necessary to build and calibrate a dynamic model.

Broadly, this type of models contains reaction rate laws that can be subdivided in data based approaches and in mechanistic approaches. Data based approaches require data over the whole range of physiological concentrations being studied, although they do not account explicitly for the mechanisms responsible for the observed phenomena. On the other hand, mechanistic based representations require knowledge of the underlying reaction physical process but need less data to perform a calibration of the system. Usually, a model will be a mixture of these types of rate laws depending on the underlying assumptions and information available. It is important to bear in mind that there are no hard boundaries within this categorization.

Reaction rates can be described by aggregated rate laws (ARLs) that encapsulate specific elementary rates in the parameters. Examples of such models include the Central Carbon metabolism of *Escherichia coli* [3], the Central Carbon metabolism of *Saccaromyces cerevisae* [4] or the TCA cycle of *Dictyostelium discoideum* [5].

A more complete version of these rate laws can be constructed by considering all the mechanistic steps and enzymatic complexes formed among all the participating species in the reaction and describing the system by mass-action el-

ementary reactions called Mass-Action Rate Laws (MARLs). This formalization produces a system with more parameters but, in theory, will require less experimental data in a smaller range of physiological values, possessing the capacity to extrapolate outside of this range. In this context, to reduce the overparameterization caused by the explicit representation of the elementary rate reactions, assumptions can be made regarding the behavior of the enzymatic system. Often, it is presumed that certain reaction steps are at rapid equilibrium conditions, or that participating enzyme complexes are at steady-state. Nonetheless, it is important to test their applicability against available experimental data. The rapid equilibrium assumption introduces tighter constraints concerning the reaction mechanism and it may be less applicable in practice.

Another approach is to ignore the underlying reaction mechanism and to utilize an approximate rate laws. Despite of usually possessing a smaller number of parameters than ARLs, these reaction rates are often only valid in the vicinity of the operational state they were calibrated to. The prowess of these rate laws to extrapolation is limited and is dependent on the existence of experimental data.

However, regardless the rate law syntactic format, it is important to assess parameters identifiability, defined as the capacity to regress their values based on perfect data (structural identifiability) and from data corrupted with noise (practical identifiability). If a parameter has structural identifiability problems, it will not be possible to estimate its value from experimental data. On the other hand, practical identifiability issues arise when there is insufficient (in the sense of non-informative) experimental data to calibrate a parameter to a desired level of uncertainty. In the first case, the issue can be overcome by changing the functional structure of enzymatic mechanism representation or fixing some of the parameters if correlated (utilizing domain specific knowledge), while in the second scenario a more well crafted set of experiments may mitigate this hurdle.

Parameter identifiability is tightly linked to sensitivity and uncertainty analyses. More sensitive parameters are easier to estimate, while less sensitive (often coined robust) may possess a negligible impact on the output variable being characterized. These analyses should be performed before regressing the parameters and collecting experimental data (called *a priori* sensitivity analyses) and antecedently to collecting experimental data (experimental design). It is also possible to use existing information and compute these analysis *a posteriori* to characterize the parameters sensitivity/uncertainty and delineate the next research steps.

Robust (in the sense of non-influential) and correlated parameters should be fixed before the regression process. Failure to do so may lead to abortion or lack of convergence in optimization algorithms.

After this process, the goodness of fit should be assessed within the research context as well as the parameter uncertainty levels.

Of particular interest in the scope of this work is the production of amino-acids, namely serine, at industrial biotechnology level [6] from a ME perspective. Serine has industrial interest as a flavor enhancer [7], and in pharmaceutical industry as a potential new drug for Amyotrophic lateral sclerosis [8].

E. coli is one of the most utilized microbial strains at the bio-industrial level due to the cultivation simplicity, low experimental expenses, ease of genetic modification and academic study focus in the past decades. There is a wide range of publicly available databases concentrating relevant biological information for the construction of models such as Eccocy [9] and EcoGene [10]. This microorganism is a prokaryote and therefore has a simpler structure and no cellular compartmentalization, contrarily to eukaryotes (such as *Saccaromyces cerevisiae*). Several industrially relevant compounds are produced by *E. coli* such as biofuels [11], bulk chemicals [12] and amino-acids [13] such as succinic

acid [14].

1.2 Motivation

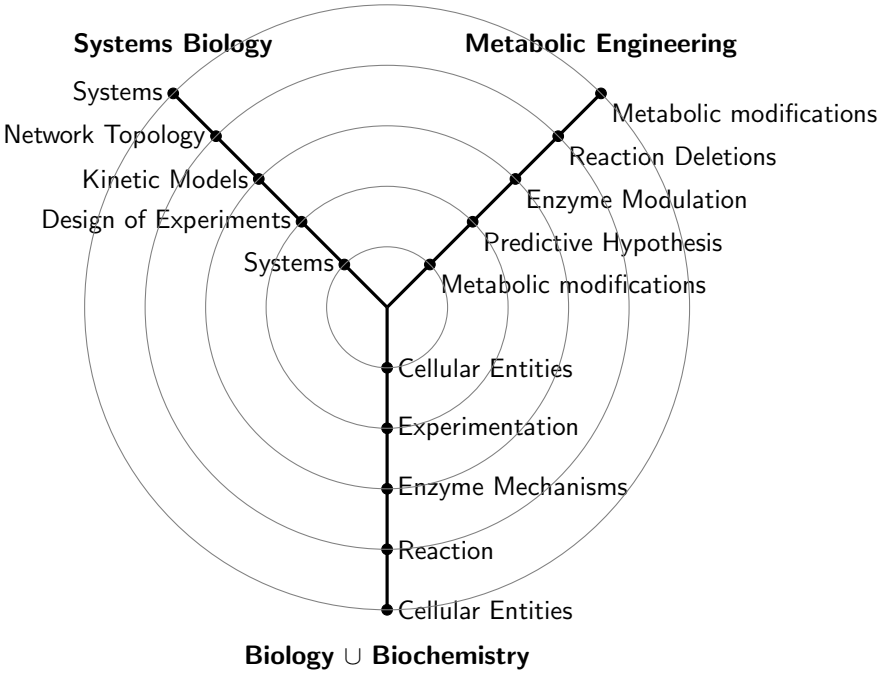


Figure 1.1: Y-Chart of the main domains addressed in this work.

The main domains that will be addressed in this work are represented on the radial axes of the y-chart in Figure 1.1. The concentric rings illustrate the distinct degrees of abstraction. The outer layer, as well as the central inner circle, represents how Systems Biology (SB), Metabolic Engineering (ME), and Biology utilize the available biological information interchangeably.

The second outer layer encompasses the usage of reaction structural information to construct reaction network models where kinetic information is absent. Under certain assumptions and conditions, it is possible to determine the network flux distribution or thermodynamically feasible concentration ranges and

to simulate the effect of gene deletions. The third outer layer refers to the usage of kinetic models containing topological and enzymatic kinetic mechanisms able to capture regulatory interactions in the cell. This information permits to devise new enzyme modulation strategies from a ME perspective.

In the following layer these models can serve as basis for the design of experiments to calibrate the model, differentiate among distinct model alternatives, or make new predictive hypothesis about changes in the system.

Current models of metabolism tend to utilize ARLs or semi-mechanistic descriptions of the central carbon metabolism of *Escherichia coli*. The main goal of this work is to construct a larger scale mechanistic model of *Escherichia coli* utilizing current available information in the literature. From an industrial point of view, this model can serve as basis for the optimization of the production of relevant compounds.

The standardization effort and unified model representation constructed in [15] by the Kronecker formulation allows to represent larger models than currently is possible, with a more generic representation scheme that utilizes rate laws with any symbolic format. This framework takes advantage of the biochemical system structure to store it in sparse matrices and allows to use specific numerical methods.

Another related goal is the development of methods that permit rational microbial strain design by modifying specific cellular entities based on guidance provided by devised quantitative dynamic models in conjunction with optimization methods and experimental validation.

1.3 Objectives

The work developed during this thesis spans across different areas of knowledge namely, SB, ME, and Computer Science, being focused on developing new methods to create improved microbial strains.

This work encompassed the following specific objectives:

- To modify the existing *Escherichia coli* kroneckerized model with more complete enzyme mechanisms and with new reactions;
- To augment the existing *Escherichia coli* kroneckerized model with new pathways;
- To engineer a software tool for the simulation and optimization of metabolic models based on an extended kronecker formalism;
- To adopt the previous tool in the *in silico* design of an overproducing serine strain of *Escherichia coli*, utilizing suitable optimization algorithms;
- To use existing stoichiometric models in conjunction with dynamic models of bioprocesses to enhance and define feeding strategies;

This work entails the extension and development of the kroneckerized mechanistic model of *Escherichia coli* developed by Joshua Apgar [15] in his PhD. work. The model comprehends the central carbon metabolism. An open source software that will enable the simulation, optimization and sensitivity analysis of these types of models and the corresponding ME tasks will be built.

1.4 Thesis Outline

The thesis is organized as follows. In the current chapter **Chapter 1** an introduction of the research problem being addressed, in conjunction with the main goals

of this work are described, along with a summary of the employed bioengineering and mathematical formalisms.

Next, in **Chapter 2** a thorough review of existing selected mathematical formalisms utilized during this work in association with the existing dynamic metabolic models of central carbon metabolism, is made. The models are contrasted and related regarding the employed kinetic descriptions as well as the metabolic engineering applications.

Chapter **Chapter 3** details the expansion of the *Escherichia coli* model devised by Joshua Apgar [15] and is broadly divided into three parts: (i) model assumptions and formalism - where the general assumptions and extensions made to kronecker formalism are explained (ii) model assembly, where the simplifying assumptions are explained within the context of the *Escherichia coli* model (iii) model calibration and identifiability analysis - where the devised and utilized methods for the calibration of reaction mechanisms are explained.

Afterwards, in **Chapter 4**, a new method for the simulation of fed-batch systems when substrate is near zero, employing a new Dynamic Flux Balance Analysis formulation is characterized.

In **Chapter 5** a new genetic algorithm representation is characterized and tested using as basis the central carbon metabolism model of Chassagnole [3] using as case study serine maximization. This case study was chosen to contrast with previous results described in the literature employing different optimization methods.

After, in **Chapter 6** the software developed during this work is presented from an user perspective. The developed file formats, as well as the integration an communication of different methods is explained.

Concluding, **Chapter 7** presents the conclusions and the future work.

REFERENCES

[1] E. Ernest Young, "Biotechnology in europe the tax, finance and regulatory framework and global policy comparison."

[2] H. Kitano, "Systems biology: a brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002.

[3] C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, and M. Reuss, "Dynamic modeling of the central carbon metabolism of escherichia coli," *Biotechnology and Bioengineering*, vol. 79, no. 1, pp. 53–73, 2002.

[4] B. Teusink, J. Passarge, C. A. Reijenga, E. Esgalhado, C. C. van der Weijden, M. Schepper, M. C. Walsh, B. M. Bakker, K. van Dam, H. V. Westerhoff, *et al.*, "Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? testing biochemistry," *European Journal of Biochemistry*, vol. 267, no. 17, pp. 5313–5329, 2000.

[5] F. Shiraishi and M. Savageau, "The tricarboxylic acid cycle in dictyostelium discoideum. i. formulation of alternative kinetic representations.," *Journal of Biological Chemistry*, vol. 267, no. 32, pp. 22912–22918, 1992.

[6] A. L. Demain, "Reviews: The business of biotechnology," *Industrial Biotechnology*, vol. 3, no. 3, pp. 269–283, 2007.

[7] M. Kawai, Y. Sekine-Hayakawa, A. Okiyama, and Y. Ninomiya, "Gustatory sensation of l- and d-amino acids in humans," *Amino Acids*, vol. 43, no. 6, pp. 2349–2358, 2012.

[8] D. T. Balu, Y. Li, M. D. Puhl, M. A. Benneyworth, A. C. Basu, S. Takagi, V. Y. Bolshakov, and J. T. Coyle, "Multiple risk pathways for schizophrenia converge in serine racemase knockout mice, a mouse model of nmda receptor hypofunction," *Proceedings of the National Academy of Sciences*, vol. 110, no. 26, pp. E2400–E2409, 2013.

[9] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp, "Ecocyc: a comprehensive

database resource for escherichia coli," *Nucleic acids research*, vol. 33, no. suppl 1, pp. D334–D337, 2005.

[10] K. E. Rudd, "Ecogene: a genome sequence database for escherichia coli k-12," *Nucleic Acids Research*, vol. 28, no. 1, pp. 60–64, 2000.

[11] S. Atsumi and J. C. Liao, "Metabolic engineering for advanced biofuels production from< i> escherichia coli</i>," *Current opinion in biotechnology*, vol. 19, no. 5, pp. 414–419, 2008.

[12] A.-P. Zeng and H. Biebl, "Bulk chemicals from biotechnology: the case of 1, 3-propanediol production and the new trends," in *Tools and Applications of Biochemical Engineering Science*, pp. 239–259, Springer, 2002.

[13] V. F. Wendisch, M. Bott, and B. J. Eikmanns, "Metabolic engineering of< i> escherichia coli</i> and< i> corynebacterium glutamicum</i> for biotechnological production of organic acids and amino acids," *Current opinion in microbiology*, vol. 9, no. 3, pp. 268–274, 2006.

[14] J. Zeikus, M. Jain, and P. Elankovan, "Biotechnology of succinic acid production and markets for derived industrial products," *Applied Microbiology and Biotechnology*, vol. 51, no. 5, pp. 545–552, 1999.

[15] J. F. Apgar, *Experiment design for systems biology*. PhD thesis, Massachusetts Institute of Technology, 2009.

CHAPTER 2

FROM METABOLIC DATA TO DYNAMIC MODELS OF METABOLISM

Escherichia coli is one the most well studied prokaroytes, being a factotum organism in the biotechnology world, utilized in the production of bulk chemicals, biofuels and drugs. This microorganism has been engineered successfully for the overproduction of biotechnological relevant compounds and can be cultured easily utilizing inexpensive media.

Systems Biology aims to characterize the functioning of cellular systems by describing the interrelated behavior of their cellular constituents. The construction of detailed whole cell metabolic models has been hampered by the lack of experimental data, detailed biological knowledge, as well as, the existence of a diverse set of methods spread trough different disciplines that are needed to assemble large scale models. The development of such large scale representations have particular interest in a Metabolic Engineering (ME) perspective. These models can serve as basis for the prediction of modifications targeting specific cellular entities to gear cell metabolism to the production of desired compounds.

In this chapter methods for the representation, calibration, sensitivity analysis, identifiability, optimization and Metabolic Engineering of such models are reviewed, together with ME applications. Existing metabolic models with special emphasis in *Escherichia coli* will also be explored.

2.1 Introduction

The emerging field of Systems Biology (SB) interconnects data from distinct omics sources, in the form of models with distinct abstraction and quantification levels [1]. Semi-quantitative approaches, despite creating more refined models, may not be able to be validated against quantitative data. On the other hand, quantitative approaches are verifiable against experimental observations but require more effort for systematizing knowledge in a formal way. These distinct model granularities represent distinct levels of knowledge that capture part of the inner workings of a cellular system [2].

Cells can be abstracted as a set of interconnected layers of a network of components, namely: (i) gene regulatory networks, (ii) metabolic networks, (iii) protein-protein interactions. Depending on the problem at hand, distinct parts of these networks may be combined and formalized as a mathematical model or a set of adjoint models. These abstractions may capture the steady-state behavior of the system [3], as well as its transient dynamics [4].

Metabolic models usually describe the interactions between metabolites and enzymes, although they may also include entities like genes or even regulatory proteins. Several formulations have been employed in the past to model metabolism, such as cellular automata [5], agent based modeling [6] and differential equations [7] [8] [9]. Ordinary Differential equations models prevail as the most used formalism due to the maturity of methods and will be the scope of the following review, given their contribution in this thesis.

Commonly, metabolic dynamics are modeled as a set of differential equations, assuming that cell contents are homogeneous and each component has a large number of units (often hundreds). If it is not possible to assure these assumptions, the system has to be modeled stochastically (due to the low number of molecules) or recurring to Partial Derivative Equations (PDEs) (if the system

is not spatially homogeneous). These models do not require a description of a biomass equation and allow to simulate non steady-state processes. The main disadvantage is the larger amount of experimental data needed to calibrate the model, due to the number of parameters. It is important to bear in mind that the necessary level of mechanistic information may also vary due to the level of detail required by the modeler and the problem being tackled.

In a metabolic model (regarding biochemical reactions), the level of detail is often related with the kinetic representations utilized. Reactions can be decomposed in atomic elementary steps containing all interactions between the enzyme and participating compounds in the reaction (often characterized by mass-action kinetics called Mass Action Rate Laws (MARLs)). Semi-mechanistic rate laws, such as Michaelis-Menten [10] [11], make simplifying assumptions to merge parts of the full elementary description of the reaction mechanism. For instance, the enzyme and the respective complexes may be deemed in equilibrium. These assumptions lead to simpler functional expressions with fewer parameters that aggregate several elementary rates. These type of reaction rates are deemed aggregated rate laws (ARLs).

In the other end of the spectrum are black box kinetics, such as lin-log [12] or neural networks [13] that may require a large amount of data to train and may not extrapolate well in unseen scenarios. It is often believed that the higher the level of the mechanistic representation of rate laws employed, the narrower the physiological range of training data needed to successfully calibrate and extrapolate (due to the fact of encapsulating physical principles of the reaction process). However, it may not be feasible to stimulate reaction participating species or inputs (compounds that affect directly the reaction rate) to identify the values of the parameters. As pointed out in [14] and [15] large mechanistic descriptions of biochemical reactions are hard to interpret in the sense of identifying which

enzyme complexes are connected to a specific system response.

After the modeler has delineated and assembled the representation of the biological system under study, he/she has to calibrate its parameters utilizing a Frequentist or Bayesian approach, that can follow a panoply of strategies as illustrated in [16] [17] [18] [19].

After assembling the model, the "simple" act of regressing the model parameters, based solely on experimental data without taking into account uncertainty and identifiability issues may produce dreadful results, namely the inability to regresss the parameter values successfully (leading to the production of meaningless results) [20].

Identifiability analysis is highly associated with Sensitivity Analysis (SA), in the sense that SA methods classify model parameters as fragile or sensitive, due to the fact of affecting the model ouputs, while robust or insensitive parameters are the ones that do not disturb the model states.

The model building process should be preceded by an analysis of the structural identifiability of the system. This scrutiny should be carried employing perfect data (noise free) to verify if it is possible to pinpoint which parameters can be identified uniquely/non uniquely and which ones do not impact the model output variability - parameters with this characteristic are often deemed non-influential.

A similar analysis should be carried out after the calibration process. Parameters classified as non-identifiable or non-influential can be fixed to their nominal value or removed from the system by altering its structure if suitable. Failure to do so may lead to instabilities in the calibration process, or in other words, produce parameter estimates with high variability [21].

Often modelers have to work with already existing experimental data and/or cannot perform more experiments. Thus, it becomes vital to know which pa-

rameters drive the calibration process and which can be distinguished from one another. Most of the times there are more parameters than data points available. Thus, several sets of parameters may fit the data equally well. The problem is generally exacerbated with higher levels of mechanistic description of the reaction rate laws (that often contain more parameters and are called over-parametrized).

Another issue the modeler should take into consideration is the numerical representation of the system. To design large scale systems with thousands of variables, special matrix representations should be employed and intertwined with symbolic representations, taking advantage of special codes for faster computation.

After calibrating and validating the model, the modeler may wish to study a set of modifications to attain a specific goal, such as maximizing the flux through a target reaction [2]. This requires specialized optimization algorithms depending on the type of modification allowed (discrete optimization in reaction knock-outs or continuous in enzyme modulation or both types of optimization when tweaking continuous and discrete variables). In [22], it was shown that the dynamic model of central carbon metabolism of *Escherichia coli* could be mapped to the same solution space as the stoichiometric counterpart by modifying (a single) parameter. Dynamic models contain a small number of achievable steady states, due to the constraints imposed by the rate laws contemplating the calibrated parameter values. In contrast, methods that ignore these restrictions, employ only mass conservation and flux boundary conditions for solving underdetermined systems coupled with a specific cellular goal, may possess an infinite number of possible solutions. This result is important from an ME point of view and dictates that a reaction flux may be tuned by tweaking the enzyme concentrations on the cell.

In this chapter, we address all of the aforementioned steps in the model construction process, as well as regarding model optimization. In the following section, an overview of the model building process is given. Next, a set of kinetic rate laws often employed in the construction of such models is described. Afterwards, the system representation regarding its numerical and symbolic portrayal are illustrated. Subsequently, sensitivity and identifiability analysis are explained giving an overview of the most important methods. After, the calibration process is also explained from the frequentist and bayesian perspectives, together with possible validation schemes. Next, an overview of the optimization methods and their application is the ME follows. To close, a discussion about existing metabolic models with special focus in *Escherichia coli* is presented. Most of these topics will not be explored in full detail due to the ample scope of this review. The most important methods for this work will be delineated more thoroughly.

2.2 Model Building Cycle

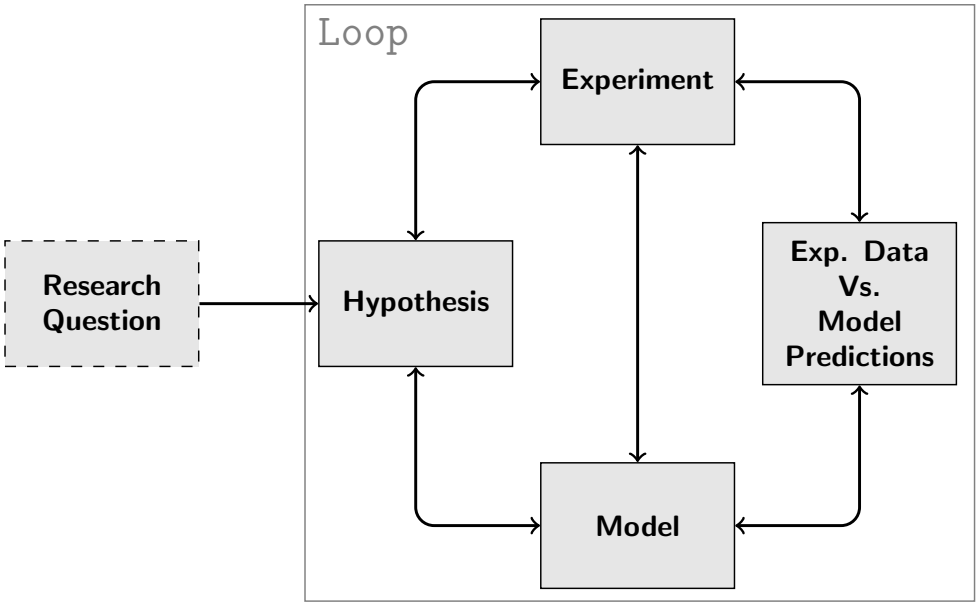


Figure 2.1: Model building cycle diagram.

Biological systems are often inherently complex due to the vast amount of interactions (often of non-linear nature) among the cellular entities. To understand the behavior of such systems and shed new insights, many times it is insufficient to perform experiments, acquire data and analyse the results. Due to the complexity of biological systems, this process has to be complemented with mathematical modeling to elucidate its behavior. In Figure 2.1, a diagram denoting possible paths for the model building cycle are shown. The first step in the process is indicated by the dashed box. This node represents the question of interest about a particular biological system that has to be translated to a particular hypothesis being investigated. This hypothesis is tested by experimentation, modeling and a comparison of model predictions against available data, thus leading to a decision regarding which step in the cycle should be taken after. This cycle may

lead to modifications in the experiments, the modeling process, redefinition of the hypotheses or the definition of new research questions. In this process, it is desirable, if possible, that the modeling stage should precede experimentation to allow the optimization of experimental data collection and the calibration of the model.

The first step in the model development process should be the unambiguous definition of the system under study. This characterization enables the definition of boundaries to confront the hypotheses being tested. The structure and mathematical formalization utilized are defined based on the model purpose. For instance, in certain cases, it may be more important to capture faithfully the biological process than to estimate accurately a specific parameter.

Next, the devised model should be calibrated against existing experimental data in conjunction, if necessary, with data from a set of designed experiments. Specific data to validate and test the hypothesis should be acquired by another set of designed experiments. Thus, these data serve as basis for comparison against model predictions. Deviations from the observed data should be carefully analysed due to the possibility of leading to new insights about the system functioning (not being modeled). It is also important to analyse model uncertainties during the distinct stages of model construction. These uncertainties can be broadly categorized as (definitions taken from [20]):

- **Random uncertainty** - This type of incertitude cannot be reduced further trough experimentation or the generation of new knowledge. However it does not possess bias and can be characterized by an adequate probabilistic framework;

- **Systemic uncertainty** - It is outlined by biased predictions caused by wrong assumptions (such as the utilization of a wrong modeling frame-

work, for a given situation, e.g. an ODE model to represent a small number of molecules), lack of specific knowledge and/or numerical errors (such as round-off errors, bugs in software, hardware failures). These errors are harder to describe by an appropriate probabilistic framework;

However, both types of incertitudes arise in several stages of the model construction cycle, namely in experimentation, simulation and model definition. For instance, errors in the model formalization are due to the physical approximation of the process or lack of knowledge concerning the system functioning.

Experimental errors ensue from equipment and human limitations concerning data acquisition and accuracy. Often, in biological systems, it is not possible to observe all the variables of interest and certain phenomena occur at very distinct time scales that may be difficult to capture in the laboratory.

Model parameters also possess uncertainties and many times can only be set by adjusting the model response to observed experimental data. When a system has a large number of parameters only a small subset will be identifiable, in the sense that only a specific parameter value will cause a determined output value. Parameters that do not correspond to this definition are called unidentifiable and can be subdivided into two groups:(i) non-influential parameters - are those that over their range produce the same model output; (ii) non-identifiable parameters - are those that possess over several alternative values that cause the same output response. These non-identifiable parameter types often pose problems to calibration algorithms leading to poor results. The identification of these parameters is often made utilizing local or global sensitivity analyses. Local sensitivity analyses are often obtained by performing infinitesimal small perturbations of a specific parameter at a given operational point (often computed recurring to derivatives or numerical approximations). Global sensitivity analysis, on the other hand, focus on the effect of varying a set of parameters on their respective

ranges. First order sensitivities are defined as [21]:

$$S_i = \frac{V_{\theta i}(E_{\theta \sim i}(Y|\theta_i))}{V(Y)} \tag{2.1}$$

where S_i stands for first order sensitivity index of parameter, $E_{\theta \sim i}(Y|\theta_i))$ represents the expected value of the output Y when θ_i is fixed in the parameter set (to a specific value), $V_{\theta i}(E_{\theta \sim i}(Y|\theta_i)$ is the first order effect of θ_i on the output Y ($V_{\theta i}$ symbolizes the variance over all values of θ_i),$V(Y)$ is the variance of the output Y. This sensitivity indexes contained in the range $[0, 1]$. Higher order sensitivities can be computed by considering:

$$S_{i,j} = \frac{V_{\theta i,j}(E_{\theta \sim i,j}(Y|\theta_{i,j}))}{V(Y)} \tag{2.2}$$

where the variables have an analogous meaning to the previously mentioned expression however, the variance is computed concerning the parameters θ_i, and θ_j. This variance corresponds to the interaction between those parameters. In a comparable form, expressions for higher order sensitivities can be computed.

The sum of all sensitivity indexes, gives origin to the total sensitivity index, or the total order effect of a parameter, given by the expression [21]:

$$S_{Ti} = 1 - \frac{E(V(Y|\theta \sim i)}{V(Y)} \tag{2.3}$$

where S_{Ti} is total sensitivity index of parameter i,$E(V(Y|\theta \sim i))$ is the expected value of the variance of output Y, when θ_i is fixed to a predefined value. This analysis permits to classify and subdivide parameter sets as identifiable or not. In [23], Saltelli identifies four scenarios for the use of global sensitivity analysis (often based on variance based methods - described in the following sections):

- Factors Prioritisation: Asks the question of which parameters should be

fixed to reduce most the variability in the output. Often this analysis is carried to identify which factors should be further researched. This is achieved by using the first order sensitivities to rank the factors. It is important to note that higher order interactions are not taken into account;

- Factors Fixing: This analysis is carried out to identify which factors have no influence in the output. The rank of the parameters is made recurring to the total sensitivity indexes;

- Variance Cutting: Determine which factors should be fixed to reduce the variability in the output to a pre-defined value;

- Factors Mapping: Identify which factors are responsible for a certain type of observed behaviour in the model output. This type of analysis is carried during the calibration process to establish which parameters are significant to drive this process.

Model calibration pertains the computation of the model parameter value or density distribution depending on the underlying framework. The frequentist approach to model calibration ascertains the existence of an unknown true fixed set of model parameter values. The purpose of this type of calibration is to estimate the set of parameter values under specified optimality criteria for $f(\theta)$. In practice, a least-squares of maximum likelihood estimators are employed (described in the following sections). In this task, the factor mapping scenario described above for global sensitivity analysis has particular importance in the identification of which parameters drive the calibration process.

It may be possible that the distribution of estimated parameter values is directly connected to the respective parameter density distribution in a probabilistic sense. If a Bayesian perspective is utilized instead (assuming that the parameters are random variables), current information regarding parameter values is

described by *a priori* probability distributions (an uniform distribution is often utilized if such information about a parameter is unavailable). These *a priori* distributions are utilized in conjunction with the observed data to update the current beliefs about the system and generate an a posteriori distribution of each model parameter that takes into account the current knowledge and the information provided by the experiment.

The variability of the parameter estimations, as well as its mean, can be computed based on the sampling distribution (the estimator is considered a random variable, thus possessing mean and covariance). The confidence intervals of this distribution can serve as a basis to quantify the precision of the calibration process.

After the calibration, the model should be validated or verified against new experimental data (not used in the calibration process) or, if such data are not available, utilizing methods such as cross validation.

2.3 Dynamic Metabolic Model Representation

Assuming the aforementioned assumptions regarding the number of molecules being model and the homogeneity in system stated in the introduction hold, dynamic models of metabolism can be generically represented by:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, \theta) \tag{2.4}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{2.5}$$

where x represents metabolite concentration vector or amounts regarding the system state, u inputs controlled by the experimenter and θ the system parame-

ters (often f is a non-linear function detailing the effects of system components in the rate of formation or consumption of x along time).

The model output is represented by:

$$y = C\mathbf{x} \tag{2.6}$$

where y is the model output, and C is a matrix (often is the identity matrix) containing the aggregation (linear combination) of the model states. In the following subsections it is assumed that C is the identity matrix, (unless stated otherwise), thus $\mathbf{x}$ will be used to refer to the outputs of the system.

In these models, the internal metabolite change rates are often represented by:

$$\frac{d\mathbf{x}}{dt} = \sum V_{creation}(\mathbf{x}, \theta) - \sum V_{consumption}(\mathbf{x}, \theta) - \mu\mathbf{x} \tag{2.7}$$

where the first term $\sum V_{creation}(\mathbf{x}, \theta)$ represents the sum of all the reaction rates that lead to the production of the compound and analogously, $\sum V_{consumption}(\mathbf{x}, \theta)$ stands for the reaction rates for the consumption of that metabolite, μ is the specific growth rate, while $\mu\mathbf{x}$ is the dilution term due to growth. The rate equations for a particular reaction may be modeled based on mechanistic physical principles, underlying the enzymatic mechanism (like for instance Michaelis-Menten), or in a black box approach relating certain species concentrations with the rate of a reaction, without taking into consideration the underlying physical processes.

Usually, a dynamic model will contain reactions modeled with different mechanistic levels, due to the limited knowledge of some enzymes (a fully complete mechanistic description of a complex enzyme can be constructed). There is a trade-off between the mechanistic level of a reaction, and the amount of data needed to calibrate the reaction rate. Often, purely mechanist reaction rates will

possess more parameters than their black-box reaction rate counterparts. On the other hand, mechanistic rate laws may be regressed on limited concentration physiological ranges maintaining extrapolation capabilities outside of these values. Purely data-driven approaches are unable to extrapolate outside of the range of values utilized in the calibration due to the lack of information concerning the enzymatic reaction physical process to help constraint and guide the prediction process [24] [25].

From an engineering and research stand point, fully mechanistic descriptions of reaction laws are appealing, due to the fact that they allow interventions and the study of enzymatic mechanisms without encasing information regarding each atomic elementary reaction step. However, the amount of experimental data, along with the signals needed to excite the system parameters, may be prohibitive.

Approximate Kinetics encapsulate partially or fully the mechanistic details of a reaction rate law, using physical assumptions (such as the rapid-equilibrium state of specific elementary reaction steps), or mathematical abstractions (to capture the Input-Output relations of a specific rate law disregarding biophysical details). These approximations serve an important role when constructing biochemical systems models, when enzyme mechanisms are not fully understood and the modeler wishes to simplify the representation of a specific reaction mechanism or to apply a specific mathematical analysis.

Models constructed with a specific approximate kinetic structure, like for instance S-Systems [26] (where rate laws are represented by power-laws) or Linlog [27], have been exploited due to the fact that certain mathematical analyses are easily carried out, sometimes providing analytical results as the computation of the steady-state. It has been shown that these descriptions may be used successfully as a first step in the construction of large scale models of metabolism,

requiring in some cases the same data employed in the construction of stoichiometric models. However, it is important to bear in mind that, despite being a very crude approximation, they may provide valuable information, such as which reaction steps are controlling certain parts of the metabolism.

Some of these representations also possess the capacity to incorporate inhibition and activation features of enzyme mechanisms in a qualitative way.

Typically, these representations possess a smaller number of parameters than fully elementary reaction descriptions, though they may also pose calibration and identifiability problems.

2.3.1 Fully Mechanistic Description of Enzymatic Reaction Rates

Reaction kinetics describe how a set of entities affects a reaction rate without considering the chemical changes taking effect. The basis for the kinetic mechanistic description of biochemical reactions is the law of mass-action, devised by Guldberg and Wage in 1867 [28].

This law states that the reaction flux will be proportional to the concentration of reactants raised to the stoichiometry (or number of molecules) of that compound taking part in the reaction. The concept of reaction order was based in this law. A zeroth order reaction occurs at a rate independent of any reaction component, while a first order reaction has a rate proportional to one reactant. A reaction is deemed of second order if its rate is proportional to the concentration of two reactants or has a single reactant with stoichiometry two.

An enzymatic mechanism may be described by an ODE system where the states contain all the species participating in a reaction, including the enzyme and all its complexes. For instance, the following reaction:

$$S \rightarrow P \tag{2.8}$$

may be represented by a Michaelis-Menten type of kinetics, and the elementary reaction system may be given by:

$$E + S \underset{k_1}{\overset{k_0}{\rightleftarrows}} ES \overset{k_2}{\rightarrow} E + P \tag{2.9}$$

The ODE system representing this reaction mechanism assumes that each elementary reaction has Mass-action kinetics and is represented by:

$$f(x, \theta, u(t)) = \begin{cases} \dot{S} = -k_0(E \times S) + k_1 ES \\ \dot{P} = k_2 ES \\ \dot{E} = -k_0(E \times S) + k_1 ES \\ \dot{ES} = -k_1 ES - k_2 ES + k_0(E \times S) \end{cases} \tag{2.10}$$

where $f(x, \theta, u(t))$ represents the derivatives of the mass action system described in 6.1, and the k_i embodies a specific elementary rate from the elementary reaction mechanism.

The ARL of the Michaelis-Menten can be deduced, considering two distinct types of assumptions that produce equivalent syntactic expressions [10] [11]. However, the parameters possess a distinct semantics. It can be assumed that all the enzyme complexes are at steady-state and the total enzyme concentration remains constant over time.

$$E_{total} = E + ES \tag{2.11}$$

Thus, the enzyme complex derivative becomes:

$$\dot{ES} = 0 \tag{2.12}$$

after the algebric manipulation of the previous expression we obtain that:

$$ES = \frac{S \times k_0 \times E_{total}}{k_1 + k_2 + S \times k_0} \tag{2.13}$$

replacing this expression (dividing the numerator and denominator by k_0) in $\dot{P}$, we attain the Michaelis-Menten expression:

$$\dot{P} = \frac{k_3 \times E_{total} \times S}{\frac{k_1+k_2}{k_0} + S} \tag{2.14}$$

Often the elementary rates and Enzyme terms are grouped as follows:

$$Vmax = k_3 \times E_{total} \tag{2.15}$$

$$Km = \frac{k_1 + k_2}{k_0} \tag{2.16}$$

where $Vmax$ depicts the maximum reaction rate value, and Km the amount of substrate needed to achieve half of the $Vmax$ value.

Nonetheless, we may arrive at a syntactic similar expression, by considering that the first step of the reaction is at rapid-equilibrium, instead of considering that enzyme complexes at steady-state. The dissociation constant of this step is given by:

$$Ks = \frac{S_{eq}E_{eq}}{ES_{eq}} \tag{2.17}$$

utilizing the expression 2.11 we obtain

$$ES = \frac{S \times k_3 \times E_{total}}{Ks + S} \tag{2.18}$$

analogously to the last derivation, we replace this expression in $\dot{P}$:

$$\dot{P} = \frac{k_3 \times E_{total} \times S}{Ks + S} \tag{2.19}$$

The $Vmax$ parameter is equal to the first derivation 2.16, while Ks corresponds to the dissociation constant of the first reaction step, instead of the half saturation constant Km.

2.3.2 From MARLs to ARLs

The King-Altman (KA) method permits to derive an ARL expression from the corresponding MARL description, by assuming that enzyme complexes are at steady-state [29]. To derive this expression, the KA method starts by constructing a graph with all the enzyme species as nodes, and the reactions and respective reactants as edges. For each of these edges, a minimum spanning tree is constructed, and serves as basis to define the patterns (as they are named in the original paper, or "the rate constants (and appropriate concentrations) associated with reaction steps which individually or in sequence lead to EX, the enzyme containing species in question"), to compute each enzyme complex fractional concentration. This fractional enzyme concentration is given by the sum of all the patterns leading to the target enzyme, divided by the sum of all the patterns corresponding to all the enzymes. The ARL is derived by replacing these enzyme fractional representations, in product rate derivative expressions with each fractional enzyme concentration multiplied by the total enzyme concentration. The complexity of the KA method grows combinatorially, as the number of cycles in the reaction graph increases.

The Cha method [30], on the other hand, simplifies the KA method (often lowering its computational complexity to manageable levels), by presuming that specific reaction steps are at rapid-equilibrium, while specific reactions are

slower than the dissociation rate constants (k_{off} - deemed as slow reaction steps).

If the slow reaction steps form cliques (containing only rapid equilibrium steps, or single enzyme species), then each of the cliques acts as a new enzyme species (that are equal to the sum of the enzyme species concentrations participating in the clique). These species outline a graph, whose edges are the slow reaction steps. This graph can serve as input for the KA method described above. The fractional enzyme concentration is given by divided by the path inside a clique.

If the graph is a single clique, then the denominator term corresponds to all the enzyme species participating in the reaction.

The derivative of the reaction product derivative is changed considering only enzyme species directly linked to slow reactions (the catalyst rate constants or k_{cat} rates). The ARL is constructed analogously to the KA method, by replacing these enzyme species by the corresponding fractional enzyme concentration expression multiplied by the participating enzyme species concentrations in the specific enzyme definition. Both of these methods are amenable to computational implementations [31] [32].

In the following subsections, the overall structure of approximate ARLs will be presented. All the rate laws will have the following high-level structure

$$\frac{d\mathbf{x}_i}{dt} = v_{foward}(\mathbf{x}, \mathbf{u}, \theta) - v_{backward}(\mathbf{x}, \mathbf{u}, \theta) = r_{\mathbf{x}_i} \tag{2.20}$$

where $\mathbf{x}_i$ is the specie i of the system, $v_{foward}(\mathbf{x}, \mathbf{u}, \theta)$ is the forward rate expression function, analogously $v_{backward}(\mathbf{x}, \mathbf{u}, \theta)$ is the backward rate function, and r_i corresponds to the time derivative of compound $\mathbf{x}_i$.

2.3.3 Power-Law and Generalized Mass Action Kinetics

A power-law rate can be described by the following expression:

$$\frac{dx_i}{dt} = \alpha \prod_{i=1}^{nSubstrates} \left(\frac{S_i}{S_{i_0}}\right)^{g_i} - \beta \prod_{j=1}^{nProducts} \left(\frac{P_j}{P_{j_0}}\right)^{h_i} \tag{2.21}$$

where α is the aggregation of the forward-rates interacting with compound S_i, analogously β is an aggregation of reverse reactions interacting with product P_i. The terms h and j represent the kinetic orders of the respective compound. These kinetic orders are the effects of a specific compound in the reaction. If they have a positive number, the compound has a positive effect, otherwise it may have no effect (if zero) or a negative impact (if less than zero). Typically, in metabolic models, reactants and products have kinetic orders corresponding to their stoichiometry, while modifier species have values denoting their effect on the reaction.

A biochemical description of a network utilizing power-law kinetics with a forward and a reverse rate is referred to as an S-System (considered as a canonical description). This formalism was conceived four decades ago by Savageau [26] with the goal of describing biological processes and providing an interpretation of systems behavior. This representation captures the non-linearities of biological processes at a local state, and allows the easy computation of system characteristics, like for instance local sensitivities, steady-state or eigenvalues. In steady-state, S-Systems are represented by a linear system. This linearity can be exploited in the optimization of the system, like it is done in [33] where is treated like a linear-programming problem, or even in the calibration process.

Generalized mass-action (GMA) rate laws are a generalization of the power-laws kinetics, where the rates are unfolded by each reaction interacting with the participating species in the reaction (equivalent to the mass-action description)

$$\frac{dx_i}{dt} = \sum_{w=1}^{nProducingReactions} \alpha_w \prod_{i=1}^{nSubstrates} \left(\frac{S_i}{S_{i_0}}\right)^{g_{w,i}} - \sum_{z=1}^{nConsumingReactions} \beta_z \prod_{j=1}^{nProducts} \left(\frac{P_i}{P_{i_0}}\right)^{h_{z,j}} \tag{2.22}$$

where w is the index of reactions that lead directly to the production of compound x_i, α_w is the corresponding reaction rate, z represents the index of the reactions consuming compound x_i, and β_z the reaction rate of the z^{th} reaction. The exponents have a similar meaning as presented for power-law rate expression, however are indexed by the corresponding reaction.

It is important to bear in mind that a GMA system can be converted to the corresponding S-System representation, notwithstanding the converse recast process cannot be done directly, due to the loss of information if the systems are not fully equivalent, or stated in another way, the GMA is already an S-system in this case. The GMA rate laws are equivalent to the elementary reaction description when utilized to describe elementary reaction steps.

2.3.4 Lin-log Kinetics

The Lin-log rate law is a black-box functional representation of an enzyme mechanism assuming that the thermodynamic driving force of a reaction is proportional to its rate. A system represented only by this type of rate laws is also in the canonical form. This canonical form allows the analytical computation of steady-states and also offers a closed form solution for the dynamics of the system [27].

The Lin-log rate law expression for flux i, in a network of M fluxes and N metabolites, is represented by:

$$r_{x_i} = J^0 \frac{e}{e^0} \left(1 + \sum_{j=1}^{N} \varepsilon_{i,j}^0 \ln \frac{M_{i,j}}{M_{i,j}^0} \right) \tag{2.23}$$

The superscripts represent the reference steady-state, where $\frac{e}{e^0}$ is the relative enzyme activity, $\varepsilon_{i,j}^0$ is the elasticity and $\frac{M}{M^0}$ represents the relative concentrations of metabolites that participate in the reaction *i*.

The elasticities possess the same interpretation as kinetic orders, the effect of a specific compound on the reaction rate (the local sensitivity).

This formalism is more appropriate for representing large concentrations of species, while power-laws are more suitable for characterizing smaller concentration ranges [34].

2.3.5 Modular Approximate Kinetics

Modular rate laws [35] encapsulate the representation of a family of semi-mechanistic rate equations in a standardized way, being described by the expression:

$$r_{x_i} = \frac{E_0 \times f_r \times T}{D + D^{reg}} \tag{2.24}$$

where r_{x_i} represents a modular rate equation, E_0 is the initial enzyme concentration, f_r represents complete or partial regulation, D is the denominator term for each specific rate law, D^{reg} the specific regulation terms and T represents an expression of stoichiometric parametrization often given by

$$r_{x_i} = k_f \prod \left(\frac{S_i}{Kms_i} \right)^{m_{s_i}} - k_r \prod \left(\frac{P_i}{Kmp_i}^{m_{p_i}} \right) \tag{2.25}$$

where k_f is the reaction forward rate, k_r is the reverse reaction rate, S_i is the

substrate i,m_{s_i} the stoichiometric value that can be multiplied by a constant h co-operative factor,P_i represents the product i, m_{p_i} the stoichiometric value that can also be multiplied by h, and the Km parameters that possess the same meaning as the MM ones (half the concentration of specific compound necessary to reach half the enzyme velocity).

2.4 Model Representation - Kronecker Formalism

Biochemical mechanistic models can be converted to a biomolecular mass action representation by decomposing the mechanistic rate law in the respective mass action reaction scheme that originated it. One problem that can arise is the fact that the original reaction scheme possesses more elementary rates than the parameters in the aggregated rate law. Thus, it is not possible to do a one-to-one mapping of elementary rates and parameters due to the fact of this system being under-determined.

This issue can be attenuated by fixing rate reactions. For instance, we can assume that the k_{on} rate reactions have a $1E6mMs^{-1}$ rate [36] due to the physical principles regarding protein diffusion. However, these values should be updated if there are experimental data available. The remaining rates can be calculated by using non-linear least squares and the KA method [29] to compute the corresponding mass action rate law for the reaction mechanism. This rate law is comparable to the mechanistic one, with the exception of carrying every rate explicitly. It is important to bear in mind that, due to the non-linear nature of some rate laws there might be distinct sets of rates that produce the same result. However, this algorithm may become impractical due to the computational cost in reaction mechanism with a large number of cycles between elementary reactions [29].

Biochemical models can be represented computationally in distinct forms. Often, a specific matrix representation takes advantage of the structural patterns of the system equations, thus allowing to design special computer codes as presented in the next section in the context of Kronecker formalism [36]. However, it is also possible to represent the model equations symbolically, for example as abstract syntax trees (ASTs), that can be directly manipulated allowing symbolic derivation and simplification of expressions.

Often in software packages, such as MATLAB, the user can explicitly define the system expressions symbolically (utilizing the adequate package) or by typing the system equations in the respective function file (implicitly, unable to do symbolic computations). In the scope of this work, two methods of depicting metabolic dynamic models take particular relevance: the Kronecker formalism [36] and symbolic model representation [37].

2.4.1 Kronecker Formalism

The interactions between different species in a biological model can be decomposed into bimolecular mass action reactions [36] [38]. These models can be represented as a system of differential equations as:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, \theta) \tag{2.26}$$

where f is a function $f|R^n \rightarrow R^m$, n is the number of species and m the number of rate equations, with $\mathbf{x}$ as concentration vector, u is a vectorial function of time and the parameter set θ as $[\mathbf{x}_0, k]^T$ (a vector initial concentrations and elementary rate parameters k). Due to the low connectivity between each species (each compound participates in average in 3.5 to 7.0 reaction as referred in [36]), and the bimolecular nature of the interactions, the function $f(\mathbf{x}, \mathbf{u}, \theta)$ can be expressed

as:

$$f(\mathbf{x}, \mathbf{u}, \theta) = A1\mathbf{x} + A2\mathbf{x} \otimes \mathbf{x} + B1\mathbf{x} + B2\mathbf{x} \otimes \mathbf{x} + k \tag{2.27}$$

where A1 ($n_X \times n_X$) is the matrix of first order interactions, A2 ($n_X \times (n_X \times n_X)$) is the matrix of second order interactions, Analogously, B1 ($n_X \times n_U$) and B2 ($n_X \times (n_U \times n_X)$) represent the first and second order interactions concerning the inputs, and k ($n_X \times 1$) describes the zeroth order reactions. Each of these matrices is stored in a sparse format, due to the low connectivity between each species. The Kronecker formalism lends its name from the Kronecker product [39] present in the expression.

The matrices A2 and B2 possess ambiguity in the representation of a biological network due to the fact of expressing explicitly all the possible interactions between two species. Thus, this idempotency permits three distinct ways of expressing second order reactions:(i) by stating that species A interacts with species B; (ii)species B interacts with A; (iii) that flux is divided in half by the two previous situations.

The matrices representing the derivative concerning the parameters of A1, A2, B1, B2 are constructed by stacking the matrix (generically called M) derivative regarding each parameter. This is done for each matrix independently. The derivative matrices possess a subscript θ. More formally, these matrices can be represented as:

$$\frac{\partial M_i}{d\theta} = \begin{pmatrix} \frac{\partial M_i}{d\theta_1} \\ \vdots \\ \frac{\partial M_i}{d\theta_n} \end{pmatrix} \tag{2.28}$$

where M_i represents one of the aforementioned matrices for specie i. To speed up the simulation of these type systems, the Kronecker products should be pre-

computed and the multiplication with A2 and B2 matrices should only occur explicitly with the non-zero terms.

Flux matrices for elementary reactions can also be decomposed in a similar fashion to the previously described mass-action representation matrices. The fluxes are a function of species concentrations, inputs, and the model parameters being described by:

$$r(\mathbf{x}, \mathbf{u}, \theta) = RA1\mathbf{x} + RA2\mathbf{x} \otimes \mathbf{x} + RB1\mathbf{x} + RB2\mathbf{x} \otimes \mathbf{x} + rk \tag{2.29}$$

where RA1 ($n_r \times n_X$) is the matrix of first order fluxes, RA2 ($n_r \times (n_X \times n_X)$) is the matrix of second order fluxes, similarly RB1 ($n_r \times n_U$) and RB2 ($n_r \times (n_U \times n_X)$) represent the first and second order fluxes respectively concerning the inputs, and rk describes the zeroth order flux reactions. Derivative matrices are also computed by stacking the matrix derivative concerning each parameter.

This format also allows to express the following partial derivatives (where the under script denotes the derivative variables) as:

$$f_{\mathbf{x}} = A_1 + A_2(I \otimes \mathbf{x} + \mathbf{x} \otimes I) + B_2(I \otimes u) \tag{2.30}$$

$$f_{\theta} = A_{1,\theta}\mathbf{x} + A_{2,\theta}(\mathbf{x} \otimes \mathbf{x}) + B_{1,\theta}\mathbf{u} + B_{2,\theta}(\mathbf{x} \otimes \mathbf{u}) + k_{\theta} \tag{2.31}$$

$$f_{\mathbf{xx}} = 2A_2 \tag{2.32}$$

$$f_{\theta\mathbf{x}} = A_{1,\theta} + A_{2,\theta}(\mathbf{x} \otimes I + I \otimes \mathbf{x}) + B_{2,\theta}(I\mathbf{x} \otimes \mathbf{u}) \tag{2.33}$$

$$f_{\theta\theta} = 0 \tag{2.34}$$

$$r_{\mathbf{x}} = RA_1 + RA_2(I \otimes \mathbf{x} + \mathbf{x} \otimes I) + RB_2(I \otimes \mathbf{u}) \tag{2.35}$$

$$r_{\theta} = RA_{1,theta}\mathbf{x} + RA_{2,\theta}(\mathbf{x} \otimes \mathbf{x}) + RB_{1,\theta}\mathbf{u} + RB_{2,\theta}(\mathbf{x} \otimes \mathbf{u}) + rk_{\theta} \tag{2.36}$$

2.4.2 Kronecker Example

To enhance the comprehension let us illustrate a small example concerning the construction of the Kronecker formalism matrices. We will utilize a Uni-Uni reversible reaction mechanism with 5 species as basis for the example:

$$E + S \underset{k_1}{\overset{k_0}{\rightleftarrows}} ES \underset{k_3}{\overset{k_2}{\rightleftarrows}} EP \underset{k_5}{\overset{k_4}{\rightleftarrows}} E + P \tag{2.37}$$

The ODE system representing this equation, assuming Mass-action kinetics- for each elementary reaction, is given by:

$$f(\mathbf{x}, \theta, \mathbf{u}(t)) = \begin{cases} \dot{S} = -k_0(E \times S) + k_1 ES \\ \dot{P} = -k_5(E \times P) + k_4 EP \\ \dot{E} = -k_0(E \times S) - k_5(E \times P) + k_1 ES + k_4 EP \\ \dot{ES} = -k_1 ES - k_2 ES + k_0(E \times S) + k_3 EP \\ \dot{EP} = -k_3 EP - k_4 EP + k_5(E \times P) + k_2 ES \end{cases} \tag{2.38}$$

It is important to notice that the order of the rows and columns can be swapped as long as it is consistent among the distinct matrices.Lets consider that there are no inputs in the system. The following matrices (zeros are not explicitly expressed) represent the species matrices:

$$A_1 = \begin{array}{c} \\ S \\ P \\ E \\ ES \\ EP \end{array} \begin{array}{c} \begin{array}{ccccc} S & P & E & ES & EP \end{array} \\ \begin{pmatrix} & & & k_1 & \\ & & & & k_4 \\ & & & k_1 & k_4 \\ & & & -k_1 & k_3 \\ & & & k_2 & -k_3 \end{pmatrix} \end{array} \tag{2.39}$$

$$A_2 = \begin{array}{c} \\ S \\ P \\ E \\ ES \\ EP \end{array} \begin{array}{c} \begin{array}{cccccccccc} S.S & S.P & S.E & S.ES & P.S & P.P & P.E & P.ES & ... & EP.ES \end{array} \\ \begin{pmatrix} & & -k_0 & & & & & & ... & \\ & & & & & & -k5 & & ... & \\ & & -k_0 & & & & -k5 & & ... & \\ & & k_0 & & & & & & ... & \\ & & & & & & k_5 & & ... & \end{pmatrix} \end{array} \tag{2.40}$$

$$
A_{1,p} = \begin{array}{c} \\ S.k_0 \\ S.k_1 \\ \vdots \\ S.k_5 \\ \vdots \\ P.k_4 \\ \vdots \\ E.k_1 \\ \vdots \\ E.k_4 \\ \vdots \\ ES.k_1 \\ \vdots \\ ES.k_3 \\ \vdots \\ EP.k_2 \\ EP.k_3 \\ \vdots \\ EP.k_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} S & P & E & ES & EP \end{array} \\
\begin{pmatrix}
 & & & & \\
 & & & -1 & \\
 & & & & \\
 & & & & \\
 & & & & \\
 & & & & 1 \\
 & & & & \\
 & & & 1 & \\
 & & & & \\
 & & & & 1 \\
 & & & & \\
 & & & -1 & \\
 & & & & \\
 & & & & 1 \\
 & & & & \\
 & & & 1 & \\
 & & & & -1 \\
 & & & & \\
 & & & &
\end{pmatrix}
\end{array}
\tag{2.41}
$$

$$A_{2,p} = \begin{array}{c} \\ S.k_0 \\ \vdots \\ P.k_5 \\ E.k_0 \\ \vdots \\ E.k_5 \\ ES.k_0 \\ \vdots \\ EP.k_5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} S.S & S.P & S.E & S.ES & P.S & P.P & P.E & P.ES & ... & EP.ES \end{array} \\ \begin{pmatrix} & & -1 & & & & & & ... & \\ & & & & & & & & ... & \\ & & & & & & -1 & & ... & \\ & & -1 & & & & & & ... & \\ & & & & & & & & ... & \\ & & & & & & -1 & & ... & \\ & & 1 & & & & & & ... & \\ & & & & & & & & ... & \\ & & & & & & 1 & & ... & \end{pmatrix} \end{array} \tag{2.42}$$

In this scenario matrices B_1, $B_{1,p}$,B_2 and $B_{2,p}$ would not possess any columns and would not be considered. If it is assumed that species S is an input then all the corresponding rows and columns of matrices A_1, $A_{1,p}$,A_2 and $A_{2,p}$ would be removed and the following set of matrices B would be created:

$$B_1 = \begin{array}{c} \\ P \\ E \\ ES \\ EP \end{array} \begin{array}{c} S \\ \begin{pmatrix} & \\ & \\ & \\ & \end{pmatrix} \end{array} \tag{2.43}$$

$$B_{1,p} = \begin{array}{c} \\ P.k_0 \\ \vdots \\ EP.k_5 \end{array} \begin{array}{c} S \\ \begin{pmatrix} & \\ & \\ & \end{pmatrix} \end{array} \tag{2.44}$$

$$B_2 = \begin{array}{c} \\ P \\ E \\ ES \\ EP \end{array} \begin{array}{c} \begin{array}{ccc} S.P & S.E & S.ES \end{array} \\ \begin{pmatrix} & & \\ & -k_0 & \\ & k_0 & \\ & & \end{pmatrix} \end{array} \tag{2.45}$$

$$B_{2,p} = \begin{array}{c} \\ P.k_0 \\ \vdots \\ P.k_5 \\ E.k_0 \\ \vdots \\ E.k_5 \\ ES.k_0 \\ \vdots \\ ES.k_5 \\ EP.k_0 \\ \vdots \\ EP.k_5 \end{array} \begin{array}{c} \begin{array}{ccc} S.P & S.E & S.ES \end{array} \\ \begin{pmatrix} & & \\ & & \\ & & \\ & -1 & \\ & & \\ & & \\ & 1 & \\ & & \\ & & \\ & & \\ & & \\ & & \end{pmatrix} \end{array} \tag{2.46}$$

If the following zeroth order reaction is added to the system

$$\xrightarrow{k_6} P \tag{2.47}$$

The previously described matrices would have to be modified accordingly and the following vectors would be created:

$$K = \begin{matrix} P \\ E \\ ES \\ EP \end{matrix} \begin{pmatrix} k_6 \\ \\ \\ \end{pmatrix} \tag{2.48}$$

$$K_p = \begin{matrix} P.k_0 \\ \vdots \\ P.k_6 \\ E.k_0 \\ \vdots \\ E.k_6 \\ ES.k_0 \\ \vdots \\ ES.k_6 \\ EP.k_0 \\ \vdots \\ EP.k_6 \end{matrix} \begin{pmatrix} \\ \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} \tag{2.49}$$

2.5 Identifiability and Sensitivity Analysis

Identifiability concerns the capability of a parameter causing a unique response in the model output being considered. If the same response is attained with distinct values of the parameter, it is classified as unidentifiable. This categorization can also be subdivided into structural identifiability, the capacity to recover the parameter's value from data free of noise and practical identifiability, the ability

to recover of the parameter's true value from data with noise [20] [21] [23].

Another important concept are non-influential parameters (a subset of non-identifiable parameters), classified in this way if the output of a model is not altered by their distinct values in a defined range.

It is important to classify the identifiability status of a parameter. The non identifiable parameters can be fixed to a baseline value reducing the overall parameter dimensionality, thus reducing the complexity of the calibration processes.

The identifiability is associated with sensitivity analysis in the sense of the analyses of the influence of the model parameters on the outputs. Sensitivity Analysis can be broadly categorized into three distinct types:

- Local Sensitivity: Based on the derivatives of outputs regarding the parameters ($\frac{\partial \mathbf{x}}{\partial \theta}$). The analysis is carried considering the system at a specified operational point and the effect of an infinitesimal change in a single parameter in the model output;

- Hybrid Global-Local Sensitivity: Utilizes the local derivatives scaled by σ_θ/σ_Y. In this scenario, global trends given by these scaling factors (the σ_θ and σ_Y - the variability over the range of the parameter) are also included in the analysis;

- Global Sensitivity Analysis (GSA): Considers the effect of the parameters on the output over the parameter ranges. These methods allow the simultaneous variation of the parameter values, and can be subdivided into:

  - Variance-based methods: decompose the variation of the output by the different model parameters. These methods allow to rank the parameters quantitatively, and identify the interactions between them;

- Screening methods: Allow to identify and rank qualitatively the parameter influence;
- Pseudo Local-global methods: utilize a sample of local derivatives over the parameter space to identify the parameters that cause the variations in the output.

Local parameter sensitivities can be computed in three distinct ways:(i) numerically, by approximating the value of $\frac{\partial \mathbf{x}_{(ti)}}{\partial \theta}$ by forward, backward or centered finite differences, or complex step differentiation. The step chosen in these methods can have a high impact in the quality of the derivative (also affected by numerical errors) leading to errors in the search directions of the optimization problem ; (ii) The model equations can be solved in conjunction with the sensitivity equations:

$$\frac{\partial \mathbf{x}_{(t)}}{\partial \theta} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \theta} + \frac{\partial f}{\partial \theta} \tag{2.50}$$

utilizing a method to integrate these equations such as the Adjoint [40] and the Green method [41]. All these methods require the computation of the Jacobian and the parametric Jacobian; (iii) The equation 2.50 can be solved *a posteriori* by approximating the system solution by a polynomial. In this case, it is not necessary to compute the system Jacobian (of equation 2.50).

In this work, the Polynomial Approximation Method (PAM) is utilized to compute the sensitivities. In the PAM, the time interval of interest concerning the sensitivity of the parameters is divided into a set of sub-intervals. In each of these sub-intervals, the participating species concentrations are interpolated by a low order polynomial.

These interpolants approximate the parametric dependence of the original system. This permits the transformation of sensitivity differential equations into

a set of algebraic ones. The computational complexity of the method scales with the number of species being independent of the number of parameters. This contrasts with the direct and adjoint methods whose complexity grows with the increasing number of parameters. The complexity of PAM is for each sub-interval $O((n_x \times l) + O((n_x \times l)^3)$, where l represents the interpolant polynomial degree.

Considering a valid time interval defined in $[c, d]$, and presuming that an Nth order Lagrange interpolation polynomial is able to describe the time behavior of i^{th} species concentration, then $\frac{\partial x(t)}{\partial \theta_i}$ can be represented as [42], [43]:

$$\frac{\partial x(t)}{\partial \theta_i} = \sum_{k=0}^{L} l_k(t) \frac{\partial x(t_k)}{\partial \theta_i}, i = 1, 2, \ldots, n_\theta \tag{2.51}$$

Replacing $\frac{\partial x(t)}{\partial \theta_i}$ in 2.51, we obtain

$$\begin{gathered} h^{-1} \sum_{k=0}^{L} l'_k(u_l) w_i(u_k) - \sum_{j=1}^{N} \frac{\partial f}{\partial x_j}(u_l) w_j(u_l) = \frac{\partial f}{\partial \theta_i}(u_l) \\ i = 1, 2, \ldots, n_x, l = 0, 1, \ldots, L \end{gathered} \tag{2.52}$$

This expression can be transformed into

$$A \frac{\partial x}{\partial \theta_i} = g \tag{2.53}$$

where A is a sparse matrix composed by $n_x \times nx$ block matrices $B_{p,q}$ with $p = 1, 2, \ldots, n_x, q = 1, 2, \ldots, n_x$. Each of these blocks is an $l \times l$ matrix represented by

$$\begin{aligned} (B_{p,p})_{(i,j)} &= h^{-1} l'_j(u_i) - \delta_{i,j} \frac{\partial f}{\partial x}(u_i) \\ (B_{p,q})_{(i,j)} &= -\delta_{i,j} \frac{\partial f}{\partial x}(u_i) \text{ with } p \neq q \end{aligned} \tag{2.54}$$

where $\delta_{i,j}$ is the kronecker delta (corresponding to the Lagrange polynomial derivative). By definition a Lagrange polynomial is described by:

$$l_k(t) = \prod_{\substack{j=0 \\ j \neq k}}^{L} \frac{t - t_j}{t_k - t_j} \tag{2.55}$$

Based on this expression, it is easily verifiable that

$$\begin{aligned} l_k(t_k) &= 1 \\ l_k(t_j) &= 0 \\ j &\neq k \end{aligned} \tag{2.56}$$

The vector g regarding parameter θ_i in equation 2.53, corresponds to

$$\begin{aligned} \Big[\frac{\partial x_1}{\partial \theta_i}(u_1), \frac{\partial x_1}{\partial \theta_i}(u_2), \ldots, \frac{\partial x_1}{\partial \theta_i}(u_l), \frac{\partial x_2}{\partial \theta_i}(u_1), \frac{\partial x_2}{\partial \theta_i}(u_2), \ldots, \frac{\partial x_2}{\partial \theta_i}(u_l), \ldots, , \frac{\partial x_{n_x}}{\partial \theta_i}(u_l) \\ -h^{-1}\Big[l_0'(u_1)\frac{\partial x_1}{\partial \theta_i}(0), l_0'(u_2)\frac{\partial x_1}{\partial \theta_i}(0), \ldots l_0'(u_l)\frac{\partial x_1}{\partial \theta_i}(0), \\ l_0'(u_1)\frac{\partial x_2}{\partial \theta_i}(0), l_0'(u_2)\frac{\partial x_2}{\partial \theta_i}(0), \ldots l_0'(u_l)\frac{\partial x_2}{\partial \theta_i}(0), \\ \ldots l_0'(u_l)\frac{\partial x_{n_x}}{\partial \theta_i}\Big]\Big] \end{aligned} \tag{2.57}$$

By solving equation 2.57, the sensitivities concerning the parameter θ_i are computed. As stated in [42] vector z is computed by a matrix-vector multiplication $A^{-1}g$, when A is inverted. It is important to note that, there is only one A^{-1} in any given time interval, independently of the number of parameters. After com-

puting first-order sensitivity coefficients is possible to compute the remaining high order sensitivity coefficients by iteratively computing matrix-vector multiplications - namely, $A^{-1}g$ for different g.

The interpolation points in each sub-interval should be chosen as the zeros of proper orthogonal polynomials, as stated and observed in [42] due to the even more spreading of the error values across the interval. For third degree polynomials, one possible choice are the zeros of the third order Legendre polynomial (the roots of the polynomial are shifted to the interval [0, 1] are $\{0.5 - (0.5)^{0.5}, 0.5, 0.5 + (0.5)^{0.5}\}$). The PAM may be utilized to calculate higher order sensitivities coefficients by recomputing vector g (for more information refer to [43]).

2.5.1 Local Sensitivities - Parameter Ranking

Both local and global SA approaches permit to rank parameters by order of importance (based on a measure of the sensitivity measure that may be quantitative or semi-quantitative in nature). In this subsection, we present several metrics to rank parameters based on derivatives (and their approximations).

- **Sensitivity Matrix**: The squared matrix $n_\theta \times n_\theta$ given by $\left(\frac{\partial y}{\partial \theta}\right)^T \left(\frac{\partial y}{\partial \theta}\right)$ or weighted by the data variance $\left(\frac{\partial y}{\partial \theta}\right)^T W^{-1} \left(\frac{\partial y}{\partial \theta}\right)$ (where W is the data variance weight matrix) serves as basis for the computation of criterion to analyse if a set of parameters is identifiable and is often called the Fisher Information Matrix (FIM). For instance, if the FIM is invertible then the parameter set is said to be structural identifiable. Often, structural identifiable parameters will be badly estimated (possessing high variance) when regressed from experimental data (thus being non-identifiable in practice). The inverse of FIM serves as basis for the unbiased lower bound estimate for the asymptotic covariance matrix of the parameter set by the Cramer

Rao theorem. $Cov \geq FIM^{-1}$. Where Cov is the system parameters covariance matrix.

In dynamical time systems the FIM is often constructed based on a sample of specified time points, and is given by:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial y_1(t)}{\partial \theta_1} & \cdots & \frac{\partial y_1(t)}{\partial \theta_n} \\ \vdots & & \vdots \\ \frac{\partial y_m(t)}{\partial \theta_1} & \cdots & \frac{\partial y_m(t)}{\partial \theta_n} \end{bmatrix} \tag{2.58}$$

where $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}}$ is the matrix of system outputs derivatives regarding the parameter set θ, it is assumed that the system has m outputs and n parameters. If this matrix has full rank, it implies that the system is structurally identifiable. Independent columns of the matrix are perpendicular, while dependent columns form a zero degree angle. All the remaining columns are partially colinear being the degree of dependence measured by the cosine angle between them [44].

- **Sum Of Relative Sensitivities**:

$$S_\theta^{\mathbf{x}} = \sum_{t=1}^{nTimePoints} \left(\frac{dln\mathbf{x}(t)}{dln\theta} \right)^2 = \sum_{t=1}^{nTimePoints} \left(\frac{d\mathbf{x}(t)}{d\theta} \times \frac{\theta}{\mathbf{x}(t)} \right)^2 \tag{2.59}$$

where t is an index starting at one, $nTimePoints$ is the number of time points considered, $S_\theta^{\mathbf{x}}$ represents a relative sensitivity, computed by summing all the relative sensitivity terms for each time point ($\frac{dln(x)}{dln(\theta)}$) of specific system output $\mathbf{x}$. Parameters with a higher mean relative sensitivity tend to affect more on average the model output [44].

- **Approximate Relative Sensitivities**:

$$S_{\theta j}^{f(\mathbf{x})} = \frac{\frac{\Delta f(\mathbf{x},\theta)}{f(\mathbf{x},\theta)}}{\frac{\Delta \theta j}{\theta j}} \tag{2.60}$$

where $f(\mathbf{x}, \theta)$ is a scalar function of the output $\mathbf{x}$ and the nominal parameter vector θ, and θ_j is a specific parameter. Contrarily to relative derivatives $\Delta\theta$ can be as large as the modeler wishes. However, it is important to bear in mind that this measure is an approximation. These sensitivities measures can be averaged out along a set of time trajectories [44].

- **Relative sensitivity matrix parameter correlation**

  One pertinent issue, concerning the parameter ranking is the identification of parameter sets, whose parameters are correlated and cannot be distinguished (their effects are compensated) and a infinite number of solutions exist, causing hurdles to optimization algorithms. These sets can be identified by computing the cosine of the angle of each pair of columns of the relative sensitivity matrix given by $\left(\frac{d\mathbf{x}(t)}{d\theta} \times \frac{\theta}{\mathbf{x}(t)}\right)$. Near parallel columns pairs are deemed unidentifiable, due to the previously stated reason [44].

2.5.2 Morrris Method - Parameter Screening

The Morris Method is a GSA that employs a one-factor-at-the-time (AOT) sampling scheme to reveal whose variables are not influential, where each parameter has a discrete set of values (called levels) and is changed a single time in a set of experiments comprehending all the parameters. This process is repeated for a predefined number of re-samples r. The method estimates the overall effect of a parameter in the output (μ) and its standard deviation that indicates the presence of higher order interactions. It is important to bear in mind that these measures are qualitative in nature.

This method has a low computational burden when compared with other GSA methods requiring $r \times (\#\theta + 1)$ simulations, where r is the number of resamples and $\#\theta$ is the number of parameters under study.

The method consists in the selection of a random initial parameter set located in a grid spaced by a predefined value δ (often a multiple of $\frac{1}{(n\#\theta-1)}$) and the subsequent realization of k experiments where in each a parameter is varied by δ . This allows to access the effect of changing a value at a time (called an elementary effect):

$$EE_i(\theta) = \frac{y(\theta_1, ..., \theta_{i-1}, \theta_i + \delta, ..., \theta_k) - y(\theta)}{\delta} \tag{2.61}$$

where $\theta = (\theta_1, \theta_2, ..., \theta_n)$ represents a parameter value in the parameter range in the space of the parameters containing also $\theta+\delta$. Elementary Effects (EEs) can be seen as coarser granularity approximation of a derivative utilizing numeric ratios. μ is computed by taking the average of this distribution for each parameter, while σ is the standard deviation of the respective distribution of elementary effects.

By random sampling the input space utilizing the OAT described scheme, is possible to construct a distribution of $EE_i(\theta)$ that serves as basis for the computation of μ and the respective standard deviation σ to rank the parameters.

Several extensions of the method have been created to deal with some of its limitations. In [45] it was assessed that mean of the absolute elementary effects distribution is a good variable in order to rank the parameters. This value is not affected by the non-monotonicity of Y, contrarily to EEs that can be cancelled out or distorted by these effects (change of signs of the function).

In [46], the EE definition was scaled by $\frac{\sigma_\theta}{\sigma_Y}$ creating a semi-hybrid local measure of sensitivity, due to the fact of contemplating a coarse approximation of a derivative with the variability of the outputs and the parameter under study).

This formulation has also the advantage of creating a dimensionless measure of sensitivity that can be compared among the different outputs in a multi-variate model.

Normally, the number of resample trajectories is chosen by trial and error. In [47] a metric was created allowing to automate the setting of the r parameter. The method consists in executing the Morris method with distinct values of r (in a ascending sequence with a fixed or variable step). Between each two experiments, the following expression (of a numerical rank giving a metric of parameter ranking between two distinct runs of the method) is computed:

$$PF_{r_i \to r_j} = \sum_{k=1}^{n\theta} \frac{|P_{k,i} - P_{k,j}|}{\frac{P_{k,i}+P_{k,j}}{2}} \tag{2.62}$$

where $P_{k,i}$ is the ranking of parameter k in resample i, similarly $P_{k,j}$ is the ranking of parameter k in resample j (with $i = j + 1$). After computing all the distinct r-experiments and computing the parameter index metrics, the r-experiment with the lowest value is chosen.

In [45] it was suggested to generate $M = 500$ to 1000 sample distributions and estimating the distance between them utilizing the expression (assuming the sample distributions are distinct, otherwise the distance is zero):

$$d_{m,l} = \sum_{i=1}^{k+1} \sum_{j=1}^{k+1} \sqrt{\sum_{z=1}^{k} [X_z^i(m) - X_z^j(l)]^2} \tag{2.63}$$

where k is the number of inputs (or parameters under study), $X_z^i(m)$ represents the point z of the m trajectory concerning input i, $X_z^j(l)$ represents the point z of m trajectory regarding input j. Afterwards, the combinations of r trajectories with the highest spread (or distance) would be selected. The drawback of this method is the combinatorial explosion of possible cases that make it infeasible even for low values of r such as 50. In [47], this problem was addressed by deriving a

method that computes only a fraction of the spreads between the trajectories. This reduces the computational time, but it may return sub-optimal solutions.

2.6 Sobol and High Dimension Model Representation - Variance Based Decomposition

High-dimensional model representation (HDMR) is a formalism that enables to capture the input-output system behaviour quantitatively shedding light on the interaction between the different variables and inputs of the system. The model output is represented by an hierarchical function expansion concerning the input variables as (often called HDMR or Sobol expansion):

$$f(\theta) = f_0 + \sum_{i=1}^{n} f_i(\theta_i) + \sum_{1\leq i\leq j\leq n} f_{ij}(\theta_i, \theta_j) + \cdots + f_{12\ldots n}(\theta_1, \theta_2, \ldots, \theta_n) \tag{2.64}$$

where f_0 represents the mean response to $f(\theta)$ and each successive order function gives the respective parameter set contribution θ to $f(\theta)$. For instance f_{ij} contains the contribution of θ_i and θ_j for the output function $f(\theta)$. It has been shown empirically that often physical phenomena can be described by an expansion up to the second order [20], given by:

$$f(\theta) \approx f_0 + \sum_{i=1}^{n} f_i(\theta_i) + \sum_{1\leq i\leq j\leq n} f_{ij}(\theta_i, \theta_j) \tag{2.65}$$

assuming that all parameters are defined in a interval $[0, 1]$, the parameter space is the n-dimensional unit hypercube (θ), where (the following explanation was borrowed from [20])

$$f_0 = \int_\theta f(\theta) d\theta \tag{2.66}$$

$$f_i(\theta_i) = \int_\theta^{\#\theta-1} f(\theta) d\theta \; i - f_0 \tag{2.67}$$

$$f_{i,j}(\theta_i, \theta j) = \int_\theta^{\#\theta-2} f(\theta) d\theta \; ijj - f_i(\theta_i) - f_j(\theta_j) - f_0 \tag{2.68}$$

The variance of the output ($Y \approx f(\theta)$) is given by

$$D = \int_\theta = f^2(\theta) d\theta - f_0^2 \tag{2.69}$$

and the partial variances

$$D_i = \int_0^1 f_i^2 d\theta_i \tag{2.70}$$

$$D_{i,j} = \int_0^1 \int_0^1 f_{ij}^2 i d\theta_i d\theta_j \tag{2.71}$$

2.6.1 RS-HDMR

The order functions (also called component functions) for the construction of the HDMR output $f(\theta)$ can be built by utilizing empirical functions fulfilling orthogonality conditions such as Legendre polynomials. This reduces the number of samples needed when contrasted with the Sobol method.

A sampling scheme (such as Latin Hyper Cube Sampling called Quasi Random Sampling HDMR, or by Random Sampling (RS) called RS-HDMR) is necessary to sample the input space (or model parameter space) and respective system response. Often, the parameters are rescaled to the interval [0, 1] being the

range of $f(\theta)$ contained in a unit hypercube. These data serve as basis for calibrating the underlying empirical functions utilized by each order function.

RS-HDMR often requires less runs than a pure Monte Carlo approach to compute each order function. Nonetheless, it is not known *a priori* which is the best polynomial degree for each component function, as well as which functions should be discarded when evaluating $f(\theta)$ (some functions may cause an increase in the output error due to the lack of information content - low Total Sobol Sensitivity).

Commonly, RS-HDMR order functions are constructed utilizing Legendre Polynomials given by:

$$\psi_1(\theta) = \sqrt{3}(2\theta - 1) \tag{2.72}$$

$$\psi_2(\theta) = 6\sqrt{5}(\theta^2 - \theta + \frac{1}{6}) \tag{2.73}$$

$$\psi_3(\theta) = 20\sqrt{7}(\theta^3 - \frac{3}{2}\theta^2 + \frac{3}{5}\theta - \frac{1}{20}) \tag{2.74}$$

Utilizing these orthonormal polynomials, $f(\theta)$ can be written as:

$$f(\theta) \approx f_0 + \sum_{i=1}^{n}\sum_{r=1}^{k}\alpha_r^i\psi(\theta_i) + \sum_{1\le i\le j\le n}\sum_{p=1}^{l}\sum_{l=1}^{w}\beta_{pq}^{ij}\psi_{pq}(\theta_i)\psi_q(\theta_j) \tag{2.75}$$

As shown in [48], α_i^r and β_{pq}^{ij} can be approximated by the expressions, respectively:

$$\alpha_r^i \approx \frac{1}{N}\sum_{s=1}^{N} f(\theta^{(s)})\psi_r(\theta_j) \tag{2.76}$$

$$\beta_{pq}^{ij} \approx \frac{1}{N} f(\theta^{(s)}) \psi_p(\theta_i^{(s)}) \psi_q(\theta_j^{(s)}) \tag{2.77}$$

In [49] and [50] these hurdles were addressed. First, each component function is built iteratively by computing a set of polynomials increasing degrees and selecting the one that produces the least squared error on the output. After, component order functions are discarded if they produce an increase in the ouput error over a certain threshold. Thus, identifying the influential parameters of the system.

It is important to bear in mind that the Sobol first and second order indices can be computed by calculating:

$$D_{\theta_i} = \frac{\alpha_i^2}{V(Y)} \tag{2.78}$$

$$D_{\theta_i \theta_j} = \frac{\beta_{pq}^2}{V(Y)} \tag{2.79}$$

where $V(Y)$ represents the total variance of model output Y.

2.6.2 Sensitivity Analysis of Time Series

In [51], a new method was developed to evaluate the global sensitivity values of time dependent series. The first step is to simulate the system utilizing the sampling scheme of the GSA being utilized. Next, functional Principal Component Analysis (fPCA) is applied to identify the set of most relevant modes (that explain the larger amount of observed variation and thus, should be selected).

Each principal component (PC) has a weight associated. This value replaces the output curve computed in the model integration. Thus, the sensitivity method estimates the sensitivity of the parameter set under study to the PC weight. Each

considered PC will have a distinct weight associated and a set of sensitivity measures (one for each parameter).

These sensitivity values (corresponding to each parameter and PC) are aggregated by summing each parameter PC score weighted by the respective PC variability value (note that this a qualitative measure). Contrary to often utilized analysis that compute the sensitivity values at distinct time points followed by their integration, this method does not weight the sensitivity of each time point equally (avoiding type I and type II errors).

2.6.3 Assessing Parameter Ranking After Screening

After ranking the parameter importance (global sensitivity indices computation or a proxy of these values), it is important to assess the correctness of the subgroup of parameters deemed as important.

The sensitivity methods do not provide a cut-off value to determine parameter importance (it may be problem dependent) and some of the methods return qualitative measures (such as the Morris method). In several works, the cut-off values are decided based on visual inspection and comparison of the measured values for the output and the respective parameters.

In [52], an elegant solution was proposed with a quantitative judgement. After ranking the parameters, the modeler would sample all parameters of the model randomly or utilizing a sampling scheme such as Latin Hyper Cube Sampling (LHCS) or Sobal random sequences [53]. Next, the modeler would define a threshold for deciding which parameters are important and non-important based on the measure returned by the sensitivity measure employed. Utilizing the previous computed samples as the basis, two new sample sets would be generated by varying randomly first the non-important parameters and then the important parameters. Subsequently, the correlation of both of the previous sets against the

first created set would be compared. This process can be iterated several times utilizing different thresholds to classify significant parameters. In the end, the threshold that produces the higer correlation with the important parameters and the lowest with non-important parameter should be used to divide and classify the parameter set. The higher correlation coefficient (assuming Pearson Correlation) should be near one, while the non-important coefficient should be near zero. If it is not the case, the analysis should be repeated with another threshold value.

2.7 Model Calibration

In the model building cycle, the problem of regressing the parameters can be often framed as a non-linear optimization problem (assuming that $x(\theta, t)$ is non-linear as in most biological systems) that contemplates a metric to evaluate the distance from the data to the devised model predictions (often utilizing Euclidean distance), assuming that the parameter values are unique. This approach is called frequentist [20]. Another approach is to consider that parameters are random variables, characterized by a probability distribution (and inherently its hyper-parameters, for instance a parameter characterized by a normal distribution will be associated with the mean and standard deviation of the distribution), and the calibration problem amounts to the computation of the posterior density function, making use of prior density functions that incorporate existing information about the parameters (such as past experiments, or information from literature). The definitions and nomenclature utilized in this section were borrowed from [20].

In the case of metabolic modeling, it is generally assumed that experimental errors (deviations between the model and measurements) are unbiased and

independent and identically distributed (iid) leading to the use of the model:

$$\psi_i = f_{x_i}(\theta) + \varepsilon_i, i = 1, \ldots, n, \tag{2.80}$$

where ψ_i is a random variable representing a measurement of the model variable i, $f_{x_i}(\theta)$ is model output for variable x_i and ε is a random variable representing measurement errors. In a frequentist approach, the set of parameters θ is unique, while in a Bayesian approach this set is characterized by an a posteriori density function.

The particular methods for the estimation of the parameters will be presented in the following sections.

2.7.1 Frequentist Parameter Estimation

If it is assumed that errors are unbiased and iid, their expected values will be zero with an unknown variance σ^2, then the following ordinary least squares estimators can be used [20]:

$$min \int_0^T (y(x(\theta, t)) - y_{data}(t))^2 dt \tag{2.81}$$

or in a discrete scenario as:

$$min \sum_{n=0}^{nTime} (y(x(\theta, t)) - y_{data}(t))^2 \tag{2.82}$$

Maximum likelihood estimators can also be employed to estimate the parameter set θ.

$$L(\theta) = L(\theta|y) \tag{2.83}$$

The likelihood $L(\theta|y)$ in this context is a function of θ and is not a probability

density function (pdf). For n iid random variables the likelihood function is given by

$$L(\theta|y) = \prod_{i=1}^{n} L(\theta|y) \tag{2.84}$$

The maximum likelihood of parameter set θ (the unique set of parameter values that maximizes the likelihood function) is calculated by maximizing the likelihood function:

$$max\ L(\theta|y) \tag{2.85}$$

In this work, the errors are assumed to be characterized by $N\ (0, \sigma^2)$. In this scenario, θ and σ are parameters being calibrated with the likelihood function:

$$L(\theta, \sigma^2|y) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(y(x(\theta,t))-y_{data}(t))^2}{2\sigma^2}} \tag{2.86}$$

Due to its monotonicity, the logarithm of likelihood function can be employed instead:

$$ln(L(\theta, \sigma^2|y)) = -\frac{n}{2}ln(2\pi) - \frac{n}{2}ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y(x(\theta, t)) - y_{data})^2 \tag{2.87}$$

The assumption of ε described by $N\ (0, \sigma^2)$ makes the solutions of maximum likelihood estimation and ordinary least-squares equal.

In the context of regressing biological models most of the variables will be constrained by bounds $lowerBound \leq x_i \leq upperBound$. Other constraints may also be imposed depending on the problem at hand. When dealing with non-linear optimization it is not possible to guarantee the global nature of the optimum.

2.7.2 Bayesian Parameter Estimation

On the other hand, if it is assumed that parameters are described by a random variable, it may be asked what set of parameters are more likely to given the observed data. This question may be performed under a Bayesian perspective, by considering [20]:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{2.88}$$

where y is the observed data, θ is the parameter set, $p(y)$ is the probability of the observed data, $p(y|\theta)$ is the maximum likelihood function of the parameter set θ and $p(\theta)$ often called the prior probability distribution encapsulates the current knowledge about the parameter set θ. The term $p(y)$ functions as a normalizing constant, and can be dropped out giving rise to the proportional relation

$$p(\theta|y) \propto p(y|\theta)p(\theta) \tag{2.89}$$

Assuming that experimental errors, and the parameter set θ are represented by normally distributed random variables, it is plausible to contemplate $p(\theta)$ as a normal distribution. In this context, this distribution is called a conjugate prior due to the fact of being of the same type as the posterior distribution. The multiplication of the terms in equation 2.89 returns a normal distribution that may be reused as a new *priori* distribution as new information becomes available. This fact allows the re-estimation of the parameter set θ, when new experimental data arises. In absence or in the presence of debatable information regarding the parameter values, the *priori* distribution should be represented by an uniform distribution (called an uninformative priori). It is important to bear in mind that, if a parameter is unidentifiable and has an uninformative prior, its calibration will not succeed.

Often, Markov Chain Monte Carlo Hastings algorithm [54] is employed to compute the parameters posterior distribution. Other algorithms, such as nested sampling [55], may also be utilized.

2.7.3 Calibration Hurdles

Over-parametrized models often pose difficulties to quantification methods due to structural and numerical identifiability issues. Numerical identifiability problems may be solved by acquiring new measurements on distinct outputs (in metabolic models these measurements may refer to metabolites or fluxes). However, a more practical alternative may be the simplification of the model by fixating the parameters whose values have no influence in the model outputs. Methods that address this issue are often based in local or global sensitivities [44] [21] [23]. In this sense, local sensitivities are referred as local in nature (in the vicinity of an operational point) and characterize an infinitesimal change (this modification may be given by a ratio of differences) in one of the parameters, while keeping the remaining variables constant. Global sensitivity methods tend to consider the parameter space range as a whole without being constrained to a specific operational point. In this scenario the simultaneous change of several parameters is accepted. The quantification of the global sensitivities allows to rank their importance for the observed dynamics of the system.

When the values of parameters are not available, assumptions have to be made to find suitable values. For instance, if the initial state of the system is in steady-state, thermodynamic information can be employed [56] [57] to devise valid metabolite bounds. In [58], the same constraints are taken in consideration with cell optimal management of enzyme levels.

Another approach to alleviate the calibration problem is to scale the rate parameters, as well as the initial concentration values, allowing to extricate the

system from the units defined by the modeler without altering the system dynamics. Often this strategy facilitates the optimization convergence [59].

2.8 Optimization

In SB, some research questions can be formulated as optimization problems from areas such as model calibration, optimal design of experiments or ME.

These approaches allow to validate, query the models and create new hypotheses. In an optimization context, these are formulated as the maximization or minimization of a cost function, possibly subject to a set of constraints (that can be linear or non-linear depending on the problem at hand). These constraints may also bound the valid range of inputs to the cost function. The methods utilized to solve these problems can be categorized broadly into the following categories: deterministic or stochastic, continuous or discrete, local or global. There are no hard boundaries and a method may belong to several of those classes.

Continuous optimization is characterized by the maximization or minimization of a cost function that accepts as input a set of real variables. In continuous optimization, local methods tend to employ gradient information to guide the search process towards an optimum. If the search space is convex, any local method that takes advantage on gradient information will suffice to find the global optimum. In non-convex search spaces, these algorithms converge to an optimum, but there are no guarantees of being the global one. Often these methods are executed in a multi-start fashion (generation of multiple initial random points for the algorithm initialization). Meta-heuristics can also be utilized and/or combined with local methods (giving origin to hybrid methods). The goal of these methods is to try to avoid getting trapped in local optima. Nevertheless, none of these methods provides any guarantee of reaching the global optima in

non-convex search spaces.

In combinatorial optimization the variables being manipulated are discrete in nature, thus not allowing the use of gradient information. If the number of variables is small enough to permit a brute force approach (testing all possible combinations of the target variables), then there is the guarantee of finding the global optima. The absence of the gradient information leads to the definition of neighbor solutions. This definition is given by the user and can have a big impact in the behavior of the optimization methods. Local combinatorial methods such as Tabu Search [60] [61], Hill Climbing or the stochastic version of these algorithms (often employed when the number of variables is large or the time to evaluate the objective function is prohibitive) tend to test all or part of the neighboring set of solutions regarding the current solution being investigated.

Global methods in combinatorial optimization are often analogous to the ones employed in continuous optimization often taking inspiration of physical or biological processes (such as biological evolution [62], the annealing of crystals [63]) to generate new solutions for the optimization problem.

In this work, continuous optimization problems regarding model calibration are solved utilizing local and global optimization methods (detailed below, namely Differential Evolution and gradient based methods such as active-set). Discrete optimization problems regarding ME strategies (reaction knockouts and enzyme level modulation) were solved utilizing meta-heuristics.

In the following subsections, a brief description of the algorithms utilized in this work is provided.

2.8.1 Gradient Descent

Gradient descent is a local method for continuous optimization problems that takes advantage the derivatives of the cost function $J(\theta)$ over parameters to reach

the optima. This method has a parameter α, often called the learning rate, that modulates the relative step taken in the direction of the optimum (as defined by the respective derivative value).

At each iteration of the algorithm, the derivative of the objective function concerning the parameters $\frac{\partial J(\theta)}{\partial \theta}$ is computed. The parameter values are updated according to (considering a minimization problem):

$$\theta_i = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \tag{2.90}$$

The algorithm stops when a user defined criterion such as solution convergence or a maximum number of iterations is reached.

2.8.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) and Genetic Algorithms (GAs) are optimization methods based on the principles that guide biological evolution, namely, selective pressure, exchange of genetic material between individuals, and genetic mutations. Computationally, these principles are emulated utilizing special function mappings (called operators) that mutate or recombine solutions. These algorithms return a solution set of evolved solutions (often the fittest individuals or the best solutions) based on a cost function supplied by the user. To apply these methods, an encoding scheme has to be defined, as well as a fitness function.

Typically, in literature, EAs contemplate a broader set of solution encoding schemes, while GAs often refer to binary representations. In this type of algorithms, solutions are often called individuals or genomes. The term gene is utilized to specify a particular position of a vector encoding a particular solution or individual or a building block of specific abstraction representing part of

a solution. In this work, these terms are utilized interchangeably.Evolutionary algorithms are composed by the following stages (Figure 2.2):

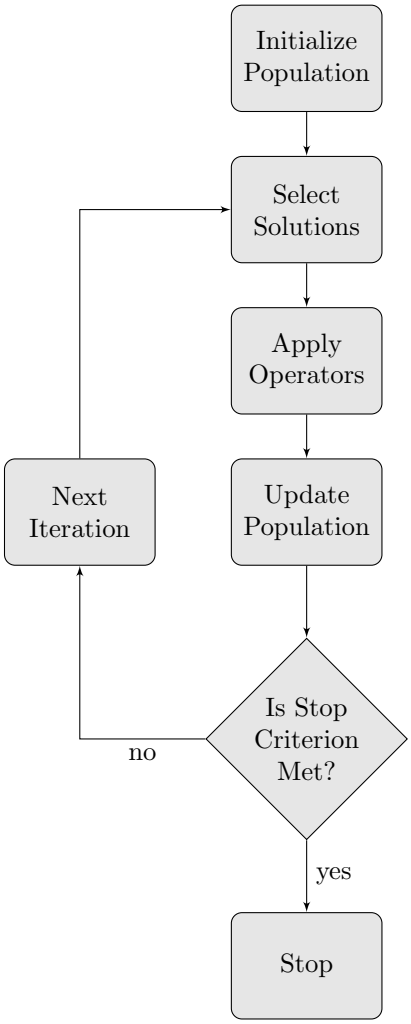


Figure 2.2: Evolutionary Algorithm schematic depiction.

1. Initialization: A set of solutions is generated representing the first generation of individuals of the algorithm. Often, these are generated by an uniform random sampling of search space variables. If specific domain knowledge is available, this process can take advantage of this information to generate the individuals;

2. Evaluation: The fitness of each solution is computed through an user defined objective function;

3. Selection: A method to select a set of solutions for the following steps of the algorithm is applied. The main goal of this procedure is to select the best solutions in detriment of the worst ones, although the process is stochastic;

4. Crossover: A set of selected individuals (solutions) are recombined, generating new individuals labelled as offspring. It is expected that the newly created solutions retain some conserved building blocks from the parents;

5. Mutation: Solutions/individuals are modified following a predefined procedure to generate new solutions;

6. The progeny created by selection, crossover and mutation replaces the ancestors in the population according to certain criteria.

The steps from 2 to 6 are repeated until a user defined stop criterion is met.

The update population node in the Figure contemplates the evaluation step as well as the replacement of the respective individuals in the population. Often, this type of algorithm is applied to problems whose optimal solutions cannot be found computationally in tractable time.

2.8.3 Differential Evolution

Differential Evolution (DE) is a population based derivative free method for continuous optimization problems with linear and non-linear constraints [64]. The population is composed by a set of real valued vectors. Each of these vectors is mapped to the set of parameters of the optimization problem being tackled.

The algorithm evolves a population of solutions to the optimization problem by perturbing each solution with the weighted difference of n distinct solutions (often $n = 2$) at each iteration.

The following is an outline of the algorithm called DE/rand/1 (utilized in this work depicted in Figure 2.3):

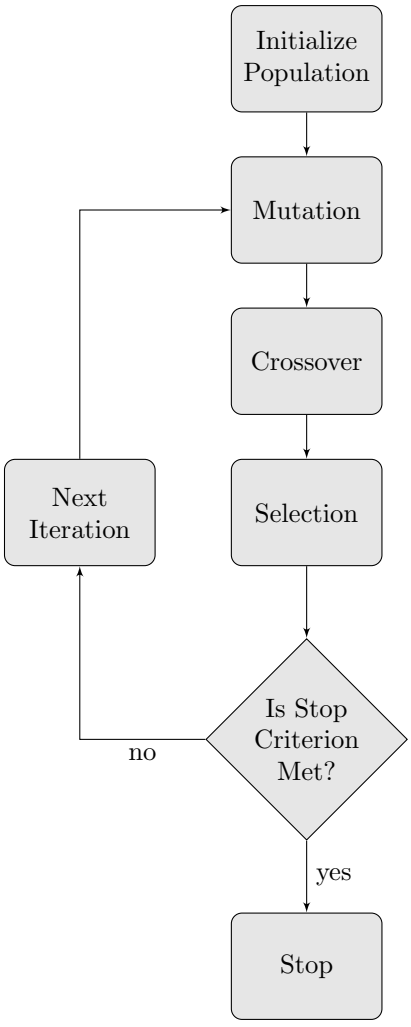


Figure 2.3: Differential Evolution Algorithm schematic depiction.

):

1. Initialization: The first step of the algorithm corresponds to the initial-

ization step (analogously to the EA). If domain specific knowledge is not available, solutions should be uniformly initialized throughout the search space.

2. Produce a new population by perturbing each solution utilizing the following steps:

   a) Select a random individual from the population; (x_{random})

   b) Two distinct solutions (x_1, x_2) are randomly selected to produce a trial solution x_{trial};

   c) Generate the trial solution x_{trial} by summing the weighted difference of two of the previously selected solutions, (The difference is multiplied by a scale factor F):

$$x_{trial} = x_{random} + F(x_1 - x_2) \tag{2.91}$$

   d) Recombine the trial vector and the respective solution in the population utilizing uniform crossover with a predefined probability CR, giving rise to a candidate solution ($x_{candidate}$). In a uniform crossover, two individuals serve as basis (often called parents) for the creation a new offspring individual. The parent vectors are compared position wise and for each position a value to pass the new offspring vector is selected based on CR value;

   e) If the newly generated solution possesses invalid positions, their values should be reset to the closest bounds;

   f) If the candidate solution ($x_{candidate}$) has a worse fitness value than the corresponding solution that originated it, the candidate solution is

discarded; otherwise, it replaces the original solution in the population;

3. Repeat this process starting from step 2, until a user defined termination criteria is met

DE algorithms can be utilized with different schemes [64], varying the number of individuals utilized in the construction of the trial vector, as well as the selection procedure of the individual to be the basis of the perturbation. Often the nomenclature of the DE scheme is given by DE/selection Procedure/number of individuals as basis of the perturbation.

2.8.4 Hybrid Differential Evolution

Adaptations have been made to the previously mentioned method, to reduce the population size, as well as to overcome the loss of genetic diversity in the individuals as the algorithm starts to converge. Of particular interest, in the scope of this work, is the adaptation made in [65] called Hybrid Differential Evolution (HDE).

The original algorithm is enhanced by the addition of two new stages after the selection procedure:

- Acceleration, in which a new solution is generated, if the previous stages (mutation, crossover and selection) did not give rise to a better solution than the previous iteration. A new solution is generated by executing a gradient descent algorithm, having as basis the best solution in the population. If the new solution has better fitness than the best solution, than it replaces it;

- Migration: To combat the loss of diversity in the population as the algorithm progresses (due to the convergence of individuals around a similar area of the search space), this stage was introduced to disperse part of the population through the search space, using as template the best individual present in the population. This stage is only activated if a metric of population diversity does not match a predefined threshold. In the description of HDE, this metric is defined in two parts. First, for each gene in the population, a genetic diversity index is computed by the expression:

$$\eta_{i,j} = \begin{cases} 1, & \text{if } \left|\frac{z_{i,j} - z_{best,j}}{z_{best,j}}\right| < \varepsilon_{genes} \\ 0, & \text{otherwise} \end{cases} \tag{2.92}$$

where $\eta_{i,j}$ is the gene diversity metric of gene j from individual i, $z_{i,j}$ corresponds to the gene j of individual i, and analogously $z_{best,j}$ is the gene j of the best individual in the population, ε_{genes} is the threshold tolerance (if the diversity $\eta_{i,j}$ is lower than this value, than $\eta_{i,j}$ equals 1 ,otherwise 0). $\eta_{i,j}$ is computed for all individuals in the population, except the best one. Afterwards, the population diversity is computed utilizing the previous calculated indexes:

$$\rho = \frac{\sum_{i=1}^{Nindividuals-1} \sum_{j=1}^{nGenes} \eta_{i,j}}{n \times (Nindividuals - 1)} \tag{2.93}$$

where ρ is the metric of population diversity, $Nindividuals$ is the number of individuals in the population and $nGenes$ is the individual size. If this metric is over a predefined threshold, the population is regenerated employing a strategy that has as basis the best individual in the population. In [65] the following schemes are utilized as examples:

$$Z_i^{putative} = Z_{best} + \mathcal{N}(0, \sigma),\ \mathrm{i} = 1\ , \ldots , Nindividuals, i \neq best \qquad (2.94)$$

where $Z_i^{putative}$ is a new putative solution to replace individual i in the population, $\mathcal{N}(0, \sigma)$ represents a vector of random generated normal distributed values with mean 0 and σ standard deviation. Another scheme is exemplified as:

$$Z_{i,j}^{putative} = \begin{cases} z_{best,j} + \delta(z_{i,min} - z_{best,j}), & \text{if } \tilde{\delta} < \frac{z_{best,j} - z_{i,min}}{z_{i,max} - z_{best,j}} \\ z_{best,j} + \delta(z_{max,j} - z_{best,j}), & \text{otherwise} \end{cases} \qquad (2.95)$$

where $Z_{i,j}^{putative}$ is a the value of gene j of new putative individual i, $z_{i,min}$ is the lower bound of gene at position i, likewise $z_{i,max}$ is the upper bound of gene at position i, $\tilde{\delta}$ and δ are random numbers.

The HDE method possesses at least two hyper-parameters more than the DE algorithm, namely, the genetic diversity threshold and the population genetic diversity threshold (without taking into consideration the gradient descent parameters).

2.8.5 Grammatical Evolution

Grammatical Evolution [66] is an EA possessing the capacity to evolve codes in a defined language, using a set of rules defined formally in a grammar. The grammar allows to express the structure of complex abstractions, such as neural networks, symbolic expressions or atomic configurations. From an algorithmic point of view, a grammar constrains what may be created, while the search algorithm explores explicit solutions to the optimization problem.

In this sense, a grammar is a set of rules describing how a complex structure is built up - each rule functions as a small building block. A valid derivation is created by chaining together a set of rules that reference each other in a particular order and give rise to a valid sentence in the target language (describing a particular abstraction).

The GE evolves solutions represented by a vector of numbers (often integers), which are utilized to select a particular rule in the derivation process. The solutions generated during this process are then translated/compiled or interpreted to a problem specific representation, and evaluated by a suitable objective function. The process may be divided into two stages: Firstly, the evolution and generation of solutions (utilizing the same operators and methods as an EA); secondly, the translation and evaluation of the abstractions generated by following the grammar rules (assuming the grammar and the translation are well defined).

Often, grammar rules are represented using the Backus Naur form (BNF) notation [67]. It is assumed that, the grammar possesses a terminal symbol set (T), contemplating all the elements that can appear in the language, and a non-terminal symbol set (NT) that contains all the rules in the grammar. Each rule is composed by head or identification string and a body with possible alternative derivations. Each derivation describes how the terminal and non-terminal symbol sets are conjugated to generate a building block of the abstraction. One of the rules is called the start rule, and is the one where the derivation process begins.

For instance, assuming we have a grammar representing simple arithmetic expressions with the following terminal and non-terminal sets:

$$T = \{+, -, /, *, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$$

$$NT = \{S, Op, BinOp, Value\}$$

and the following grammar rules:

$$\begin{aligned} S &::= < Op > \\ Op &::= < Value >< BinOp >< Op > | < Value > \\ BinOp &::= +|-|/|* \\ Value &::= 1|2|3|4|5|6|7|8|9|0 \end{aligned}$$

where S is the starting rule containing one derivation with the non-terminal symbol <Op>. The rule Op contains two possible derivations, one composed by the non-terminal symbols <Value> <BinOp> <Op> and another one with the non-terminal symbol <Value>. For example, the expression '1 + 2' can be generated by chaining the following rules:

1. S -> <Op>
2. Op -> <Value> <BinOp> <Op>
3. Value -> 1
4. BinOp -> +
5. Op -> <Value>
6. Value -> 2

The GE algorithm follows the same pattern as described for EAs in the previous section. However, the GE possesses a pre-defined decoding scheme that takes advantage of the grammar structure (defined by the user) to produce a valid mapping from the grammar rules and produce a valid textual code that will be translated to a problem specific abstraction. For example, assuming that an individual is represented by the following integer valued vector:

1	2	4	0	1	8

The method starts by the grammar start rule and chooses the current production by computing the modulus of the number in the first position of the vector by the number of productions in the rule. For each non-terminal symbol the respective rule is unfolded and the following position of the vector is utilized like in the start rule. The method is repeated until there are no more non-terminals to visit. Thus, a valid code or textual representation is generated (assuming the grammar is well defined).

In this example, the derivation process would be described by the following steps:

1. rule: S
   number Of Derivations: 1
   individual vector value:1
   chosen production index: 1 Mod 1 = 0
   chosen production: <Op>

2. rule: Op
   number Of Derivations: 2
   individual vector value:2
   chosen production index: 2 Mod 2 = 0
   chosen production: <Value><BinOp><Op>

3. rule: Value
   number Of Derivations: 10
   individual vector value:4
   chosen production index: 4 Mod 10 = 4

chosen production: 5

4. rule: BinOp
   number Of Derivations: 4
   individual vector value:0
   chosen production index: 0 Mod 4 = 0
   chosen production: +

5. rule: Op
   number Of Derivations: 2
   individual vector value:1
   chosen production index: 1 Mod 2 = 1
   chosen production: <Value>

6. rule: Value
   number Of Derivations: 10
   individual vector value:8
   chosen production index: 8 Mod 10 = 8
   chosen production: 9

It may happen that the length of the vector does not allow to form a code without non-terminal symbols. In this case, the solution may be deemed invalid or the vector may be wrapped around like in a circular vector and the process repeated until a valid code is generated or a maximum number of non-terminal symbols is visited. The depicted vector would produce the expression 5 + 9, utilizing the previously algorithm.

Variants of the canonical GE have been proposed in the literature, where the EA is replaced by other search algorithm with non-integer representations, such as a DE. The overall philosophy of the algorithm is the same as explained in this section.

GE is appealing in problems of inferring functions such as symbolic regression, due to the avoidance of explicit defining collocation points as it happens with traditional interpolation schemes. These algorithms may also capitalize on the hierarchical nature of grammars by the implicit discovery of the size and topology of a solution.

2.9 Large Scale Dynamic Models Of Metabolism

Dynamic models of metabolism present the general form of equation 2.7. Nonetheless, the discrete structure of the system (the biochemical reaction graph representation) and the symbolic representation of the rate equations are chosen based on the research question, as pointed in section 2.2. Often, non-important reactions (in the sense that they not affect the outputs being studied) may not be explicitly modelled or be represented by approximate kinetics. Notwithstanding, most of the rate laws only portray a part of the interactions with the enzyme mechanisms, mostly due to the lack of experimental data or knowledge about a particular interaction. For instance, in [68], a macroscopic model of HIV infection is presented without taking into consideration the finer-grain details of the infection (such as a mesocale representation of interaction of the viral proteins and the host T-Cell proteins).

It is important not to loose sight that a mechanistic model is a mere hypothesis to be tested, validated, modified or discredited, and whose assumptions are always open to be questioned. The risk of trying to represent too much

information about mechanistic details without evidence has to be taken into consideration, when other approximate methods may be used instead (and may be regressed equally well).

Most of the dynamic models of *Escherichia coli* tend to focus on the representation of the main pathways connected to the central carbon metabolism, due to the availability of information in the literature regarding the enzyme mechanisms and the high influence of metabolic regulation on the system behavior (genetic regulation plays a secondary role in the central carbon metabolism [69]).

The construction process of large scale dynamic models can contemplate a spectrum of experimental data, from the complete absence of experimental data regarding metabolites and fluxes measurements (the model is only based on stoichiometric information), up to the other extreme where data is collected following optimal design of experiments to attain parameter values within certain variability.

The following subsections will focus on the description and contextualization of the use of distinct rate laws for dynamic models of *Escherichia coli* portraying part of the central carbon metabolism.

In Table 2.1, a set of dynamic models regarding the *Escherichia coli* central carbon metabolism are listed. For each model, the type of data utilized during the calibration process, the identifiability methods and reaction rate detail level are catalogued.

All the listed abstractions portray part of the central carbon metabolism network. When devising a model of metabolism one of the first questions asked is what part of the biochemical network should be represented. In *Escherichia coli* most of the enzymes of the central carbon metabolism are well characterized, thus allowing to devise rate expressions to mimic their functioning. It is also important to bear in mind that the central carbon metabolism is well con-

Table 2.1: Dynamic Models of *Escherichia coli* metabolism

Model	Calibration Data	Identifiability Analysis	Rate Law Detail Level
Chassagnole [9] [70]	Concentrations	None	Semi-Mechanistic (ARL)
Peskov [71]	Concentrations	None	Semi-Mechanistic
Kronecker Chassagnole [36]	ARLs	None	Mechanistic (MARL)
Kronecker Chassagnole Extended [72]	ARLs	None	Mechanistic (MARL)
Zhao Approximate Chassagnole [73]	Concentrations and Fluxes	Coefficients of Variation	Mechanistic (ARL) and Semi-Mechanistic (MARL)
SmallBone I [74]	None	None	Approximate
SmallBone II [75]	Fluxes and Concentration	None	Approximate
Costa [76]	Fluxes and Concentration	None	Approximate
Usuda [77]	Fluxes and Concentrations	None	Approximate
Degenring [78]	Concentrations	FIM Based	Semi-Mechanistic (ARL)
Kotte [69]	Fluxes and Concentrations	None	Approximate
Standford [79]	Flux Distribuition and Parameter data	None	Semi-Mechanistic (ARL)

served across distinct microbial strains, with variations in some enzyme mechanisms [69], permitting to utilize information from different species, when necessary with explicit assumptions.

Most of the models in Table 2.1 recur to semi-mechanistic rate laws, often utilized in the same spirit as black box kinetics (like in [77]). These ARLs are employed due to uncertainty regarding the full enzyme mechanism specification.

All of these models lack any specific representation of the genetic regulation due to central carbon metabolism being regulated at enzyme level (except for [69], that contemplates the genetic regulation needed for mimicking the consumption of acetate and the switch of carbon sources).

However, most of the indexed models lack any type of identifiability analysis. Thus, some of the calibrated parameters may possess high variability, without having any biological meaning. These hurdles may arise due to structural or numerical identifiability issues, and should be taken into account when utilizing these models to answer research questions.

2.9.1 Rate Laws

One important aspect of the dynamic modeling of metabolism is how to capture the rate of change of each participating reaction in the system. Several levels of detail may be utilized under distinct assumptions, such as how enzyme complexes behave and the rates of specific elementary reactions, or certain species concentrations. These granularities should also take into account the importance of a reaction rate in the system outputs. Reactions whose effect is negligible may be modelled by simpler rate expressions (often simpler in the sense of possessing smaller number of parameters).

However, the accrue of these trade-offs between rate laws of different granularity levels should be taken into account, when studying system perturbations. Simpler rate expressions may represent well a nominal state of a system, but due to simplifications may lose extrapolation capabilities when simulating specific system perturbations due to the non-linearities, that exist in the biological system not being captured by the approximate rate law representation.

Often, the Occam razor principle is advised as the "more correct" way to model a system. However, when modelling complex biological systems it may be not the best strategy, due to the multitude of states and interactions that may affect the system impacting the final outcome of a simulation. One should strive to balance the required level of exactitude with the available experimental data and importance of the reaction on the phenomena being researched.

In most models in Table 2.1, the authors chose to utilize semi-mechanistic rate laws that mimic partially the interactions of the observed reaction enzymes

Another approach is to model the most important reactions of the system using more complex rate laws. However, assumptions such as rapid-equilibrium and steady-state regarding enzyme complexes should also be considered when selecting an appropriate rate law.

In the other extreme of the model construction is the application of approximate kinetics that utilize rate laws with a generic structure, that capture partially the behavior (often only around a nominal state) of the reaction rate of change. A particular example of such approach is [74], that constructed the network based on stoichiometric structure and parametrized the model based on stoichiomentric information without taking into account modifier species.

A full black box approach to model the rate of change of a reaction is only feasible (utilizing abstractions such as Neural Networks) if a large amount of experimental data is available. However, currently there is lack of information to model reaction kinetics under all physiological states of interest, what hampers the application of these approaches.

In 2002, Chassagnole developed (what as considered at the time) a large scale dynamic model of *Escherichia coli* central carbon metabolism [9]. The model was composed by semi-mechanistic rate equations and encompassed the phosphotransferase system, glycolysis and the pentose-phosphate pathway. When information was not available regarding certain reaction mechanisms, other more well studied mechanisms from other organisms were employed and modified instead. For instance, PFK was modeled based on the knowledge from *Saccharomyces cerevisiae* and adapted to the existing information concerning *Escherichia coli* described in literature and other available public data bases (such as Ecocyc [80]).

The reaction parameters were fitted to transient (metabolite measurements were taken at second and sub-second time scales) and to steady-state data. The *Vmax* parameters were calibrated first, having as basis the flux distribution computed utilizing a simplified stoichiometric model and the assumption that cell maximizes its growth rate.

Some of the initial concentration values had to be estimated, assuming that

specific reactions operated at near-equilibrium conditions, due to the absence of measurements concerning some metabolites. The remaining parameters, were calibrated to the transient data generated by a glucose pulse experiment, initially set to literature reported values.

In [70], authors approached the fitting of model parameters utilizing Differential Evolution [64]. The calibrated model represented similar results to the previously mentioned one. The absence of derivatives regarding the calibration is pointed out as the main advantage of the method. Deviations from experimental course experiments and the model can be explained by experimental errors and lack of representation of all the factors affecting cellular constituency.

Co-metabolites are unbalanced in this model and were modeled as time dependent functions. This approach hampers the capability of representing other transient states of the system, due to the fact of model simulation being dependent on those inputs. One way to avoid this issue, loosing the capacity to represent their evolution along time is to consider that these metabolites do not deviate from the equilibrium state. However, most of ME approaches utilizing this model follow this assumption [81] [82] [83].

In [73], the calibration of MARL was approached, by converting part of the ARL from the reactions of the Chassagnole central carbon metabolism model [9] to the equivalent MARLs. The enzyme total concentrations were determined, based on existing protein gel measurements, while their initial concentrations were determined based on KA method computations [29]. Some of the reaction mechanisms were updated, based on literature data, while others were simplified utilizing the CHA method [30] such as pyruvate kinase (PK), and phosphofructokinase (PFK) . In both cases, the tense and relaxed forms of the enzymes are deemed in rapid-equilibrium and constitute a new species, while the remaining reactions are not under this assumption.

In this work, the authors also recur to elementary reactions with non-integer orders [84], to represent enzyme conformational changes when a certain compound binds to it. This approximation violates a pure bi-molecular mass-action approach, but simplifies the mechanistic representation, avoiding the need to explicitly include all the elementary reactions to describe the behavior of the enzyme, for example to represent substrate activation. To calibrate the reactions, the authors utilize a weighted least-squares approach, to minimize the distance to the original model time series, utilizing Simulated Annealing [63] as the optimization algorithm. A calibration is discarded, if the objective function value is not below a predefined threshold.

The successful calibrations, serve as basis to compute the coefficients of variation of each elementary rate. Reactions whose rate laws are mechanistically based (and support KA method assumptions) are re-optimized, by having their less identifiable parameters constrained, by the expressions derived by the Cleland method [85].

In [36], the *Escherichia coli* model developed in [9] was extended by replacing the semi-mechanistic rate laws by the corresponding elementary reaction description MARL (represented by a set of ODEs per reaction). All the reaction mechanisms were decomposed in unitary reaction steps, with all enzyme intermediates, being the system represented by mass-action equations encoded in matrix form in the Kronecker Formalism. Co-metabolites were assumed to be at equilibrium and were not represented explicitly. This fact simplified the representation of enzymatic mechanisms where these compounds participated - (these compound were not explicitly represented, as well as the specific enzyme complexes).

The calibration of the elementary reaction rates was made by computing, for each mechanism, the steady-state rate law representation by utilizing the King-

Altman method [29]. Therefore, the calibration problem becomes to find the rate values for each reaction expression that minimize the least squares based distance to the original aggregated rate law in the participating species physiological domain. This method avoids the use of integration. The model was constructed only by fitting each reaction individually with no need of global adjustments. Murine Synthesis reaction (murSynth) in the original model [9] was changed to Michaelis-Menten kinetics without affecting the final steady-state. This reaction was originally modeled as a zeroth order reaction, thus it could lead to negative concentration when the fructose-6-phosphate was already depleted.

Another approach to modeling biochemical systems and alleviate the need for complete mechanistic information is to reduce the complexity of enzymatic workings by utilizing rate laws that approximate to a degree the true system. These approximations vary in the level of detail, and can contemplate some mechanistic aspects of the reaction or can be a full black box model that only maps the concentration values to the respective rate value without any physical connection. The main inconvenience of these rate laws is the need for a wide range of metabolite measurements along all the valid physiological states of interest, and the limited capacity of extrapolation outside this domain. This may hamper the capacity to predict certain metabolic engineering perturbations, such as reaction deletions.

In [77], a dynamic model of metabolism was created, encompassing glycolysis and TCA, with mechanistic regulation mechanisms. All the reactions were modeled as Michaelis-Menten, some with more than one binding site per compound incorporating a Hill coefficient. Enzymatic levels were also converted to reaction rates based in a linear relationship. After manual adaptations, the model was able to reproduce the *in-vivo* behavior of diauxic growth with glucose and

acetate as carbon sources. Nonetheless, there are divergences with observed external metabolites. The model was fitted using as basis 13C data and external metabolite measurements.

In [69], a simplified network (with some macro reactions, that encapsulate several non-regulated reactions), also containing glycolysis, TCA and mechanistic regulation mechanisms was created, while special attention was given to the isolated fitting of each reaction. The authors describe what they called a flux distributed sensing mechanism, where the interactions of fluxes, through specific metabolites in conjunction with transcription factors, explain the shifts in the metabolism autonomously, due to the presence of distinct carbon sources (glucose or acetate), without the presence of another higher level regulation mechanism, like signalling . As noted, stoichiometric models, with boolean regulatory networks are not able to capture this phenomena, due to their inability to incorporate enzymatic regulation, and capture the feedback loop between the distinct cellular networks (for instance, in these types of models concentrations are not modelled directly).

It was shown that the Lin-Log formalism concerning the Central Carbon metabolism of *Escherichia coli* could approach satisfactorily the concentration changes of metabolites up until 10 fold and that enzyme changes could be mapped until two fold [86]. Nonetheless, extreme values of concentrations of certain metabolites with allosteric interactions cannot be captured properly by the model. For instance, it is shown that Phosphoenolpyruvate carboxylase (PEPC) due to the fact of the system being calibrated in steady-state where the effects of the effector, are not noticeable, thus they are not reproduced by the Lin-log rate equation. However, if extra data (concerning system perturbations) were utilized in the calibration of the elasticities, maybe this effect would be better captured, by the Lin-log model.

In [76], the previously mentioned Chassagnole model [9] is extended to incorporate the TCA cycle, as well as acetate production. Several assumptions had to be made concerning the metabolite concentrations at steady-state, when experimental data was not available, such as assuming that certain reactions were at near equilibrium levels, to create a linear system of equations that allowed to solve for these missing variables. Fluxes at steady-state lacking experimental data or values were computed based in results, returned by FBA simulations with maximization of the biomass flux as objective function.

In [71], a central carbon metabolism model is presented extended with TCA utilizing aggregated rate laws encompassing four distinct levels of detail, from simple Michaelis-Menten kinetics for reactions with limited availability of information, until full mechanistic descriptions based on information in literature mostly concerning enzymes with allosteric effectors. Some of these interactions are hypothesized based on previous experimental work but even so good agreement is obtained with experimental data. This model is also used to verify the possibility of PGDH in PP pathway being regulated by PEP by constructing a similar model with this reaction modelled distinctly. This model produces a more likely agreement with observed data than the first one. This model also includes gluconeogenesis enzymes that may not be active during growth in a glucose limited media chemostat. Further experimental validation is needed to validate these hypothesis.

2.9.2 Calibration Data

There is a debate on how models should be calibrated regarding the origin of the data. It is generally agreed that *in vivo* data should be utilized to calibrate the rate laws to capture the true cellular state. *In vitro* data should only be used in last resort due to the distinct conditions of the experiment and the *in vivo* cellular

state, that may affect the behavior of the entities being measured [87].

Most of the models are calibrated utilizing data from pulse experiments, pioneered in the work of Chassagnole [9]. These experiments are characterized by a pulse of substrate when the system is at steady-state, followed by a sample of cellular states at different time points after the pulse, utilizing elaborate quenching techniques to measure metabolite concentrations. Often, there is data about the biological system at steady-state produced by distinct omics experimental techniques such as the data set produced in Keio collection [88], where steady-state data for proteomics, metabolomics and fluxomics for *Escherichia coli* at distinct dilution rates and reaction deletions are provided. Contrarily to pulse experiments, steady-state data does not provide information regarding the transient behavior of the system.

However, generally it is not possible to measure all of the system state variables, and assumptions have to be made. For instance, in Chasssagnole [9], it is assumed that for reactions in equilibrium that (near equilibrium) constant is given by (like in [89]):

$$K_j = \delta K_{eq,j} = \delta \prod_{i}^{nReactionJCompounds} x_{ss,i}^{S\,i,j} \tag{2.96}$$

$$0 \leq \delta \leq 1 \tag{2.97}$$

K_j is the near equilibrium constant for reaction j, δ is a value in the range $[0, 1]$, $K_{eq,j}$ is the near equilibrium constant for reaction j, and $x_{ss,i}^{S\,i,j}$ is the steady-state concentration of metabolite i with exponent equal to $S_{i,j}$ the stoichiometric coefficient of metabolite x_i in reaction j.

δ is set to 0.9 to simulate a deviation of 10% regarding the thermodynamic equilibrium. This constant is utilized in the computation (by the definition of

equilibrium constant) of a missing metabolite initial concentration, when a near equilibrium reaction possesses all species with a initial state defined except one. If the reaction is not at near equilibrium, this assumption does not hold and should not be used.

Another hindrance to take into consideration is the fact that the original system's steady-state flux distribution is often an accrue of existing experimental data and stoichiometric flux distribution computation methods (such as least-squares, or other optimization based approaches such as FBA depending on the system state). Generally, information about all network fluxes will be unavailable.

In [9] a reduced genome scale model is used in conjunction with rate fluxes measurements to estimate a least squares solution for the flux distribution. Based on these values, and the remaining system parameters and the $Vmax$ values are estimated by rearranging the expression:

$$V_i = Vmax_i f_i(x_{ss}, \theta) \tag{2.98}$$

$$Vmax_i = \frac{V_i}{f_i(x_{ss}, \theta)} \tag{2.99}$$

where V_i is the steady-state flux for reaction i, analogously $Vmax_i$ is the maximum attainable flux in reaction i, and $f_i(x_{ss}, \theta)$ rate function with metabolite concentrations at steady given by x_{ss} and the parameter set θ. This computation may be coupled with the calibration of parameter set θ by computing first the $Vmax$ values, followed by the calibration of the remaining parameters, and iterating this process until a termination criteria is met (such as the parameter convergence).

However, it is important to note that these $Vmax$ values in [9] may be under-

estimated, due to the fact of this particular system being at steady-state with a dilution rate of $0.1h^{-1}$. Thus, the system may lose extrapolation capabilities at different dilution rates. One possible way to overcome this hurdle would be to utilize data at different dilution rates and compute the maximum attainable *Vmax* parameter value.

When the assumptions do not hold, it may be possible to utilize more data from other authors or redefine the model structure. This fact poses another interesting question on how to integrate data from distinct experimental sources. One hypothesis is to normalize the available data around a specific state and re-fit the model utilizing a multi-objective optimization (where each distinct dataset is calibrated utilizing its own objective weighted by a specific factor) like it was done in [72] (more detailed in Section 2.9.5).

Other approaches to merge distinct data sources may comprehend one of the following strategies (such methods are not the focus of this state of the art):

- Naive pooling: where all the distinct objectives for each data source are weighted equally. It is like all the data came from the same data source and the distinct variabilities of each data source had been aggregated;

- Bayesian based methods: where previous knowledge is utilized as *a priori* distribution and updated by current experimental data given rise to *a posteriori* distribution that reflects this update in the knowledge of the system parameters (see Section 2.7.2);

- Another set of methods calibrate a model to each experimental data set computing a distinct parameter set and estimating the effect of the independent components of variance (often these methods are called two stage, due to first the individual calibration of the model for each data set, fol-

lowed by the analysis of variance individual components), while other set of approaches is only focused on the estimation of the effects;

- Data based approaches: take existing data about specific parameters such as enzyme's Kms for different organisms in distinct conditions and try to predict new values for missing parameter values and adjust existing ones with the available information.

2.9.3 Large Scale Metabolic Model Construction Automation

Another set of approaches try to automate the construction of large scale models of metabolism utilizing existing genome scale models, public available information in enzyme databases, literature and experimental data conjugated with computational methods for the construction, calibration and filling in of missing information. Most of these procedures resort to kinetic expressions with predefined structure that are applied to models at genome-scale. These approaches contemplate the possibility of constructing large scale models in the presence of only stoichiometric information or to utilize any available experimental data sources.

In [74], the authors describe a process of constructing a dynamic metabolic model employing only stoichiometric information with fluxes characterized by Lin-log kinetics and an underlying flux distribution (that can be computed by any stoichiometric based method). The elasticities were set to the negative of the stoichiometric value, as done in [90] with tendency kinetics.

Smallbone and colleagues applied this method to the construction of a model of *Saccharomyces cerevisae* glycolysis, based on the semi mechanistic model developed in [87]. The authors derived two Lin-log models: one calibrated based on the original semi-mechanistic model, and another one utilizing only the sto-

ichiometric information. Near the steady-state, both these models predicted the accurate values.

Indeed, the model constructed with only stoichiometric information presented an error of 25%, while the model calibrated to the semi-mechanistic presented an error inferior to 1% when considering all the approximations.

Contrarily to pure stoichiometric models, these dynamic models allow to verify the stability of the steady-state, as well as to identify which parameters, fluxes or metabolites exert the largest control over the observed flux distribution (in a range near the steady-state compound concentrations and flux values).

In a related work, Smallbone and co-authors [75], utilize a genome-scale stoichiometric description of the *Saccharomyces cerevisae* network, with a biomass equation from iND750 model [91] a set of reference fluxes (extracted from the literature), a set of metabolite concentrations, and elasticities to build a Lin-log genome-scale representation. The metabolite concentrations were extracted from models present in the BioModels database, by averaging these values for each distinct metabolite. Metabolite concentrations from other organism were utilized when not present in the yeast models. The remaining missing metabolites were given the median concentration value of the previously devised set. Elasticities were computed (when available rate laws were present in the models) by symbolic integration of the respective kinetic equation. Otherwise, the same strategy utilized in the previously mentioned work was followed.

The authors also applied Geometric FBA [75], that overcomes the hurdle of stoichiometric based methods that return a possible flux distribution from a set of possible hypotheses (many times this set is infinite), by returning a unique unbiased solution (no preference is given to how to distribute the fluxes, e.g. in a pathway with two branches the flux will be equally divided between the two branches) with no thermodynamically infeasible cycles. Another advantage

of the method is reproducibility and independence from the underlying linear programming solver.

After estimating these parameters, the model was utilized to compute the control coefficients of the fluxes over the glucose transport and the biomass. It was found in this model that L-asparginase may have an important role in controlling the glucose uptake. Also, it is noted in this work that reactions that have a negative impact in glucose uptake have a positive effect on the biomass flux. Without the utilization of genome scale models, with these type of methodologies, such hypothesis would hardly be generated. Notwithstanding, this model is only valid near the reference steady-state it was calibrated to. These approximative kinetics formalisms assume that cells are at specified state constrained by small variations in the environment. Thus, large perturbations to the system may not be well captured.

In [79], a similar procedure was utilized to construct genome scale metabolic model using semi-mechanistic rate laws. The computation of a basic flux distribution is similar to the previously described process in [75]. For each reaction, it was assessed which rate law would describe its behaviour based on literature and transcribed to a corresponding modular rate law [35]. The equilibrium constants for these rate equations were computed based on a method called parameter balancing [92], that contemplates thermodynamic information generating a compatible set of parameters.

The previously computed parameter values are used in the rate law constants and the value of *Vmax* parameter for each kinetic equation has to be adjusted to match the flux value computed in the Geometric FBA computation. The authors suggest utilizing Metabolic Control Analysis [93] (MCA) control coefficients to identify the most important reactions in the dynamic behavior of the network (at that particular steady-state).

All the described approaches highlight the possibility of automating the process of constructing such models. These modeling processes also highlight a trade off between accuracy and predictive capability of the models given the available information to construct them.

2.9.4 Identifiability Analysis

Identifiability analysis plays an important role in model quality assessment of existing models, while being an indispensable tool for the construction of large scale dynamic models with the currently available data. It may not possible to characterize every single state of enzyme's reactions with complex mechanisms due to the high number of possible interactions between compounds and enzyme complexes, but we may aim to construct useful detailed sub-representations that do not include all interactions, but are as close as possible to the physical process. In [78], two distinct sensitivity methods are tested to fix model parameters and reduce model complexity. A set of roughly 200 hundred models (with small differences in structure and regulatory effectors) were calibrated to experimental data, and 13 were selected, due to their goodness of fit. The models contained Michaelis Menten based ARLs with distinct regulatory hypothesis. The first, sensitivity method consisted in ranking the parameters by their relative sensitivities, regarding the model squared relative error. However, the authors note that, even parameters possessing a low sensitivity may bear a non-negligible effect on the calibration process. Thus, after fixing a parameter, its effect shall be inferred.

The second sensitivity method is the conjugation of three methods, that operate on the eigenvalues of the FIM of the system (without considering a weight matrix). After selecting the number of parameters to be reduced, the first two methods are focused in selecting the least important parameters, by using strategies to select FIM columns corresponding to low eigenvalues. On the other

hand, at the end of the procedure, the last applied method selects the most important eigenvectors and chooses the less important parameters associated with that vector, to be reduced (or fixed). Both methods tend to select a similar set of parameters to fix. However, the second approach can be utilized automatically, until a certain error threshold is passed.

The literature describes several methods for the identifiability analysis of dynamic model parameters, with distinct trade-offs concerning computational requirements and assumptions regarding the model parameters. However, most of these methods have not been applied in the construction of *Escherichia coli* models of metabolism, leading to parameter calibrations whose values are biological meaningless (due to high variability of the parameter values). In Table 2.2 several identifiability methods described in the literature are listed.

Table 2.2: Identifiability methods

Method	Type	Computational Requirements
FIM based Methods [44]	Local	Low
Sobol method [94]	Global	High
Weighted Average of Relative Sensitivities (WAR) [44]	Global	High
Multi-Parametric Sensitivity Analysis (MPSA) [44]	Global	Medium
High Dimensional Model Representation (with sampling scheme) [94] [95] [48]	Global	Medium
Morris Method [96]	Global	Low
Fourier amplitude sensitivity testing [97] [98]	Global	Low

2.9.5 Local and Global Methods in the Design Of Experiments

Model calibration may require a large set of data to fit parameters to a desired degree of confidence. It has been shown in [25] that by performing a specific set of informative experiments that try to complement each other, it is possible to calibrate a model to a desired level of accuracy. The accuracy of the experiment

design is computed by calculating a metric (such as D-Optimality) based on the FIM of the experiment.

Nonetheless, experimental constraints may hamper the ability to perform the most informative set of experiments, or even then distinct sets of parameters can describe the observed data equally well. It is important to bear in mind, that parameter sets with worse fitting may be closer to the true value of the parameters, and due to noise in the experimental setting are not portrayed as the true set.

One way to deal with this uncertainty in the parameters is to construct a model ensemble that captures the system behavior under distinct sets of parameters. The selection of these sets is dependent on a criteria set defined by the modeler. When new experimental data becomes available, it may possible to discard some sets of parameters that do not corroborate the observations. In [72], the model developed by Joshua Apgar in [36] was extended to include the EMF pathway, the enzyme isomers of glycolysis, and PP pathways, to fit a model to experimental data from different sources including the Keiko collection of steady-state knockout data concerning *Escherichia coli* and Chassagnole model ARL reaction information. The fitting was constructed as a multi-objective problem by minimizing the following objective function:

$$J(p) = w \sum_{n=1}^{nARL_{chassagnole}} \left(\frac{r_{MA,i}(x,\theta) - r_{ARL,chassagnole,i}(x)}{\sigma_{chassagnole,i}} \right)^2 + \sum_{j=1}^{nx_{ishii}} \left(\frac{y(x_i(p,t_{ss})) - x_{data,ishii,i}}{\sigma_{data,ishii}} \right)^2 \tag{2.100}$$

where w is weighting factor, $nARL_{chassagnole}$ is the number of Chassagnole model ratelaws ,$r_{MA,i}(x,\theta)$ is the MARL, $r_{ARL,chassagnole,i}(x)$ is the Chassagnole model ARL, $\sigma_{chassagnole,i}$ is the standard deviation of the Chassagnole ARL, nx_{ishii} is the number of Keiko collection metabolites measurements that are present in the

MARL (and may correspond to different system perturbations, such as knock-outs), $y(x_i(\theta, t_{ss})$ is the MARL model output (metabolite concentration at steady-state at time t_{ss}), $x_{data,ishii,i}$ is a metabolite measurement at steady-state, concerning Keiko collection data set, rescaled to MARL model steady-state.

The weight factor (w) affecting the Chassagnole fitting (the first sum in the expression) of the ARLs serves as the measure of emphasis given to fitting in detriment of the other goal. This problem was solved on a range of different predefined weight values returning in the end a Pareto front. Any solution in this set is optimal in a Pareto sense. From this set of solutions, one was chosen, and afterwards a sampling of the parameters was performed utilizing Latin Hyper Cube Sampling (LHCS). This scheme gave rise to sets of solutions and those that did not conform to a defined tolerance value for the calibration to Ishii and Chasagnole data were discarded. All parameters were presumed to be part of a multivariate Gaussian distribution with mean at each parameter best fit value and a covariance matrix computed based in the inverse of the FIM for the best fit model. These models were employed to find metabolic engineering strategies for the modulation of enzyme expression levels. All the remaining models in the Pareto solution were also included in the ensemble to reduce the bias.

2.10 Metabolic Engineering Utilizing Dynamic Models of Metabolism

The ME of dynamic models of cellular systems may be posed as an optimization problem, or a sensitivity problem (global or local), depending on the question being asked. Typically, these ME problems may be divided into two groups:

- System at steady-state: Often the goal is to maximize or minimize a cer-

tain reaction flux or metabolite concentration. For instance, at an industrial level, one pertinent question is how to maximize the target compound yield;

- System at transient state: Tries to answer questions such as what signal should some input have to increase volumetric productivity, or how to transit from one steady-state to another with the minimum cellular disruption.

The optimization of a dynamic metabolic model at steady-state, generally comprises two types of modifications: reaction deletions (discrete variables) and/or enzyme levels modulation (that can be represented by continuous or discrete variables). However, any system parameter may be modified, depending on the question being answered and the model used in the process.

On the other hand, the optimization of a transient signal is attained by regulating specific model and inputs can be formulated as a continuous (the signal or signals are given by functional representations) or a discrete problem (if the signal is represented by a set of discrete points o be interpolated).

The sensitivity analysis from a ME standpoint is concerned with the identification of the system entities who possess a larger effect on a desired target output (or a function of system outputs). Analogously, to the optimization case, SA can also be applied in steady-state or transient systems.

All those ME questions may be posed as a Multi-Objective Optimization problems, where several ME objective functions are conjugated simultaneously [99]. Nonetheless, the non-linear essence of biochemical systems may pose hurdles for these methods [100].

In Table 2.3, the main types of strategies that can be employed to attain a ME goal are listed. It is important to bear in mind that a dynamic model in this

Table 2.3: Metabolic Engineering of Dynamic Models of Metabolism - Overall Strategies

Modification	Modification Type	Applicable Methods	System State
Enzyme Modulation	Kinetic Parameters		
	• Discrete level change	OPT/SA	SS/Trans
	• Continuous level change	OPT/SA	SS/Trans
Reaction Deletion	Rate Law Present in Model		
	• Discrete level (boolean)	OPT/SA	SS/Trans

[OPT] Global or local optimization methods
[SA] Global or local Sensitivity analysis methods
[SS] Dynamic system at steady-state
[Trans] Dynamic system in a transient state

scenario may be utilized in conjunction with global or local optimization or sensitivity methods. Usually, reaction deletion ME strategies employ optimization methods, whose purpose is to find a discrete set o reaction deletions to optimize an objective function. However, a criterion may be defined to delete a reaction based on the results from SA method or even a continuous optimization method. The reaction deletion modification can be viewed as subset of the enzyme modulation problem. The reaction deletion approach can be tackled from a continuous or discrete stand-point (with proper modifications, such as considering discrete enzyme levels).

For instance, in [82], the problem of finding the best set of enzyme expression levels and reaction knockouts using a dynamic model of central carbon metabolism of *Escherichia coli* [9] was addressed. A Mixed Integer Linear Programming (MILP) formulation and a generalized linearization of the kinetic model were used to find a ME strategy. However, like in Optknock [101], the effort to solve a MILP problem increases exponentially with the size of the problem at hand. This method also assumes flux and concentration bounds around the reference state, to control the error of the linerarized model regarding the

original model.

In [81], the problem of finding the best set of enzyme expression levels using the aforementioned model was addressed. Simulated Annealing [63] was used to search the enzyme set space, while a sequential quadratic programming method estimated the respective enzyme expression levels, forcing the objective function and the constraints to be continuous in the considered ranges and of class C^2. This method assumes a value for the overall maximum allowed metabolite changes at steady-state, and also that overall system enzyme levels remain constant within a constant value proportional to the number of modifications. In both these ME problems, co-metabolites were considered to be fixed at steady-state.

Another approach that does not utilize optimization methods is, after the model calibration process, to utilize local methods, such as Metabolic Control Analysis (MCA), to gauge the systemic sensitivity of the parameter, to indicate the effectors that should be tweaked.

It is important to bear in mind that MCA [93] serves as basis for studying how an infinitesimal change in a component of biochemical system will affect the other entities. It is not obvious how to extend these results to larger perturbations without recurring to simulations of these disturbances, since they may produce a complete different set of effects. Nonetheless, this method may be used to pinpoint in a particular metabolic flux state which reactions may be the bottleneck of the system in a local sense.

In [102], Chassagnole constructed a model of threonine synthesis pathway of *Escherichia coli* utilizing an initial condition set chosen to be biologically pertinent. The ARLs were designed to be the simplest as possible but contemplating all the effector interactions for metabolites included in the model.

An altered version of this model served as basis for the study of the distribution of control fluxes of threonine by applying MCA [103] to identify the more prone reactions to adjustments. It was shown, under distinct metabolic conditions in the physiological range that the control of the enzymes remains in the first three reactions of the network. However, the aspartate concentration can affect the flux through the pathway. It is also demonstrated that feedback-inhibition mechanisms can occur on near-equilibrium steps, and it also provides two instances of enzymes that are close to equilibrium yet have finite flux control coefficients.

In [104], the MCA approach is extended by a Monte Carlo (MC) sampling of the control coefficients around the operational state of the model. This approach allows to account for the variability of the control coefficients and overcome the local nature of the original MCA method. This fact allows to pin-point which reactions exert more control on the biochemical network under uncertainty.

Another strategy to modulate enzyme expression is to utilize meta-heuristics to define the best strategy to attain a specific goal such as maximizing a production yield. In [83], the enzyme expression level was modulated utilizing an ensemble of dynamic models containing glycolysis and the TCA cyle of *Escherichia coli*. In this work, an ensemble was built by constructing a set of models that fitted equally well within a range to each experimental steady-state measurement. Enzyme expression levels were divided a in discrete interval. A Genetic Algorithm was used to systematically predict which enzyme level modifications should be carried out to maximize the production of a compound based in experimental measurements of some metabolites. The employed model consisted of the mass-action model describing the reaction mechanisms. First, an ensemble of models was calibrated to experimental data concerning steady-state metabolite measurements. Next, an optimization problem was formulated to pre-

dict which models should be removed from the ensemble for each modification based in experimental measurements. After these screening tests, a set of models is obtained with predictive capabilities and utilized in the remaining part of the algorithm.

The solutions were encoded by defining an array of integers with the double number of positions as the number of modifications. Each odd position codified the enzyme index, while the even position defined the enzyme modulation. Two objectives functions were utilized against a set of experimental steady-state measurements: (i) The maximization of variability of the target flux; (ii) The maximization of steady-state and average ensemble prediction distance. At each round of modifications, the models that were outside a predefined range were discarded. The best modifications were the ones that discarded the larger number of models. It was found, based in experimental data, that the first objective function leads to better prediction of the experimental results.

Another approach for the optimization of ME problems is to recast a model as one with a more suitable mathematical structure for the optimization process. In [100], a method where a model is first recast (if it is not already an S-system) into the corresponding S-system abstraction, followed by a linear programming optimization of the steady-state system is formulated. The result of this LP problem is then tested on the original model. If the result is satisfactory, the solution is kept, otherwise a new constraint is added to the previously LP problem and the process is repeated. This method has advantages concerning stoichiometric based due to the fact of considering metabolite and flux levels and computational has a similar cost when compared with other LP methods of genome-scale methods, utilizing only stoichiometric information.

Generally, different optimization formulations with distinct models should be tested utilizing distinct search algorithms to answer the research problem being

tackled. There is no predefined answer for each research question and several approaches should be tested. One formulation may behave well under certain circumstances and fail with a complete different question.

Ongoing critical questions for the industrial design of microbial strains are related with the extension of existing models and the integration of other cellular components that affect the regulation of metabolism and the integration of information from different omic sources in a coherent way, and how to use these abstractions to improve bio-industrial processes.

REFERENCES

[1] H. Kitano, "Systems biology: a brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002.

[2] G. Stephanopoulos, A. A. Aristidou, and J. Nielsen, *Metabolic engineering: principles and methodologies*. Academic press, 1998.

[3] B. Palsson, "Systems biology: Properties of reconstructed networks cambridge university press," *New York*, 2006.

[4] B. Ø. Palsson, *Systems biology: simulation of dynamic network states*. Cambridge University Press, 2011.

[5] D. S. Wishart, R. Yang, D. Arndt, P. Tang, and J. Cruz, "Dynamic cellular automata: an alternative approach to cellular simulation," *In silico biology*, vol. 5, no. 2, pp. 139–161, 2005.

[6] M. T. Klann, A. Lapin, and M. Reuss, "Agent-based simulation of reactions in the crowded and structured intracellular environment: Influence of mobility and location of the reactants," *BMC systems biology*, vol. 5, no. 1, p. 71, 2011.

[7] J. J. Hornberg, B. Binder, F. J. Bruggeman, B. Schoeberl, R. Heinrich, and H. V. Westerhoff, "Control of mapk signalling: from complexity to what really matters," *Oncogene*, vol. 24, no. 36, pp. 5533–5542, 2005.

[8] T. Chen, H. L. He, G. M. Church, *et al.*, "Modeling gene expression with differential equations.," in *Pacific symposium on biocomputing*, vol. 4, p. 4, 1999.

[9] C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, and M. Reuss, "Dynamic modeling of the central carbon metabolism of escherichia coli," *Biotechnology and Bioengineering*, vol. 79, no. 1, pp. 53–73, 2002.

[10] L. Michaelis and M. L. Menten, "Die kinetik der invertinwirkung," *Biochem. z*, vol. 49, no. 333-369, p. 352, 1913.

[11] G. E. Briggs and J. B. S. Haldane, "A note on the kinetics of enzyme action," *Biochemical journal*, vol. 19, no. 2, p. 338, 1925.

[12] J. J. Heijnen, "Approximative kinetic formats used in metabolic network modeling," *Biotechnology and bioengineering*, vol. 91, no. 5, pp. 534–545, 2005.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[14] A. R. Schulz, *Enzyme kinetics: from diastase to multi-enzyme systems*. Cambridge University Press, 1994.

[15] R. Heinrich and S. Schuster, *The regulation of cellular systems*. Springer Science & Business Media, 2012.

[16] M. Rodriguez-Fernandez, M. Rehberg, A. Kremling, and J. R. Banga, "Simultaneous model discrimination and parameter estimation in dynamic models of cellular systems," *BMC systems biology*, vol. 7, no. 1, p. 76, 2013.

[17] M. Rodriguez-Fernandez, P. Mendes, and J. R. Banga, "A hybrid approach for efficient and robust parameter estimation in biochemical pathways," *Biosystems*, vol. 83, no. 2, pp. 248–265, 2006.

[18] N. Pullen and R. J. Morris, "Bayesian model comparison and parameter inference in systems biology using nested sampling," *PloS one*, vol. 9, no. 2, p. e88419, 2014.

[19] J. A. Egea, E. Vazquez, J. R. Banga, and R. Martí, "Improved scatter search for the global optimization of computationally expensive dynamic models," *Journal of Global Optimization*, vol. 43, no. 2-3, pp. 175–190, 2009.

[20] R. C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*, vol. 12. SIAM, 2013.

[21] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[22] D. Machado, R. S. Costa, E. C. Ferreira, I. Rocha, and B. Tidor, "Exploring the gap between dynamic and constraint-based models of metabolism," *Metabolic engineering*, vol. 14, no. 2, pp. 112–119, 2012.

[23] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity analysis in practice: a guide to assessing scientific models*. John Wiley & Sons, 2004.

[24] A. F. Villaverde and J. R. Banga, "Reverse engineering and identification in systems biology: strategies, perspectives and challenges," *Journal of The Royal Society Interface*, vol. 11, no. 91, p. 20130505, 2014.

[25] J. F. Apgar, D. K. Witmer, F. M. White, and B. Tidor, "Sloppy models, parameter uncertainty, and the role of experimental design," *Molecular BioSystems*, vol. 6, no. 10, pp. 1890–1900, 2010.

[26] W. Knorre, "Ma savageau, biochemical systems analysis. a study of function and design in molecular biology. 396 s., 115 abb., 14 tab. reading, mass. 1976. addison-wesley pbl. co./advanced book program.£ 26, 50," *Zeitschrift für allgemeine Mikrobiologie*, vol. 19, no. 2, pp. 149–150, 1979.

[27] D. Visser and J. J. Heijnen, "Dynamic simulation and metabolic re-design of a branched pathway using linlog kinetics," *Metabolic engineering*, vol. 5, no. 3, pp. 164–176, 2003.

[28] E. Lund, "Guldberg and waage and the law of mass action," *Journal of Chemical Education*, vol. 42, no. 10, p. 548, 1965.

[29] E. L. King and C. Altman, "A schematic method of deriving the rate laws for enzyme-catalyzed reactions," *The Journal of physical chemistry*, vol. 60, no. 10, pp. 1375–1378, 1956.

[30] S. Cha, "A simple method for derivation of rate equations for enzyme-catalyzed reactions under the rapid equilibrium assumption or combined assumptions of equilibrium and steady state," *Journal of biological chemistry*, vol. 243, no. 4, pp. 820–825, 1968.

[31] A. Cornish-Bowden, "An automatic method for deriving steady-state rate equations.," *Biochem. J*, vol. 165, pp. 55–59, 1977.

[32] H. Ishikawa, T. Maeda, H. Hikita, and K. Miyatake, "The computerized derivation of rate equations for enzyme reactions on the basis of the pseudo-steady-state assumption and the rapid-equilibrium assumption.," *Biochem. J*, vol. 251, pp. 175–181, 1988.

[33] A. Marin-Sanguino and N. V. Torres, "Optimization of biochemical systems by linear programming and general mass action model representations," *Mathematical biosciences*, vol. 184, no. 2, pp. 187–200, 2003.

[34] F.-S. Wang, C.-L. Ko, and E. O. Voit, "Kinetic modeling using s-systems and lin-log approaches," *Biochemical engineering journal*, vol. 33, no. 3, pp. 238–247, 2007.

[35] W. Liebermeister, J. Uhlendorf, and E. Klipp, "Modular rate laws for enzymatic reactions: thermodynamics, elasticities and implementation," *Bioinformatics*, vol. 26, no. 12, pp. 1528–1534, 2010.

[36] J. F. Apgar, *Experiment design for systems biology*. PhD thesis, Massachusetts Institute of Technology, 2009.

[37] R. Albrecht, B. Buchberger, G. E. Collins, and R. Loos, *Computer algebra: symbolic and algebraic computation*, vol. 4. Springer Science & Business Media, 2013.

[38] N. Jamshidi and B. Ø. Palsson, "Mass action stoichiometric simulation models: incorporating kinetics and regulation into stoichiometric models," *Biophysical journal*, vol. 98, no. 2, pp. 175–185, 2010.

[39] H. Neudecker, "A note on kronecker matrix products and matrix equation systems," *SIAM Journal on Applied Mathematics*, vol. 17, no. 3, pp. 603–606, 1969.

[40] Y. Cao, S. Li, L. Petzold, and R. Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint dae system and its numerical solution," *SIAM Journal on Scientific Computing*, vol. 24, no. 3, pp. 1076–1089, 2003.

[41] M. Kramer, J. Calo, and H. Rabitz, "An improved computational method for sensitivity analysis: Green's function method with 'aim'," *Applied Mathematical Modelling*, vol. 5, no. 6, pp. 432–441, 1981.

[42] J.-T. Hwang, "Sensitivity analysis in chemical kinetics by the method of polynomial approximations," *International Journal of Chemical Kinetics*, vol. 15, no. 10, pp. 959–987, 1983.

[43] J.-T. Hwang, "A computational algorithm for the polynomial approximation method of sensitivity analysis in chemical kinetics," *Journal of the Chinese Chemical Society*, vol. 32, no. 3, pp. 253–261, 1985.

[44] J. DiStefano III, *Dynamic systems biology modeling and simulation*. Academic Press, 2015.

[45] F. Campolongo, J. Cariboni, and A. Saltelli, "An effective screening design for sensitivity analysis of large models," *Environmental modelling & software*, vol. 22, no. 10, pp. 1509–1518, 2007.

[46] G. Sin and K. V. Gernaey, "Improving the morris method for sensitivity analysis by scaling the elementary effects," *Computer Aided Chemical Engineering*, vol. 26, pp. 925–930, 2009.

[47] M. Ruano, J. Ribes, A. Seco, and J. Ferrer, "An improved sampling strategy based on trajectory design for application of the morris method to systems with many input factors," *Environmental Modelling & Software*, vol. 37, pp. 103–109, 2012.

[48] G. Li, S.-W. Wang, and H. Rabitz, "Practical approaches to construct rs-hdmr component functions," *The Journal of Physical Chemistry A*, vol. 106, no. 37, pp. 8721–8733, 2002.

[49] T. Ziehn and A. Tomlin, "Global sensitivity analysis of a 3d street canyon model—part i: The development of high dimensional model representations," *Atmospheric Environment*, vol. 42, no. 8, pp. 1857–1873, 2008.

[50] T. Ziehn and A. S. Tomlin, "A global sensitivity study of sulfur chemistry in a premixed methane flame model using hdmr," *International Journal of Chemical Kinetics*, vol. 40, no. 11, pp. 742–753, 2008.

[51] T. Sumner, E. Shephard, and I. Bogle, "A methodology for global-sensitivity analysis of time-dependent outputs in systems biology modelling," *Journal of The Royal Society Interface*, vol. 9, no. 74, pp. 2156–2166, 2012.

[52] T. Andres, "Sampling methods and sensitivity analysis for large parameter sets," *Journal of Statistical Computation and Simulation*, vol. 57, no. 1-4, pp. 77–110, 1997.

[53] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational mathematics and mathematical physics*, no. 7, pp. 86–112, 1967.

[54] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[55] J. Skilling, "Nested sampling," *Bayesian inference and maximum entropy methods in science and engineering*, vol. 735, pp. 395–405, 2004.

[56] N. Zamboni, A. Kümmel, and M. Heinemann, "annet: a tool for network-embedded thermodynamic analysis of quantitative metabolome data," *BMC bioinformatics*, vol. 9, no. 1, p. 199, 2008.

[57] A. Kümmel, S. Panke, and M. Heinemann, "Putative regulatory sites unraveled by network-embedded thermodynamic analysis of metabolome data," *Molecular systems biology*, vol. 2, no. 1, 2006.

[58] N. Tepper, E. Noor, D. Amador-Noguez, H. S. Haraldsdóttir, R. Milo, J. Rabinowitz, W. Liebermeister, and T. Shlomi, "Steady-state metabolite concentrations reflect a balance between maximizing enzyme efficiency and minimizing total metabolite load," *PloS one*, vol. 8, no. 9, p. e75370, 2013.

[59] A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, *et al.*, "Lessons learned from quantitative dynamical modeling in systems biology," *PloS one*, vol. 8, no. 9, p. e74335, 2013.

[60] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[61] F. Glover, "Tabu search—part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[62] D. E. Goldberg, *Genetic algorithms*. Pearson Education India, 2006.

[63] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[64] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[65] J.-P. Chiou and F.-S. Wang, "Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process," *Computers & Chemical Engineering*, vol. 23, no. 9, pp. 1277–1291, 1999.

[66] M. O'Neil and C. Ryan, "Grammatical evolution," in *Grammatical Evolution*, pp. 33–47, Springer, 2003.

[67] D. D. McCracken and E. D. Reilly, "Backus-naur form (bnf)," 2003.

[68] A. S. Perelson, A. U. Neumann, M. Markowitz, J. M. Leonard, and D. D. Ho, "Hiv-1 dynamics in vivo: virion clearance rate, infected cell life-span, and viral generation time," *Science*, vol. 271, no. 5255, pp. 1582–1586, 1996.

[69] O. Kotte, J. B. Zaugg, and M. Heinemann, "Bacterial adaptation through distributed sensing of metabolic fluxes," *Molecular systems biology*, vol. 6, no. 1, 2010.

[70] C. Chassagnole, J.-C. A. Rodriguez, A. Doncescu, and L. T. Yang, "Differential evolutionary algorithms for in vivo dynamic analysis of glycolysis and pentose phosphate pathway in escherichia coli," *Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies*, pp. 59–78, 2006.

[71] K. Peskov, E. Mogilevskaya, and O. Demin, "Kinetic modelling of central carbon metabolism in escherichia coli," *FEBS Journal*, vol. 279, no. 18, pp. 3374–3385, 2012.

[72] Y. Cui, *Computational Modeling Techniques for Biological Network Productivity Increases: Optimization and Rate-limiting Reaction Detection*. PhD thesis, Massachusetts Institute of Technology, 2013.

[73] J. Zhao, D. Ridgway, G. Broderick, A. Kovalenko, and M. Ellison, "Extraction of elementary rate constants from global network analysis of e. coli central metabolism," *BMC systems biology*, vol. 2, no. 1, p. 41, 2008.

[74] K. Smallbone, E. Simeonidis, D. S. Broomhead, and D. B. Kell, "Something from nothing- bridging the gap between constraint-based and kinetic modelling," *Febs Journal*, vol. 274, no. 21, pp. 5576–5585, 2007.

[75] K. Smallbone and E. Simeonidis, "Flux balance analysis: a geometric perspective," *Journal of theoretical biology*, vol. 258, no. 2, pp. 311–315, 2009.

[76] R. S. Costa, D. Machado, I. Rocha, and E. C. Ferreira, "Hybrid dynamic modeling of< i> escherichia coli</i> central metabolic network combining michaelis–menten and approximate kinetic equations," *Biosystems*, vol. 100, no. 2, pp. 150–157, 2010.

[77] Y. Usuda, Y. Nishio, S. Iwatani, S. J. V. Dien, A. Imaizumi, K. Shimbo, N. Kageyama, D. Iwahata, H. Miyano, and K. Matsui, "Dynamic modeling of escherichia coli metabolic and regulatory systems for amino-acid production," *Journal of Biotechnology*, vol. 147, no. 1, pp. 17 – 30, 2010.

[78] D. Degenring, C. Froemel, G. Dikta, and R. Takors, "Sensitivity analysis for the reduction of complex metabolism models," *Journal of Process Control*, vol. 14, no. 7, pp. 729–745, 2004.

[79] N. J. Stanford, T. Lubitz, K. Smallbone, E. Klipp, P. Mendes, and W. Liebermeister, "Systematic construction of kinetic models from genome-scale metabolic networks," *PloS one*, vol. 8, no. 11, p. e79195, 2013.

[80] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp, "Ecocyc: a comprehensive database resource for escherichia coli," *Nucleic acids research*, vol. 33, no. suppl 1, pp. D334–D337, 2005.

[81] E. V. Nikolaev, "The elucidation of metabolic pathways and their improvements using stable optimization of large-scale kinetic models of cellular systems," *Metabolic engineering*, vol. 12, no. 1, pp. 26–38, 2010.

[82] F. G. Vital-Lopez, A. Armaou, E. V. Nikolaev, and C. D. Maranas, "A computational procedure for optimal engineering interventions using kinetic models of metabolism," *Biotechnology progress*, vol. 22, no. 6, pp. 1507–1517, 2006.

[83] A. R. Zomorrodi, J. G. Lafontaine Rivera, J. C. Liao, and C. D. Maranas, "Optimization-driven identification of genetic perturbations accelerates the convergence of model parameters in ensemble modeling of metabolic networks," *Biotechnology journal*, 2013.

[84] H. Li, S. Chen, and H. Zhao, "Fractal mechanisms for the allosteric effects of proteins and enzymes.," *Biophysical journal*, vol. 58, no. 5, p. 1313, 1990.

[85] W. Cleland, "The kinetics of enzyme-catalyzed reactions with two or more substrates or products: I. nomenclature and rate equations," *Biochimica et Biophysica Acta (BBA)-Specialized Section on Enzymological Subjects*, vol. 67, pp. 104–137, 1963.

[86] D. Visser, J. W. Schmid, K. Mauch, M. Reuss, and J. J. Heijnen, "Optimal re-design of primary metabolism in escherichia coli using linlog kinetics," *Metabolic engineering*, vol. 6, no. 4, pp. 378–390, 2004.

[87] B. Teusink, J. Passarge, C. A. Reijenga, E. Esgalhado, C. C. van der Weijden, M. Schepper, M. C. Walsh, B. M. Bakker, K. van Dam, H. V. Westerhoff, *et al.*, "Can yeast glycolysis be understood in terms of in vitro kinet-

ics of the constituent enzymes? testing biochemistry," *European Journal of Biochemistry*, vol. 267, no. 17, pp. 5313–5329, 2000.

[88] N. Ishii, K. Nakahigashi, T. Baba, M. Robert, T. Soga, A. Kanai, T. Hirasawa, M. Naba, K. Hirai, A. Hoque, *et al.*, "Multiple high-throughput analyses monitor the response of e. coli to perturbations," *Science*, vol. 316, no. 5824, pp. 593–597, 2007.

[89] S. Vaseghi, A. Baumeister, M. Rizzi, and M. Reuss, "In vivodynamics of the pentose phosphate pathway insaccharomyces cerevisiae," *Metabolic engineering*, vol. 1, no. 2, pp. 128–140, 1999.

[90] D. Visser, R. van der Heijden, K. Mauch, M. Reuss, and S. Heijnen, "Tendency modeling: a new approach to obtain simplified kinetic models of metabolism applied to saccharomyces cerevisiae," *Metabolic Engineering*, vol. 2, no. 3, pp. 252–275, 2000.

[91] N. C. Duarte, M. J. Herrgård, and B. Ø. Palsson, "Reconstruction and validation of saccharomyces cerevisiae ind750, a fully compartmentalized genome-scale metabolic model," *Genome research*, vol. 14, no. 7, pp. 1298–1309, 2004.

[92] T. Lubitz, M. Schulz, E. Klipp, and W. Liebermeister, "Parameter balancing in kinetic models of cell metabolism†," *The Journal of Physical Chemistry B*, vol. 114, no. 49, pp. 16298–16303, 2010.

[93] D. A. FELL and H. M. SAURO, "Metabolic control analysis," *European Journal of Biochemistry*, vol. 192, no. 1, pp. 183–187, 1990.

[94] I. M. Sobol', "On sensitivity estimation for nonlinear mathematical models," *Matematicheskoe Modelirovanie*, vol. 2, no. 1, pp. 112–118, 1990.

[95] H. Rabitz and Ö. F. Aliş, "General foundations of high-dimensional model representations," *Journal of Mathematical Chemistry*, vol. 25, no. 2-3, pp. 197–233, 1999.

[96] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[97] R. Cukier, C. Fortuin, K. E. Shuler, A. Petschek, and J. Schaibly, "Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory," *The Journal of chemical physics*, vol. 59, no. 8, pp. 3873–3878, 1973.

[98] J. H. Schaibly and K. E. Shuler, "Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. ii applications," *The Journal of Chemical Physics*, vol. 59, no. 8, pp. 3879–3888, 1973.

[99] M. Rocha, P. Maia, R. Mendes, J. P. Pinto, E. C. Ferreira, J. Nielsen, K. Patil, and I. Rocha, "Natural computation meta-heuristics for the in silico optimization of microbial strains," *BMC bioinformatics*, vol. 9, no. 1, p. 1, 2008.

[100] N. V. Torres and E. O. Voit, *Pathway analysis and optimization in metabolic engineering*. Cambridge University Press, 2002.

[101] A. P. Burgard, P. Pharkya, and C. D. Maranas, "Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization," *Biotechnology and bioengineering*, vol. 84, no. 6, pp. 647–657, 2003.

[102] C. Chassagnole, B. Rais, E. Quentin, D. Fell, and J. Mazat, "An integrated study of threonine-pathway enzyme kinetics in escherichia coli," *Biochem. J*, vol. 356, pp. 415–423, 2001.

[103] C. Chassagnole, D. Fell, B. Rais, B. Kudla, and J. Mazat, "Control of the threonine-synthesis pathway in escherichia coli: a theoretical and experimental approach," *Biochem. J*, vol. 356, pp. 433–444, 2001.

[104] L. Miskovic and V. Hatzimanikatis, "Production of biofuels and biochemicals: in need of an oracle," *Trends in biotechnology*, vol. 28, no. 8, pp. 391–397, 2010.

CHAPTER 3

CONSTRUCTION OF DETAILED MASS-ACTION ENZYMATIC REACTION SYSTEMS AND CONVERSION OF EXISTING CENTRAL CARBON METABOLISM MODEL OF *ESCHERICHIA COLI*

Novel technological advances in high-throughput experimental methods yielded new quantitative understandings of the phenomena occurring in a cellular system. These data, in conjunction with the increasing availability of mechanistic kinetic descriptions of biological reactions in publicly available databases, has allowed to elucidate and construct more complete mathematical models describing enzymatic reactions mechanistically.

Such descriptions are usually modelled using rate laws that encapsulate and make simplifying assumptions regarding the full set of elementary reaction steps that compose a specific enzymatic reaction. A fully mass-action description of the mechanism accounting for all enzyme complexes may portray better the system under all physiological ranges of participating metabolites in the reaction. However, such characterizations may possess several parameters with a high level of uncertainty due to the lack of experimental data to calibrate them satisfactorily or structural identifiability issues. These levels of uncertainty should be characterized to identify non-influential parameters, as well as the ones which are non-identifiable due to the lack of experimental data to excite the system in a specific way, or the underlying mathematical structure of the system.

In this study we devise a set of methods to regress the parameters of Mass

Action (MA) system based on existing information such as aggregated rate law (ARL) equations, experimental data or existing metabolic models in conjunction with sensitivity analysis to assess the parameters identifiability.

We also describe the construction of a mechanistic model of *Eschericha coli* based in the extension of the existing Kronecker version of central carbon metabolism with more detailed mechanisms and currency metabolites.

This knowledge possesses industrial relevance in the rational design and engineering of microbial strains for the production of industrial relevant compounds.

3.1 Introduction

Until recently, technological limitations made impractical to research cellular processes at a molecular level, making a system wide approach to cell metabolism infeasible. Systems Biology advocates a systems view to model cellular systems [1] by employing quantitative information from reductionist approaches to complement the knowledge of cellular constituents. This creates the possibility of generating, testing and validating new hypothesis using predictive models and experimentation.

Generally, in Bioengineering, cellular systems were modeled as black-boxes hiding the relationships between the different encapsulated systems within the cells. These models try to capture at a higher level the macromolecular reaction scheme occurring in a bioreactor (for instance, in [2] a model of a batch fermentation of baker yeast is presented and in [3] an Ovarian cancer cell in a batch fermentation model is described). These models are geared towards the description of the behavior of the bioreactor operation and macroscopic interaction of the cells.

If one aims at identifying targeted metabolic modifications to combine cell physiology necessities with the desired outputs, then these models are not suitable. Thus, a formalized description of the entities and factors influencing the cellular components of interest is needed.

This problem has been addressed by simulating metabolic models using steady state assumptions, utilizing formalisms that require only stoichiometric information. Due to the limited amount of information to determine the state of the system, simplifying assumptions have to be made. Erroneous results are obtained if these assumptions do not hold in reality (eg. the overall goal of the cell). One of the most widely used methods is Flux Balance Analysis [4] that is expressed as a linear optimization problem where it is assumed that the cell has

a specific goal such as maximizing the growth rate. However, after altering the cellular metabolism it is not guaranteed that the cell will have the same cellular objective as in the wild-type strain. These methods also tend to return a specific flux distribution, while there may be an infinite set of optimal solutions.

Nonetheless, these models have shown good prediction capabilities (e.g. finding essential sets of genes), but often do not enforce enzymatic regulatory restrictions or utilize more limited descriptions such as Boolean models. Thus, some solutions may present discrepancies with reality, due to the incapacity of incorporating non-modeled effects, such as genetic regulation. Other modelling approaches utilize experimental data, such as Metabolic Flux Analysis [5] or Thermodynamic FBA [6] (such as fluxomic and thermodynamic data, respectively) to constrain the possible solution space.

Dynamic models offer a better perspective of the cellular phenomena occurring *in vivo* by allowing to incorporate enzyme regulatory activity besides describing the state of the system temporally and returning a single solution at steady state (without making explicit assumptions about the cellular overall objective).

When contrasted with the aforementioned methods, the system is no longer under-determined. Ordinary differential equation systems (ODEs) are often utilized to model cellular metabolism assuming the intra cellular compartments are in a homogeneous state (thus avoiding the need to track the position of each compound explicitly) and that each specie being modeled possesses a high number of molecules (at least hundreds of units to avert stochastic effects by averaging the compound behavior). On the down side, these models may require detailed enzyme kinetic information and the acquisition of experimental data.

Experimental data acquisition poses distinct levels of difficulty and complexity depending on the type of information being captured. For instance, glucose

pulse experiments (with the collection of samples to characterize a set of metabolite profiles at distinct times) try to acquire a snapshot of metabolism behavior, requiring the use of convoluted experimental techniques with complex experimental protocols that capture data with large experimental errors and the assumption that cells in the sample are in the same metabolic state. Despite these hurdles, several dynamic models of metabolism have been built utilizing glucose pulse experiments [7] [8].

Enzyme kinetics are often represented by recurring to mathematical expressions that give the reaction rate based on the interaction of the distinct compounds participating in the reaction. These expressions can capture different levels of detail concerning the underlying reaction mechanism.

One may divide rate laws into two broad overlapping categories: (i) data base rate laws; (ii) mechanistic rate laws:

- Pure data based reaction rate expressions require a large set of measurements for the full range of metabolite concentrations at distinct conditions, making them infeasible for a large scale model of metabolism. Besides, these types of models do not capture a detailed physical explanation for the observed phenomena, loosing extrapolation capabilities (such as feed foward neural networks [9] and Extreme Learning Machines [10]).

  Other approximations avoid all mechanistic details like linear approximation, power law [11] and Lin-log [12]. However, most of these methods are only applicable in the vicinity of a specific steady state /operational state due to local approximation assumptions. Thus, these approximations are not feasible to study and extrapolate rate changes due to large variations in the metabolite or enzyme concentrations;

- Mechanistic based rate laws are more attractive due to the fact of being

based on the distinct enzymatic reaction steps and possess better extrapolation capabilities. Nevertheless, most mechanistic based kinetic expressions are an aggregation that abridges some of the parameters of the full mechanism, loosing extrapolation capabilities. These type of rate laws are called Aggregated rate laws (ARLs), such as Michaelis-Menten kinetics, Modular rate laws [13].

Fully mass action based rate laws describing the full enzymatic mechanism may not present these issues, but carry more parameters and may need more information to do parameter estimation (taking into account parameter identifiability issues).

These modeling approaches are not mutually exclusive and can be combined. Data based rate laws are usually employed when there is missing mechanistic information regarding certain reactions of interest in the model or there is a specific need to reduce model complexity.

The main fact hampering the construction of detailed dynamic biological models is the insufficiency of data concerning the mechanistic description of enzymatic reactions. Another issue is the fact that *in vivo* and *in vitro* conditions affect acquired experimental data due to distinct conditions inside the cell and the experimental assay (e.g. pH, temperature, etc), that may influence parameter regression. Enzyme kinetic databases [14] [15] often reflect information (e.g. enzymatic reaction mechanisms and parameter values such as Km values) acquired under distinct experimental conditions. Occasionally, this information can be contradictory in the sense that the same enzyme in the same organism may be portrayed as having distinct enzymatic mechanisms. These discrepancies may indicate lack of experimental data to constrain the enzyme behavior (two distinct rate laws may behave similarly in the same region of the concentration space), experimental errors, and identifiability issues concerning the parameters.

The parameters in complex models of metabolism (describing sets of connected reactions) may not be identifiable due to the lack of experimental data availability. The effect of these parameters should be assessed, to characterize with finer granularity reactions that have a large effect on the calibration process and possibly simplify unimportant enzymatic processes, although it is important to note that, under a distinct set of experimental data, a previously deemed unimportant reaction may be classified as important.

It is also crucial to bear in mind that models with distinct structures, assumptions and parametrizations may describe identically the available experimental data.

In this work, we develop several alternative methods to calibrate mass-action descriptions of enzymatic reactions to existing ARL equations, and experimental data. We apply these methods to convert an existing ARL based model of central carbon metabolism of *Escherichia coli* to a mass action format.

These processes were complemented with global sensitivity analysis to identify the reactions that affected the most the calibration process, as well as the individual reaction parameters.

In computational terms, a framework was developed in [16] to take advantage of the underlying characteristics of biochemical systems, represented by mass-action kinetics, called Kronecker. This framework takes advantage of the underlying topological properties of biological networks, namely the low connectivity among interacting components, as well as the mass-action description of reaction kinetics, represented as bimolecular interactions. This allows to represent a biochemical system with sparse matrices that take advantage of special numerical codes. This formalism unifies and standardizes the description of biochemical systems allowing the scalable incorporation of cellular processes for which data is available and the modeler wishes to take into account. In this work

we also extend this formalism to deal with free form symbolic expressions.

3.2 Methods

The goal of this work is to develop a mechanistic model of central carbon metabolism of *E. coli* based on the dynamic model developed by Chassagnole [8] and the existing Kronecker version of the same model [16]. This process rests upon the connection of several identifiability and calibration procedures. In the following subsections the main methods created and utilized during this work for the identifiability analysis and parameter regression are explained.

3.2.1 Calibration Procedures

Several methods can be applied to calibrate the elementary rate laws of mass action rate law (MARL) systems described by ODEs. In this work we treat each reaction calibration independently, taking an divide and conquer approach, to simplify the process. It is assumed that, for each MARL system being regressed there is a corresponding ARL. These regression approaches can be further subdivided, concerning the use of the MARL system integration as part of the method.

The following procedures were utilized in the calibration process of each reaction of the MARL model:

(A) King Altman Method: The base calibration procedure was developed in [16]. The method consists in the conversion of the target reaction MARL reaction being calibrated to an ARL expression by King-Altman (KA) method (following the process described in [17]). Afterwards, the input spaces (concerning the species participating in the reaction) of the MARL system and the corresponding ARL are sampled uniformly within a range

of user defined bounds. The goal of the calibration procedure is to minimize the distance between the rate surfaces of the MARL and the ARL system. In other words, the fitting problem consists in finding the set of elementary rate parameters, such that ARL and MARL produce the same values under distinct metabolite concentration states.

The problem may be described as:

$$min \int_x (r_{ma}(x,\theta) - r_{arl}(x))^2 dx$$

where x is the metabolite state, θ is a set of elementary rate parameters , $r_{ma}(x,\theta)$ represents the MARL expression being calibrated and $r_{arl}(x)$ the ARL. The main advantage of this method is the fact of not requiring the integration of the reaction system. For more complex enzyme systems (with a large number of cycles), it is not possible to calculate the KA rate law, since the computation of all minimum spanning trees in a graph is a NP-Hard problem.

It is important to note that, in the KA ARL the enzyme complexes are assumed to be in steady-state. In this work we take advantage of existing information of the ARLs that are mechanistic based and possess the same assumptions as the KA ARL, by computing the symbolic expression of elementary rate constants corresponding aggregated parameters of the ARL (such as Vmaxs, Kms, Kis, Keqs). These expressions are computed utilizing the Cleland method [18]. Each of these expressions can be utilized as an extra constraint (fixing an elementary rate law) in the calibration process and permits to utilize existing available information. The parameters are fixed based on their degree of identifiability, where the least important parameters are constrained by the values of the aforementioned expres-

sions. The process is explained in the following subsections.

In this work, we utilized two distinct representation formats for the KA ARL depiction (they are equivalent, if the assumptions for the enzyme steady-state hold):

- Symbolic KA ARL: The rate law is derived and the enzyme complex steady state concentrations are implicitly included in the formula.
- Symbolic Derivative Representation: In this scenario, instead of constructing the KA ARL, we represent explicitly the derivative of a product compound of the reaction MARL being calibrated. The enzyme steady-state concentration expressions are derived utilizing the KA method. In this scenario, the calibration problem becomes the minimization of the distance between the product derivative represented by the ARL and the symbolically derived expression for the same compound. More formally, the ARL derivative regarding the product may be described as:

$$\dot{P_{arl}} = Arl(x, \theta_{arl}) \tag{3.1}$$

where $\dot{P_{arl}}$ expresses the derivative of a product of a reaction with an ARL rate law as a function of species concentration of the system x and a set of parameters θ_{arl}. Analogously, the elementary description of the product derivative is given by:

$$\dot{P_{ma}} = MA(x, \theta_{ma}, e_n(x, theta_{ma})) \tag{3.2}$$

where $\dot{P_{ma}}$ defines the derivative of the previously mentioned compound, assuming mass action kinetic description of the system, given

by MA dependent on compound concentrations x with elementary rate parameters θ_{ma}, and a set of functions representing enzyme concentrations (dependent on compound concentrations x and elementary rate parameters θ_{ma}) portrayed by $e_n(X, theta_{ma})$.

Like in the previous methods the calibration problem is given by the minimization of distance. In this particular problem, this is described as:

$$min \int_x (\dot{P_{ma}}(x, \theta_{ma}, e_n(x, theta_{ma})) - \dot{P_{arl}}(x, \theta_{arl}))^2 dx \tag{3.3}$$

This method has the same caveats as the KA method (when it is assumed that enzyme complex concentrations are at steady-state).

(B) Local Simulation (Naive Approach): The MARL system and the ARL are integrated in a predefined time interval within a set of initial conditions (defined by a sampling strategy or by the user), sampled at specific times. The goal in this procedure is to minimize the least squared distance between the original ARL system trajectories metabolites and the MARL system.

(C) Optimal Design of Experiments in conjunction with local sensitivities and hybrid derivative free simulation methods:

The naive approach for calibrating MARL system rate laws utilizing a set of empirically chosen simulations may produce unsatisfactory results due to the lack of informative data to calibrate the model parameters (reducing the MA parameter variability). First, one has to know the number of experiments to perform and the set of initial conditions to stimulate the system in a meaningful way. If such care is not taken into consideration, the cal-

ibration process may produce a good fit but, the rate law will not display the desired behavior when inserted into the original model (by replacing the ARL counterpart and repeating the original experiments in [8]).

To avoid these hurdles, a method based in Design Of Experiments was constructed. The method takes an Evolutionary Computation approach in the design of the experiments, employing Grammatical Evolution (GE) to devise a functional form for an input to excite the system in the most informative way. These ARL systems are often expressed as non-linear functions, thus, it is not known *a priori* how the utilized input signals and their magnitudes will affect the identifiability of the parameters.

The first step in this procedure is to simulate the MARL system utilizing an initial set of parameter values (that may be computed using other calibration methods). Afterwards, the next step is the definition of possible inputs in the MARL model and the values these inputs can take. These inputs are generated by GE as symbolic functional representations. Generally, the inputs of those functions correspond to system variables (species) and time. The GE algorithm evolves the input signal utilizing a context-free grammar in Back-Narus form [19].

These inputs serve as basis to stimulate the original system, the one represented by the ARL rate law. An experiment is designed in the following way: the time interval of the experiment is discretized at pre-defined points; subsequently, for each time point, a relative sensitivity matrix is computed. The global sensitivity of each model parameter is ranked (in descending order), by computing the squared root of the mean squared relative sensitivities. This rank orders the way the columns (that correspond to an ARL model parameter) of the relative sensitivity matrices are utilized iteratively to compute the Fisher Information Matrix (FIM) of the

experiment. It is important to note that the FIM of the experiment is computed by summing the FIMs computed at each discretized time point. The FIM at a specific time point is given by the following expression:

$$FIM = \left(\frac{\partial x}{\partial \theta}\right)^T \left(\frac{\partial x}{\partial \theta}\right) \tag{3.4}$$

The derivative terms $\frac{\partial x}{\partial \theta}$ are computed recurring to the sensitivity polynomial approximation (explained in chapter 2). These relative sensitivity matrices may only contain the observable species in the experiment (rows of the sensitivity matrix) and the set of identifiable parameters (columns of the sensitivity matrix). The ranking of the parameters serves as basis for a local identifiability analysis to construct the experiment FIM. The most sensitive parameter is chosen and the experiment FIM is computed utilizing this parameter, the process is repeated by adding the next highest ranked parameter. When a column (corresponding to a parameter) is added to the FIM construction process and the matrix becomes singular, the parameter is discarded (it is correlated with other parameters already present in the FIM matrix). The FIM is scored based on a optimality criteria. In this work, we utilize the following metric also utilized in [20]:

$$goal_i = \begin{cases} 1, & \text{if } \lambda > 1/\sigma^2_{threshold}. \\ \frac{\frac{\lambda_i}{1}}{\frac{}{\sigma^2_{threshold}}} \times n_\theta, & \text{otherwise.} \end{cases} \tag{3.5}$$

$$Goal = \sum_{i=1}^{n\theta} goal_i \tag{3.6}$$

where *Goal* represents the number of parameters, that are under a predefined uncertainty value, λ_i is the eigen value i of the FIM, and $\sigma_{threshold}$ the

threshold value, to partition parameter counts based on their uncertainty as shown on equation 3.5.

After designing and selecting the best experiment (using GE), a calibration process is carried out utilizing the MARL system and previously designed experiments. Thereafter, the previously designing process is repeated (taking into consideration previously designed experiments), until a termination criterion is met.

(D) Global Simulation: In this method, the target reaction ARL is replaced in the original model by the corresponding MARL ODE system. The elementary rate parameters are then calibrated to the available experimental data by integrating the system and utilizing a weighted least squares approach to minimize the distance between the behavior of the system and the available data. A similar method was used in [21] and the objective function was characterized by least squares weighted by the maximum value of the specified metabolite during the time course.

In the MARL system, the enzyme concentrations can be computed by utilizing expressions like the ones derived by the KA method, if it is assumed that all the enzyme complexes are at steady-state. Another approach for this steady-state assumption is to integrate the system for a "long" time until it reaches the steady-state, and to utilize this state instead as the initial condition.

(E) Direct Partial Global Simulation: To avoid the integration of the full system, only reactions in a predefined radius from the target reaction are considered. The target reaction ARL is replaced in this system representation by the corresponding MARL. Metabolites that interact directly in the target reaction are considered species (except for modifiers - activators and

inhibitors) while the remaining compounds are modelled as inputs (this requires the previous simulation of the original system to represent the inputs in the distinct states). The calibration problem is similar to the previous one but utilizing an abridged model representation. However, if the problem requires the initial conditions to be at steady-state after changing the original system parameters, then the original system has to be simulated first and the state of the system passed to the abridged model representation.

(F) Cha Method: This regression method is analogous to the previous presented global simulation approaches. However, instead of the MARL system, the CHA ARL representation (derived by the Cha method) is utilized [22]. This method can serve as an alternative to KA method to generate ARL representations. The Cha method produces simpler ARLs than the KA method, but it requires the assumption that a set of elementary reaction steps are in rapid equilibrium, while another group of reactions occurs at lower rates (often named slow steps). The rapid-equilibrium assumption is more limiting to enzyme representation than the premise of KA of the steady-state of enzyme compounds, thus losing prediction capabilities when the rapid equilibrium assumptions do not hold. These CHA ARL expressions have to be calibrated to existing experimental data or to pseudo-experimental data using a similar method to the previously described global calibration procedures.

It is important to note that the first method is derivative free, the DOE based approach integrates the original system during the DOE phase, but may avoid the integration of the MARL system by utilizing a collocation point strategy (an algebraic system that captures de simulation like in [23]), while the remaining

methods require the full simulation of the system.

The KA and DOE based methods try to approximate the rate surface described by the elementary mass action system, assuming that the ARL is the true characterization of enzyme rate behavior, while the remaining procedures minimize the distance to the available experimental data. If the amount of information in these data is not sufficient to discriminate among distinct ARLs, then multiple descriptions may be utilized to describe the enzymatic reaction activity, which may affect significantly the extrapolation capabilities.

3.2.1.1 Objective Functions

For each of the aforementioned procedures, one of the following objective function is considered:

- When data is noise-free:

$$J_1(\theta) = \sum_{i=1}^{nTimePoints} \frac{(f(x,\theta,t_i) - f(x,t_i)_{arl})^2}{(max(f(x,t_i)_{arl}))^2} + \sum_{i=1}^{nTimePoints} \frac{(f(x,\theta) - f(x,t_i)_{arl})^2}{(max(1, f(x,t_i)_{arl}))^2} \tag{3.7}$$

  where $f(x,\theta,t)$ corresponds to a function of metabolite x concentrations that can either be the concentration itself or flux values, θ parameter values, and possibly time t if the problem is time dependent. The first summation in time dependent problems describes the sum of the two terms over all experimental time points (including different experiments). In non time dependent problems the summation is over all the cases considered in the specific procedure - For instance, in the KA calibration method, the summation corresponds to all distinct metabolite states considered.

The first term in $J_1(\theta)$ treats all values equally in their own scale regarding the specific metabolite concentration or flux error due to the division by the corresponding squared maximum value. The second term corresponds to a relative squared error, except when the measured values are below 1, where the squared error is considered instead.

- When data is not perfect (contains noise):

$$J_2(\theta) = \sum_{i=1}^{nTimePoints} \frac{(f(x,\theta,t_i) - f(x,t_i)_{measured})^2}{(\sigma_{x(t_i)})^2} \tag{3.8}$$

The symbolic specification is analogous to the previous objective function, except for $(\sigma_{f(t_i)})^2$ that represents the variance of $f_{measured}(x,t_i)$. This objective function is similar to Maximum Likelihood estimation when the errors are normally distributed.

In both cases, symbolic derivatives are utilized by the MATLAB [24] fmincon optimization function.

3.2.1.2 Calibration Data

The previously described calibration procedures that require the use of experimental data were executed in two distinct settings: (i) calibration against only available experimental data; (ii) calibration against pseudo experimental data generated by the original model. The simulated data mimics the original pulse experiment, but captures all the metabolite profiles at equally spaced time intervals (defined at 0.1 seconds during the 40 seconds of the pulse experiment). In this work reaction rate mechanisms were calibrated using pseudo experimental data generated from strategy (ii).

3.2.2 Optimization Algorithms

The aforementioned regression problems were solved utilizing an hybrid global optimization approach. Empirically, the first main goal is to set the parameters in a good region of the search space. Thus, a global optimization algorithm is utilized. In this work, Hybrid Differential Evolution as described in [23] was used (and explained in Chapter 2). This algorithm is based on DE, however it employs derivative information and a smaller population to alleviate the computational burden. Afterwards, a multi-start process is carried with Matlab [24] fmincon with active-set algorithm to find a local optimal solution. When it was infeasible to utilize fmicon, fminsearch was utilized instead. The algorithms parametrization are given in appendix.

3.2.3 Parameter Identifiability

Different identifiability strategies were followed for distinct calibration procedures due to the computational burden associated with each method. The Design Of Experiments process already possesses an incorporated identifiability scheme.

3.2.3.1 King Altman Method

The first step corresponds to the calibration of the King-Altman rate law, utilizing the previously described optimization algorithms. Afterwards, the standardized Morris method [25] was utilized to rank the parameter total sensitivity, assuming a deviation of 50% for each elementary rate concerning the first calibration. The number of r-levels employed was chosen based on the method developed by [26]. The parameters deemed as more robust where fixed based on the previously computed ARL coefficient expression computed using Cleland

method [18]. It is important to note that, if more sensitive parameters are fixed instead, the calibration process may fail.

3.2.4 Remaining Calibration Methods

A similar strategy to the one employed in [21] was utilized in this work. For each of the MARL regressed using this method, 10 runs where executed using the aforementioned optimization algorithms. For the best solutions of each run, a local sensitivity analysis based on the construction of the FIM was carried (analogous to the a priori sensitivity analysis explained in the previous sections). The parameters of these solutions with higher sensitivity were regressed, while the remaining were fixed based on literature values or ARL coefficients computed utilizing the Cleland method. It may not be possible to compute these expressions if the ARL does not have a mechanistic representation or the enzyme complexes are not under the steady-state assumption.

It is important to note that unidentifiable elementary rates may be sensitive in a narrower finite interval than the physiological range originally considered.

3.2.5 MARL Model Composition

After regressing a MARL reaction the calibration is tested by replacing the original ARL in the Chassagnole model [8] and computing the Mean Squared Average error against the experimental data concerning the pulse experiment used originally to calibrate that model. If the value of this goodness of fit metric is above a threshold, the calibration is discarded. Otherwise, the result is kept.

3.3 Results and Discussion

In this work, we developed a model of *Escherichia coli* central carbon metabolism based on previously developed models, namely the one in [8] and the Kronecker version of the same model with constant co-metabolites [16].

The approaches and corresponding results developed in this chapter are presented in the following subsections. First, the Kronecker symbolic extension is described. Afterwards, the model construction is characterized. Next, we analyse the local parameter sensitivity of the original model, as well as the robustness of a set of parameters connected to the rate law of each reaction. These analyses served as basis to delineate model simplifications, and are described in the following subsection. Subsequently, the calibration and identifiability analysis of the MARL representations for each reaction are outlined. Next, the differences between the aforementioned regression procedures are contrasted and highlighted.

3.3.1 Kronecker Symbolic Extension

The Kronecker formalism serves as the low-level representation of the systems developed during this work. This abstraction was extended to contemplate the utilization of free form symbolic expressions, thus permitting the incorporation of other types of rate equations, such as ARLs. This expansion was achieved by introducing functions that given the state of the system, the parameters and the inputs return the respective matrix based on symbolic rate expressions. In the scope of this expansion a new model format (and the respective conversion tool for existing file formats) was created to allow the development of cellular models in a modular fashion like in an imperative programming language (more details about this file format are given in Chapter 7). It is important to note

that a similar extension was developed simultaneously at MIT by David Hagen (member of Tidor's Lab) utilizing Matlab symbolic representation. In this work an interpreter and abstract symbolic representation were built from ground up in Scala. The Kronecker formalism is explained thoroughly in Chapter 2 (consult this chapter for the definitions).

The function $f(x, u, \theta)$ becomes:

$$f(x, u, \theta) = A1x + A2x \otimes x + B1x + B2x \otimes x + k + AS(x, u, \theta) \tag{3.9}$$

where $AS(x, u, \theta)$ is a function that receives as input the system species concentrations, the inputs, the parameters and returns a matrix $(n_X \times 1)$. The result for each position in this vector is symbolically computed. Identically, the reaction rate is given by:

$$r(x, u, \theta) = RA1x + RA2x \otimes x + RB1x + RB2x \otimes x + rk + RS(x, u, \theta) \tag{3.10}$$

where $RS(x, u, \theta)$ is a function that receives as input the system species concentrations, the inputs, the parameters and returns a matrix of fluxes $(n_r \times 1)$. For each derivative of mass-action and flux matrices, a function exists that returns the adequate vector.

The symbolic part of these functions can be precomputed. To reduce the computation time of the derivative matrices, only expressions containing the species or parameters being part of the derivative should be processed. In this work, these computations were performed by implementing a symbolic expression tree and respective methods for derivation and symbolic expression simplification [27].

This format also allows to express the following partial derivatives (where the under script denote the derivative variables) as:

$$f_x = A_1 + A_2(I \otimes x + x \otimes I) + B_2(I \otimes u) + AS_x(x, u, \theta) \tag{3.11}$$

$$f_\theta = A_{1,\theta}x + A_{2,\theta}(x \otimes x) + B_{1,\theta}u + B_{2,\theta}(x \otimes u) + k_\theta + AS_\theta(x, u, \theta) \tag{3.12}$$

$$f_{xx} = 2A_2 + AS_{xx}(x, u, \theta) \tag{3.13}$$

$$f_{\theta x} = A_{1,\theta} + A_{2,\theta}(x \otimes I + I \otimes x) + B_{2,\theta}(Ix \otimes u) + AS_{xp}(x, u, \theta) \tag{3.14}$$

$$f_{\theta\theta} = AS_{\theta\theta}(x, u, \theta) \tag{3.15}$$

$$r_x = RA_1 + RA_2(I \otimes x + x \otimes I) + RB_2(I \otimes u) + RS_x(x, u, \theta) \tag{3.16}$$

$$r_\theta = RA_{1,\theta}x + RA_{2,\theta}(x \otimes x) + RB_{1,\theta}u + RB_{2,\theta}(x \otimes u) + rk_\theta + RS_\theta(x, u, \theta) \tag{3.17}$$

3.3.2 Model Description

The MARL version of the Chassagnole model of the Central Carbon Metabolism of *Escherichia coli* [8] devised in this work is composed by the Phosphotrans-

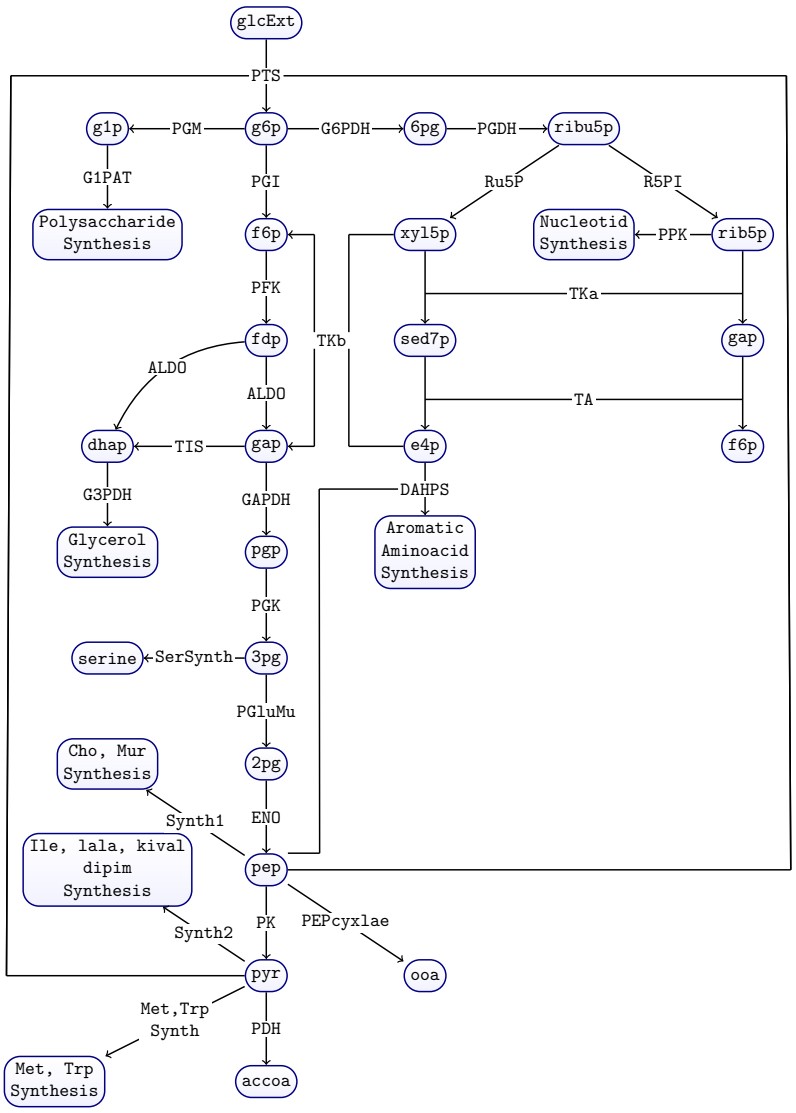


Figure 3.1: *Escherichia coli* Central Carbon Metabolism Model Graphical Representation.

ferase system, Glycolysis and the Penthose-phostphate pathway, possessing the same reactions as the original model, delineated by mass-action descriptions, as shown on Table 3.1. In this version of the model, co-metabolites are represented explicitly, while their behavior is depicted by time dependent functions like in the original model. In Figure 3.1 a graphical description of the system is shown.

Table 3.1: Model Reactions

Reaction	EC Number	Reaction	Activators	Inhibitors	Mechanism
PTS[a]	-	$glc + pep \rightarrow g6p + pyr$		g6p	bi-bi irreversible
PGI[a]	5.3.1.9	$g6p \leftrightarrow f6p$		pg	revesible uni-uni
PFK[d]	2.7.1.11/44	$f6p + atp \rightarrow fdp + adp$		pep/adp	allosteric model (n = 4)
ALDO[a]	4.1.2.13	$fdp \leftrightarrow gap + dhap$			ordered uni-bi
TIS[a]	5.3.1.1	$dhap \leftrightarrow gap$			reversible uni-uni
GAPDH[a]	1.2.1.12	$gap + nad \leftrightarrow pgp + nadh$			reversible bi-bi
PGK[a]	2.7.2.3	$pgp + adp \leftrightarrow pg3 + atp$			reversible bi-bi
PGluMu[a]	5.4.2.1	$pg3 \leftrightarrow pg2$			reversible uni-uni
ENO[a]	4.2.1.11	$pg2 \leftrightarrow pep$			reversible uni-uni
PK[d]	2.7.1.40	$pep + adp \rightarrow pyr + atp$	fdp		allosteric regulation
PDH[a]	1.2.4.1	$pyr \rightarrow \emptyset$			hill equation (n = 3)
PEPCxylase[a]	4.1.1.31	$pep \rightarrow oaa$	fdp		hill equation (n = 4)
PGM[a]	5.4.2.2	$g6p \leftrightarrow g1p$			reversible uni-uni
G1PAT[a]	2.7.7.27	$g1p + atp \rightarrow polysac + adp$			irreversible bi-bi
G3PDH[a]	1.1.1.94	$dhap \rightarrow glycerol$			irreversible bi-bi
SerSynth[a]	-	$pg3 \rightarrow ser$			irreversible uni-uni
MurSynth[a]	-	$f6p \rightarrow murine$			irreversible uni-uni
DAHPS[a]	2.5.1.54	$pep + e4p \rightarrow aromaticaminoacids$			irreversible bi-uni
TrpSynth[a]	-	$\emptyset \rightarrow pyr + gap$			steady-state flux
MetSynth[a]	-	$\emptyset \rightarrow pyr$			steady-state flux
G6PDH[b,c]	1.1.1.49 / 3.1.1.31	$g6p + nadp \rightarrow pg + nadph$			Irreversible uni-uni
PGDH[b,c]	1.1.1.44	$pg + nadp \leftrightarrow rib5p + nadph$			reversible bi-bi
Ru5p [a]	5.1.3.1	$ribu5p \leftrightarrow xyl5p$			reversible uni-uni
R5P1[a]	5.3.1.6	$ribu5p \leftrightarrow rib5p$			reversible uni-uni
Tka[a]	2.2.1.1	$xyl5p + rib5p \leftrightarrow gap + sed7p$			reversible bi-bi
TKb[a]	2.2.1.1	$e4p + xyl5p \leftrightarrow f6p + gap$			reversible reversible bi-bi
TA[a]	2.2.1.2	$sed7p + gap \leftrightarrow f6p + e4p$			reversible bi-bi
Synth1[a]	-	$pep \rightarrow \emptyset$			irreversible uni-uni
Synth2[a]	-	$pyr \rightarrow \emptyset$			irreversible uni-uni
RPPK[a]	2.7.6.1	$rib5p \rightarrow nucleotide$			irreversible uni-uni

[a] Chassagnole et al. (2002) [8]
[c] Ecocyc database [28] (as consulted in 2013)
[d] Zhao et al. (2008) [21]

The ARLs of the original model can be converted to a bimolecular mass action representation by decomposing the mechanistic rate law in the respective mass action reaction scheme that originated it. One problem that can arise is the fact that the original reaction scheme may possess more elementary rate parameters than the parameters in the mechanistic rate law. Thus, it is not possible to do a one-to-one mapping of elementary rates and parameters due to the fact of this system being under-determined. Thus, several sets of parameters may produce similar results. This issue may be mitigated by fixing parameters based on domain knowledge.

For instance, it can be assumed that the *kon* elementary rate of a reaction mechanism has a fixed rate $1E6s^{-1}$ value [16] due to the physical principles regarding protein diffusion limits. However, this value may have to be modified on a reaction mechanism basis. These values should also be updated if there is experimental data available to calibrate them. Another assumption that has to be taken into account is the enzyme complexes concentration. It can be assumed, depending on the modeling task, that total enzyme concentration (including the free enzyme and the bound complexes) remain constant over time. In [16], the total enzyme concentration concerning one reaction was assumed to be $0.001mM$ based in the average enzyme concentration in a bacterial cell.

If the system is under-determined the remaining rates can be calculated by using non-linear least squares in conjunction with one of the aforementioned methods to compute the corresponding elementary rates of the mass action reaction mechanism. It is important to bear in mind that due to the non-linear nature of some rate laws there might be distinct sets of rates that produce the same output.

3.3.3 Prior Sensitivity Analysis

We executed a local sensitivity analysis of the Chassagnole model, by computing the square root of the sum of the squared relative sensitivity values for each parameter at the time points of the pulse experiment described in [8]. This sensitivity metric can be expressed as:

$$\delta^{msqr} = \frac{1}{n_{Species}} \sum_{j=0}^{n_{Species}} \sqrt{\frac{1}{n_{timePoints}} \sum_{i=0}^{n_{timePoints}} \left(\frac{dln(x_{i,j})}{dln(\theta)} \right)^2} \tag{3.18}$$

where $n_{timePoints}$ represents the total number of time points considered, x_j the specie j concentration at time i. The parameters are ranked in descending order

Table 3.2: Model Parameter Ranking - δ^{msqr}

Rate Equation	Parameter	δ^{msqr}
rPts	nPTSg6p	98.76
rPfk	nPFK	97.86
rPfk	KPFKf6ps	74.06
rPgi	KPGIeq	73.68
rPfk	KPFKpep	24.24
rPdh	nPDH	23.99
rRpglumu	KPGluMueq	18.44
rEno	KENOeq	18.40
rPts	rmaxPTS	17.04
rPts	KPTSg6p	16.92
rGapdh	rmaxGAPDH	16.86
rGapdh	KGAPDHgap	16.83
rGapdh	KGAPDHpgp	16.78
rPgk	KPGKeq	15.63
rPts	KPTSa1	15.18
rMursynth	rmaxMurSynth	0.0307
rRpglumu	KPGluMupg2	0.029
rTa	rmaxTA	0.0288
rTkb	rmaxTKb	0.0252
rPgi	KPGIf6p	0.0243
rPgi	KPGIf6ppginh	0.0203
rTka	rmaxTKa	0.0110
rDahps	nDAHPSpep	5.857E-4
rPk	nPK	4.27E-4
rDahps	KDAHPSpep	2.83E-4
rPk	KPKamp	6.390E-5
rPts	KPTSa2	3.517E-5
rPk	KPKfdp	2.841E-5
rPk	LPK	2.685E-5
rPk	KPKatp	1.241E-5

based on the value of this metric.

In Table 3.2, the fifteen most important, and the fifteen least important parameters of the model (classified by the sum of relative sensitivities) are shown. The top two most important parameters are related with the number of subunits

in the PTS reaction as well as in the PFK reaction.

The bottom five parameter set in Table 3.2 contemplates four parameters from the PK reaction. It is important to bear in mind that these results are related to the experimental data utilized to calibrate the model extracted from [8]. With other stimuli the PK rate law parameters may have higher relative sensitivities values. As noted in [29], a parameter with low overall relative sensitivity value may possess an impact in the calibration process and may not be discarded.

Another relevant question in this context is which parameters may effectively be estimated from the available data, considering the original parameter set as the nominal parameter set. It is important to note that the method used to perform this analysis devised in [30] is a local method that returns a near optimal distinguishable parameter set based on available information (the least correlated parameter set up to a cut-off value - in this work the method was applied with the most stringent cut-off value described in the article of 0.3).

From Table 3.3, it can be seen that six out of the fifteen classified as most important parameters can be estimated from the data (namely the two most important parameters), while no parameters from the least important parameter set can be classified as estimable by the aforementioned method.

A global sensitivity analysis was also applied to identify which reactions influence the most the calibration process (Maximum Likelihood estimation). The reaction global sensitivity was assessed by executing Morris method with groups as explained in [31]. Each group consisted of the particular set of parameters corresponding the a specific reaction ARL. Each parameter was allowed to vary 50% relative to its nominal value. In Figure 3.2 a bar chart is shown, where each bar corresponds to the mean absolute elementary group value for each reaction. The three most important reaction parameter groups correspond to four of the most important parameters in the Table 3.2. The grouping of parameters

Table 3.3: Model Estimable Parameters

Rate Equation	Parameter
rAldo	kALDOeq
rTis	kTISeq
rPgk	KPGKeq
rPfk	nPFK
rEno	KENOeq
rPts	nPTSg6p
rPepcxylase	rmaxpepCxylase
rG1pat	rmaxG1PAT
rSersynth	rmaxSerSynth
rSynth1	rmaxSynth1
rG3pdh	rmaxG3PDH
rPpk	rmaxRPPK
rMursynth	rmaxMurSynth
rSynth2	rmaxSynth2
rRpglumu	KPGluMueq
rPgi	KPGIeq
rDahps	nDAHPSe4p
rTkb	KTKbeq
rPdh	nPDH
rRu5p	KRu5Peq
rTka	KTKaeq
rG1pat	nG1PATfdp
rR5pi	KR5PIeq
rGapdh	rmaxGAPDH
rG6pdh	rmaxG6PDH
rPepcxylase	KpepCxylasefdp
rPgm	KPGMeq
rPfk	KPFKpep
rPepcxylase	npepCxylasefdp
rTa	KTAeq
rPgdh	rmaxPGDH

shadows the importance of each parameter individually, nonetheless it requires less computation power (the number of simulations required is equal to a constant multiplied by number of groups instead of constant multiplied by number of parameters).

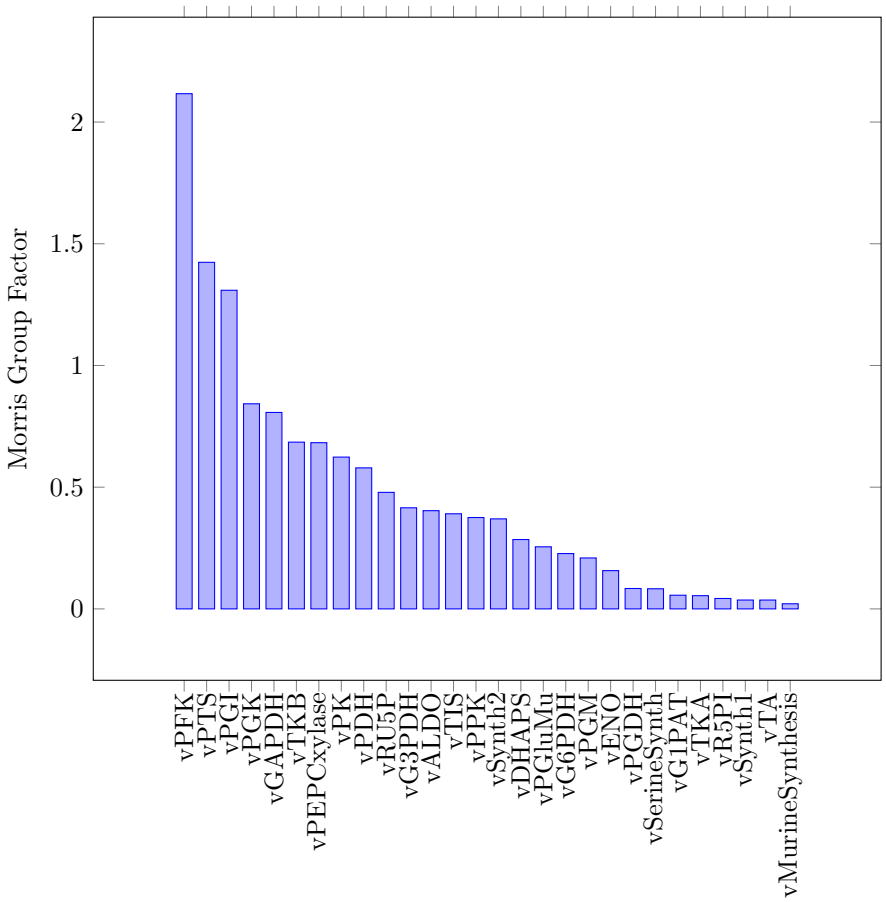


Figure 3.2: Morris Method with groups - Chassagnole model.

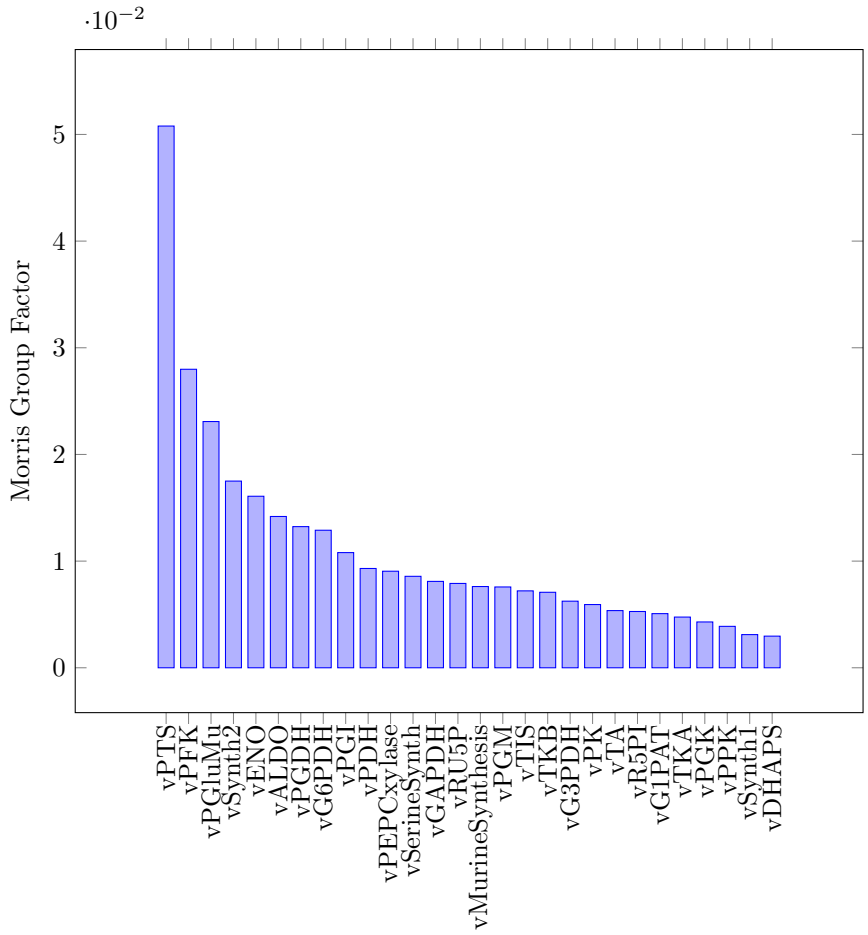


Figure 3.3: Morris Method with groups analysis regarding equivalent topological LinLog model.

To evaluate the effects of the absence of enzymatic regulation and simpler kinetic representation, an equivalent Lin-log model in terms of topology and initial state was built using default parameters (without calibration), with elasticities equal to the negative value of the corresponding stoichiometric coefficient like in [12] [32] [33]. The previous Morris method by group analysis was carried in this model. In Figure 3.3, the results are shown. The two most important reactions are classified as in the original model leading to the question if this effect

is solely caused by the topology of the network.

However, it is important to note that these results are only concerned with the pulse experiment data. In this analysis, the co-metabolites are modeled like in the original model as time dependent functions.

3.3.4 MARL Model Simplifications

In Table 3.1, the MARL utilized for each reaction are shown. The reaction mechanisms where updated based on the work described in [21] and on the previous *a priori* sensitivity analysis.

Several simplifications were adopted, such as the representation of Monod-Wyman-Changeux model (MWC) reactions with Cha method, where the relaxed free enzyme and the tense state forms are treated as a single species - with the tense transitions assumed to be in rapid equilibrium. Enzyme conformational changes were also modelled by assuming non-elementary rate equations with non-integer coefficients [21] for PK and PFK reactions.

The PDH MARL was represented as a mass-action elementary reaction with fourth order in the first elementary reaction (when pyr binds to the PDH free enzyme).

These simplifications were made to reduce the model complexity and ease the calibration of the model parameters.

3.3.5 Calibration Methods

Table 3.4: Model Calibration Procedure

Reaction	Calibration Procedure
PTS	KA;GS;CHAGs
PGI	KA;GS;CHAGs
PFK	KA;GS;CHAGs
ALDO	KA;GS;CHAGs
TIS	KA;GS;CHAGs
GAPDH	KA;GS;CHAGs
PGK	KA;GS;CHAGs
PGluMu	KA;GS;CHAGs
ENO	KA;GS;CHAGs;FIMGE
PK	KA;GS;CHAGs
PDH	KA;GS;CHAGs
PEPCxylase	KA;GS;CHAGs
PGM	KA;GS;CHAGs
G1PAT	KA;GS;CHAGs
G3PDH	KA;GS;CHAGs
SerSynth	KA;GS;CHAGs
MurSynth	KA;GS;CHAGs
DAHPS	KA;GS;CHAGs
TrpSynth	-
MetSynth	-
G6PDH	KA;GS;CHAGs
PGDH	KA;GS;CHAGs
Ru5p	KA;GS;CHAGs
R5P1	KA;GS;CHAGs
Tka	KA;GS;CHAGs
TKb	KA;GS;CHAGs
TA	KA;GS;CHAGs
Synth1	KA;GS;CHAGs
Synth2	KA;GS;CHAGs
RPPK	KA;GS;CHAGs

The nomenclature utilized in this table to describe the utilized calibration procedure is the following: KA - King-Altman method; CHAGs - Cha Method with original model simulated data; GSExp - Global Simulation Method with Experimental Data; GS - Global Simulation Method with original model simulated data; DOEGE - FIM based method executed with Grammatical Evolution

In Table 3.4 the employed calibration methods for each reaction are shown. In all the calibration processes except for the DOE simulation, a hybrid approach was followed. First, HDE (described in chapter 2) would be utilized to kick-start the optimization process followed by Fmincon with active-set algorithm (with

explicit usage of parameter derivatives). Such choice of algorithms is purely empirically, pursuing the rational that the global optimization algorithm will be used to explore the search space and possibly guide the search near optima, followed by a gradient based method to reach the optima.

We focused our work in the construction of MA descriptions of the reactions (including co-metabolites) in the *Escherichia coli* model of Central Carbon Metabolism delineated by Chassagnole [8]. The devised MA model contains parameters, many of those unidentifiable due to the limited amount of data utilized in the calibration process, as shown in the next subsection.

The internal kronecker structure of the system had to be modified, to speed up the computation of the system derivatives. All the derivatives were expressed as symbolic expressions to avoid the computation overhead of re-constructing Matlab matrices.

3.3.5.1 Calibration Procedures Applied to One Reaction

In this subsection a distinct set of calibrations utilizing distinct procedures presented previously, regarding the ENO reaction (due to its simplicity it is modelled as a uni-uni reaction without any modifiers) are displayed. The calibration result for the remaining reactions is in appendix A.0.1. In the following figures the results obtained with the distinct methods are shown:

In Figure 3.4 the rate vs subtrate for the ENO reaction is shown. The dashed red line represents the MARL mechanism and is on top of the ARL mechanism represented by the solid blue line.

In Figure 3.5 the CHA rate parameters were regressed utilizing CHA ARL rate equations against the experimental data. Afterwards, a similar process to KA method was utilized to calibrate the derived MARL system based on the previously regressed CHA ARL. In Figure 3.6 the effect of replacing the original

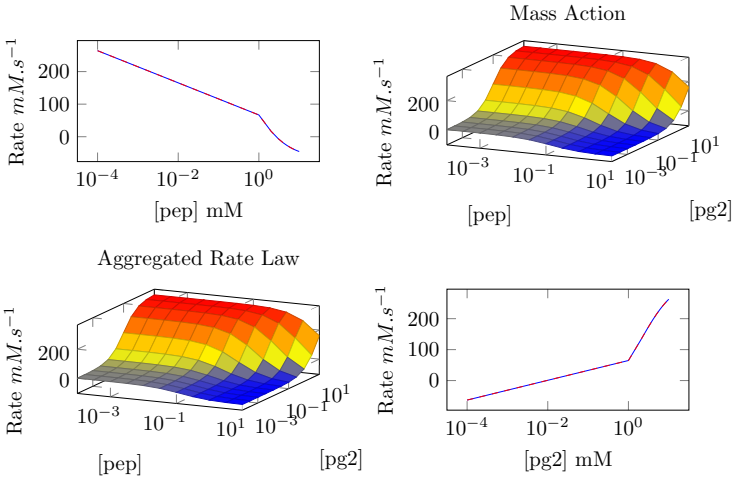


Figure 3.4: KA Calibatration -Rate vs Specie plot of regression of ENO MARL to the respective ARL.

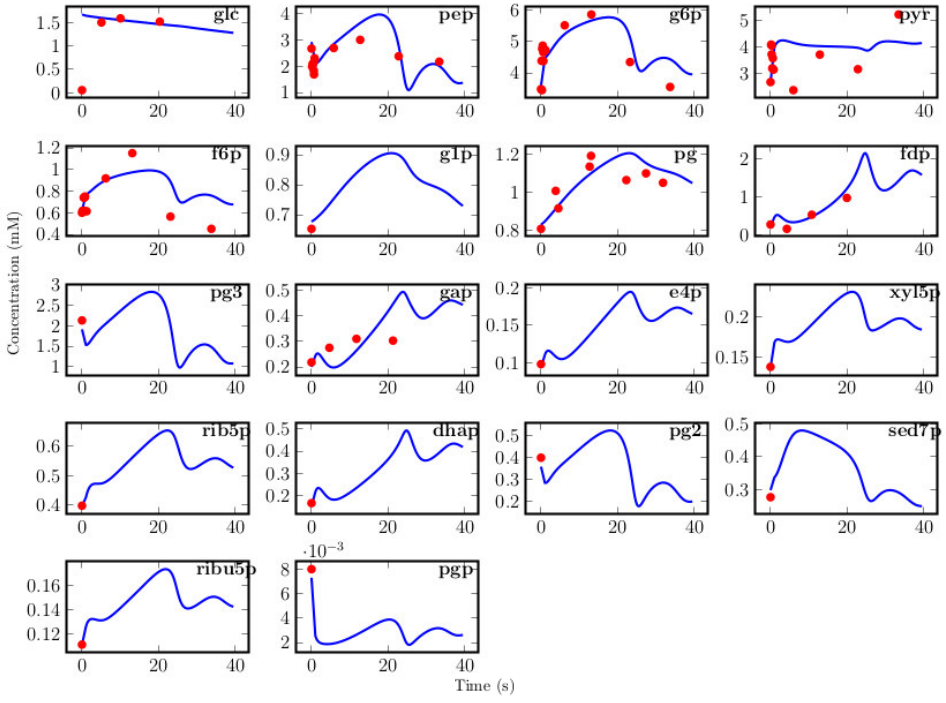


Figure 3.5: CHA ARL ENO Calibration - Simulation of the full system with ENO ARL replaced by the CHA equivalent against experimental data.

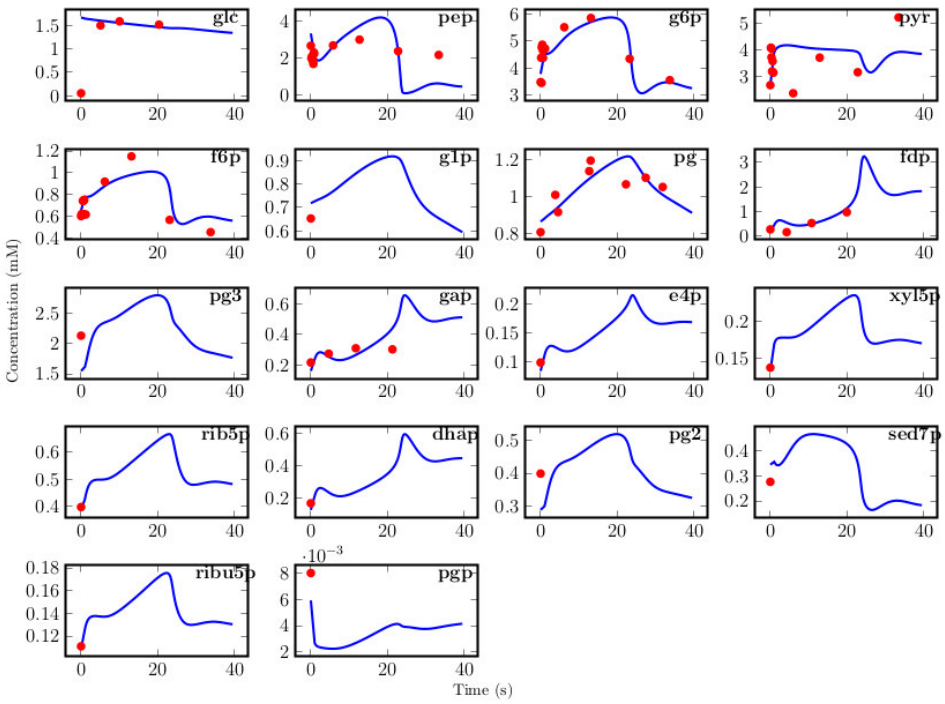


Figure 3.6: CHA ENO MARL Calibration - Simulation of the full system with ENO ARL replaced in the original model by the CHA equivalent against experimental data.

ENO ARL by the calibrated ENO CHA MARL system is shown. In Figure 3.7 the result of the Global ENO MARL calibration without recurring to the previous calibration of the respective CHA ARL is displayed.

The Grammar in Back-Narus form utilized by the GE algorithm is shown below:

```
Start {'['<SpecieList>']'};
SpecieList { '(pep:'<Value>')[(pg2::'<OperationPep>')]'|
'(pg2:'<Value>')[(pep::'<OperationPg2>')]'};
OperationPep { <OperationPep> <BinOp> <OperationPep>
              | <UniOpPep> |<ValuePep> };
OperationPg2 { <OperationPg2> <BinOp> <OperationPg2>
```

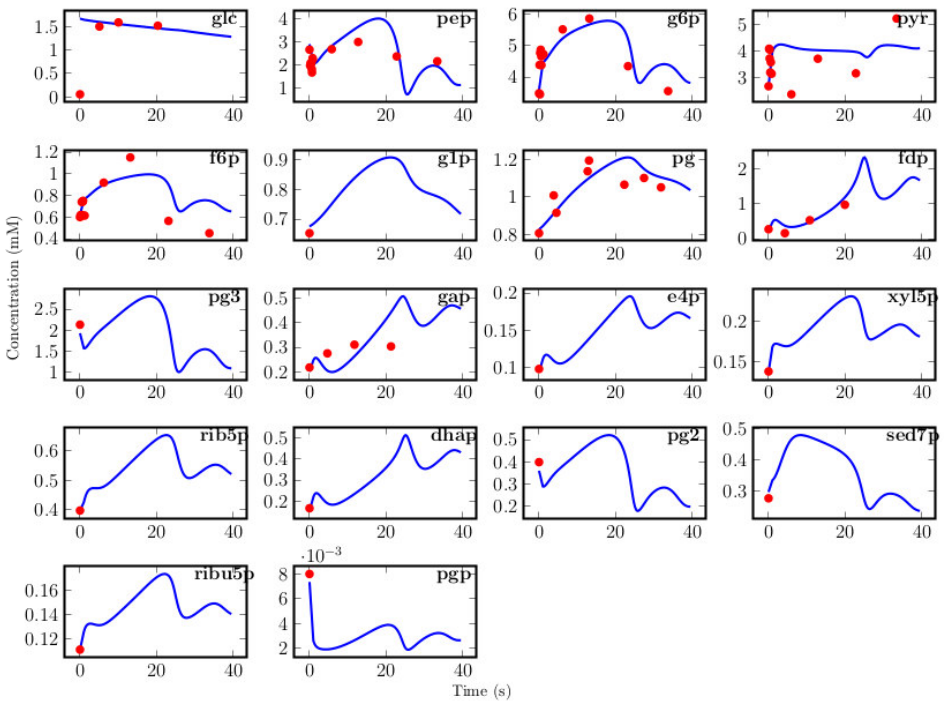


Figure 3.7: Global ENO MARL Calibration - Simulation of the full system with ENO ARL replaced by the CHA equivalent against experimental data.

```
                | <UniOpPg2> |<ValuePg2> };
ValuePep {<Number> | <Number>'.'<Number> | 'pep' | 'time' };
ValuePg2 {<Number> | <Number>'.'<Number> | 'pg2' | 'time' };
UniOpPep {'cos('<OperationPep>')'| 'sin('<OperationPep>')'
         |'ln('<OperationPep>')' | '('<OperationPep>')^('<OperationPep>')' };
UniOpPg2 {'cos('<OperationPg2>')' | 'sin('<OperationPg2>')'
         |'ln('<OperationPg2>')' | '('<OperationPg2>')^('<OperationPg2>')' };
BinOp {'+' | '-' | '/' | '*'  };
Value {<Number> | <Number>'.'<Number> };
InitialValue {'0.'<Number>|'1.'<Number>|'2.'<Number>
              |'3.'<Number>|'4.'<Number>|'5.'<Number>};
Number {'1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | '0' | <Number>}
```

One of the species of the reaction is utilized as input of the experiment as it can be seen in the rule SpecieList. The other specie serves as input function that can have any valid mathematical form composed from the pre-specified operations and the other specie as input as well as the time.

The initial ENO mechanism was calibrated against a simulation of the original system with both specie values set to the double of their respective Km from the original ARL. The inputs devised by GE in the calibration process (in chronological order for each experiment):

1.
$$pep = 2.4mM$$

2.
$$pg2 = 7.6mM$$

3.
$$pg2 = cos(sin(3.0))^{time} mM$$

The goal function threshold was set to consider a parameter as estimated when the parameter estimated value had less than 50% variation regarding the original nominal value.

In the Figures 3.8 and 3.9 the influence of the calibration process can be seen as it the MARL converges to the ARL surface. In Figure 3.10 the simulation of the MARL system from steady-state against the original model is shown. The solid lines represent the ARL, the x represent the devised KA rate law, and the circles the MARL model.

The devised regression methods can be utilized in the following order to calibrate an arbitrary MARL expression:

- KA/Derivative: Both are equivalent methods, with distinct symbolic representations, that require the computation of all the minimum spanning trees from the

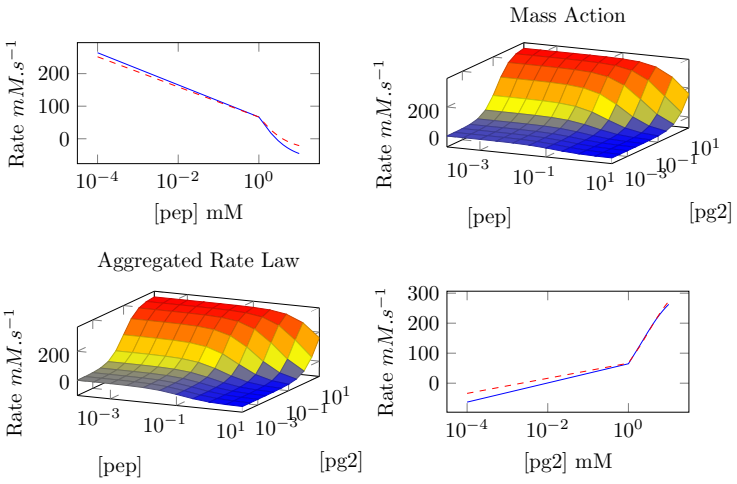


Figure 3.8: DOE Calibatration -Rate vs Specie plot of regression of ENO MARL to the respective ARL, using data from the first and second inputs devised by GE in the calibration process in conjunction with the initial simulation data.

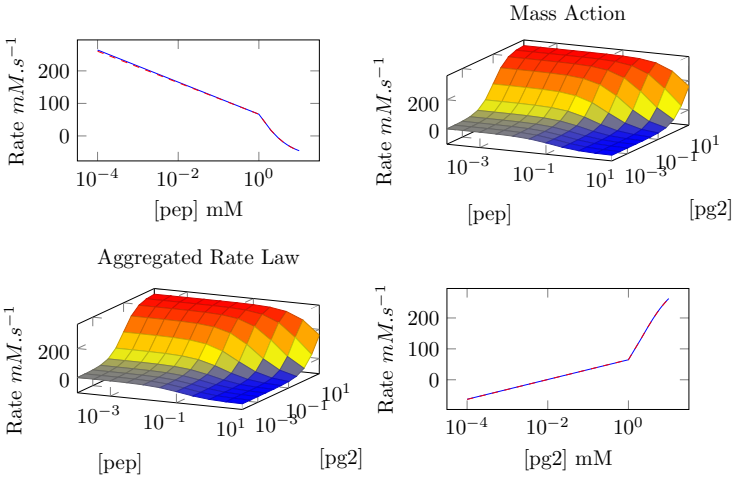


Figure 3.9: DOE Calibration - Rate vs Specie plot of regression of ENO MARL to the respective ARL using data from the all experiments and simulations.

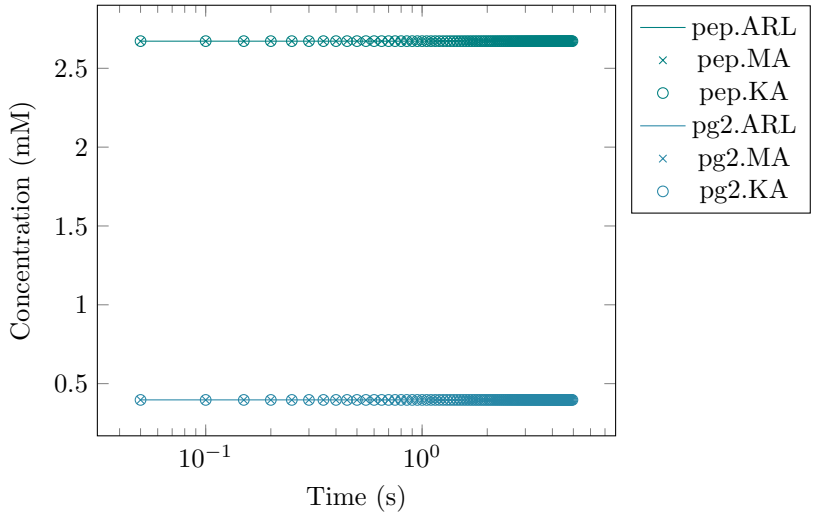


Figure 3.10: ENO simulation from initial conditions after the calibration process of DOE.

graph representation of the enzyme mechanism, what is a np-hard problem. For more complex enzyme mechanisms, these methods are not suitable;

- Cha(Global Calibration): To overcome the issues with KA, further assumptions can be made, such as the existence of rapid-equilibrium elementary reaction steps what gives origin to simplified KA mechanisms and simpler rate expressions. However, care must be taken in the applicability of those assumptions. The enzyme fractional expressions (consult [22]) generated by this method can also be applied in the derivative method but if the previously mentioned assumptions are not met, the calibration process tends to converge to worse values;

- DOE: The DOE based calibration method has incorporated an identifiability process based on a local sensitivity approach. The devised process may also avoid the integration of the MARL system by utilizing a collocation method that corresponds to a algebraic problem (with lower computational burden). However, if the ARL being utilized in the calibration process assumes the steady-state for enzyme

complexes, then it is necessary to compute the same expression that is computed in the KA method, becoming as computational intensive as the KA method. Any of the previous calibration methods that supports integration of the MARL can be utilized to calibrate the elementary rate parameters to the experimental data.

The global and partial global regression methods may also be utilized to take advantage of existing models (constructed with ARLs) and existing experimental data. However, the surface of the original ARL and the MARL system may diverge outside the concentration input space utilized in the calibration process. This also makes these calibration problems easier than the approximation of the full ARL and MARL model rate surfaces like the aforementioned strategies. It is also important to note that, the partial calibration process may require the simulation of the full model, if the initial system state, such as a steady-state concentration of enzyme complexes is not known *a priori*.

3.3.6 MARL Model and Identifiability Analysis

The MARL reaction descriptions were calibrated utilizing the previously described optimization methods in conjunction with one of the calibration procedures (describe in Section 3.2.1). Subsequently, the parameter identifiability was assessed by first, ranking the parameters based on the relative sensitivity metric described in the previous section in equation 3.18, followed by iterative construction of the experiment FIM (adding first more fragile parameters). Only the relative sensitivity matrices, corresponding to the available experimental points, were considered in this analysis. In this process, if the FIM matrix becomes singular after the addition of a certain parameter, the parameter is discarded (due to the fact of being correlated to an other parameter already present in the matrix). Afterwards, the calibration process is repeated with the non-identifiable parameters fixed to their first calibration value.

This approach is analogous to the one presented in DOE calibration procedure to compute the FIM. Still, reactions whose MARL can be described by a KA rate law, permit to constrain their more robust parameters, utilizing the equality of the ARL con-

stants (such as Kms, $Vmaxs$, Kis, $Keqs$) to the elementary rate parameter expressions derived by the Cleland method [18]. Setting up, these equality constraints targeting non-identifiable parameters tends to accelerate the calibration process and allows the utilization of extra information from the original ARL. However, if fragile parameters are chosen as targets of these constraints, the regression process may fail or be hampered. Non-identifiable parameters may not be able to alter the target output significantly, or possess the same output with distinct values. If these parameters are correlated, then there is the possibility of distinct parameters compensating their changes in value during the optimization (producing an infinite set of parameter combinations that produce the same output).

In Table 3.5, the reactions are characterized regarding their calibration and identifiability profile. Most of the reactions possess a limited number of identifiable parameters (in the sense of not being correlated with other reaction rate law parameters, often two or three parameters are non-correlated). It is also possible to observe that most of these non-correlated parameters cannot be estimated with good accuracy, due to the small amount of experimental data utilized in the calibration process. Due to the local nature of the employed identifiability procedure, it is also observable that reactions with different parametrizations may become more identifiable, like in the case of reaction Synth2. However, this phenomenon is independent of the calibration process. It all depends on how the optimization algorithm converges in the search space. Nonetheless, distinct calibration methods may create different search landscapes.

All calibration procedures tend to converge to solutions with the same level of Mean square average error (MSAE). Notwithstanding, most of reaction parameters may be fixed to their nominal values, due to their non identifiable profile in this analysis. Reactions with a limited number of parameters (up to 10) can be calibrated with any of the presented procedures. More complex reaction mechanisms (with higher number of parameters) may require less regression effort (in the sense of time to converge to a solution) if other methods besides KA are utilized.

KA based methods are limited in scope due to the increase in complexity in rate ex-

Table 3.5: MARL Reaction Identifiability Against Available Experimental Data

Reaction	Initial Calibration Method	#Parameters	#Estimable Parameters	MAE experimental data
ALDO	KA	8	3	0.06
ALDO	GS	8	3	0.06
DHAPS	GS	25	2	0.06
ENO	KA	6	2	0.06
ENO	GS	6	2	0.06
ENO	DOE	6	1	0.06
G1PAT	KA	46	Integration Error	0.06
G3PDH	GS	3	0	0.06
G6PDH	GS	25	3	0.06
GAPDH	GS	18	2	0.06
Murine Synth	KA	3	0	0.06
PDH	GS	3	0	0.06
PEPC	KA	23	0	0.06
PEPC	GS	23	3	0.06
PFK	GS	12	0	0.06
PDGH	GS	21	2	0.06
PGI	KA	16	3	0.06
PGI	GS	16	3	0.06
PGK	GS	18		0.06
PGluMu	KA	6	0	0.06
PGluMu	GS	6	0	0.06
PGM	KA	6	2	0.06
PGM	GS	6	2	0.06
PK	GS	7	1	0.06
PPC	GS	23	Singular FIM	0.06
PPK	GS	3	0	0.06
PTS	GS	40	4	0.06
R5PI	KA	6	0	0.06
R5PI	GS	6	0	0.06
RU5P	KA	6	0	0.06
RU5P	GS	6	0	0.06
Serine Synth	KA	3	0	0.06
Synth1	KA	3	2	0.06
Synth1	GS	3	2	0.06
Synth2	KA	3	3	0.06
Synth2	GS	3	0	0.06
TA	GS	10	2	0.06
TIS	KA	6	2	0.06
TIS	GS	6	2	0.06
TKA	KA	10	2	0.06
TKA	GS	10	2	0.06
TKB	KA	10	2	0.06
TKB	GS	10	2	0.06

pressions (concerning the number of terms), and the computational tractability of computing all the minimum spanning trees in the reaction graph representing a reaction (for more details about the method consult KA original article [17]). Another hurdle is the increase in the required sampling effort of the input space as the number of participating species in a reaction increases. A crude comparison of the complexity of the ARL generated by KA can be made by contrasting the difference in bytes between the ARLs. For instance, the KA of G1PAT reaction occupies 103 MBytes, while the PGK KA ARL has 26.8 KBytes.

Furthermore, all the calibration methods based on previously devised ARLs are bounded by their quality (concerning the identifiability of their parameters and the goodness of the fit to the available experimental data).

All the procedures consider a divide and conquer strategy, where each reaction is calibrated individually, before being plugged into the final model.

3.3.6.1 Model Assembly

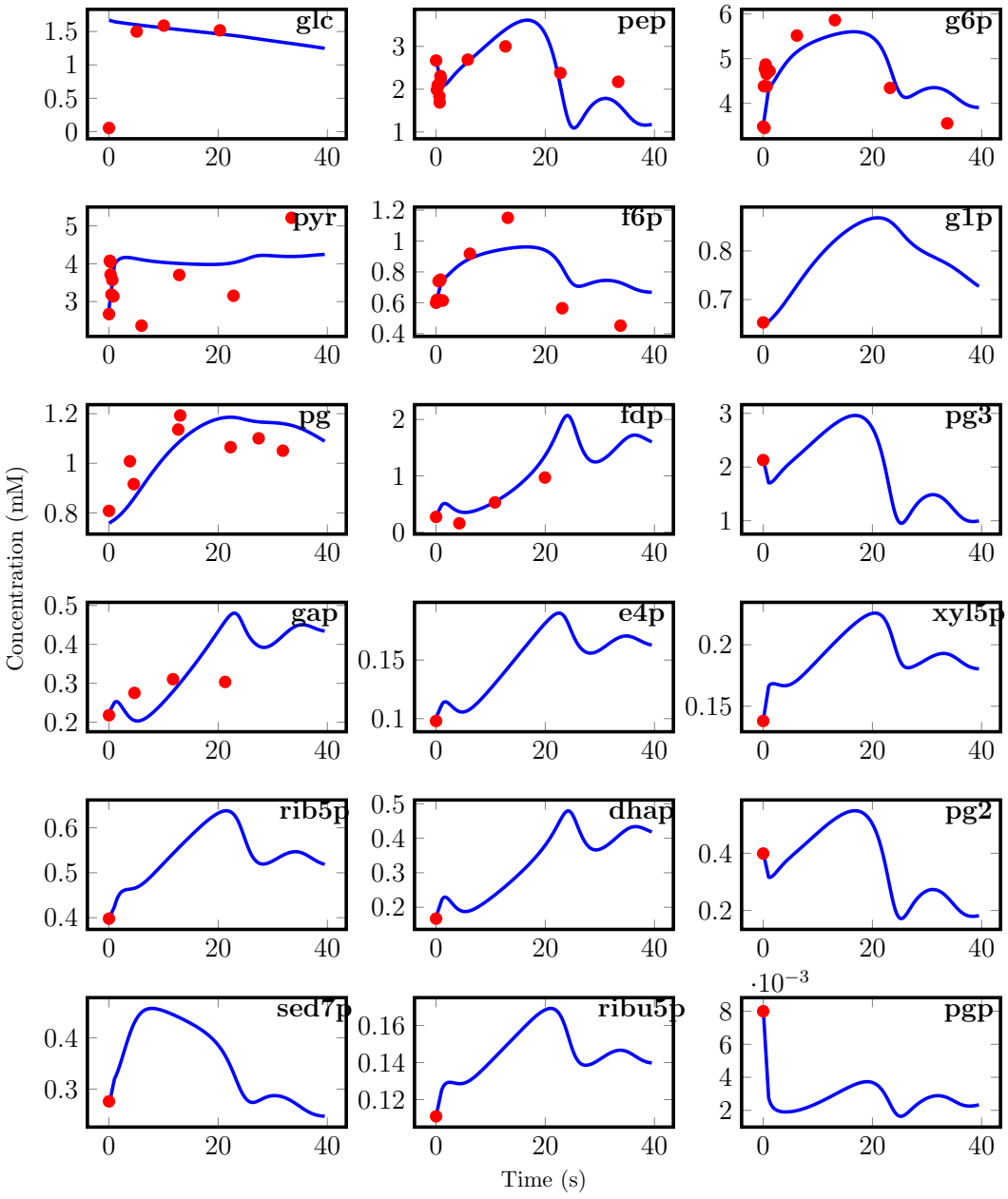


Figure 3.11: Experimental data versus MARL calibrated model (containing original PFK reaction).

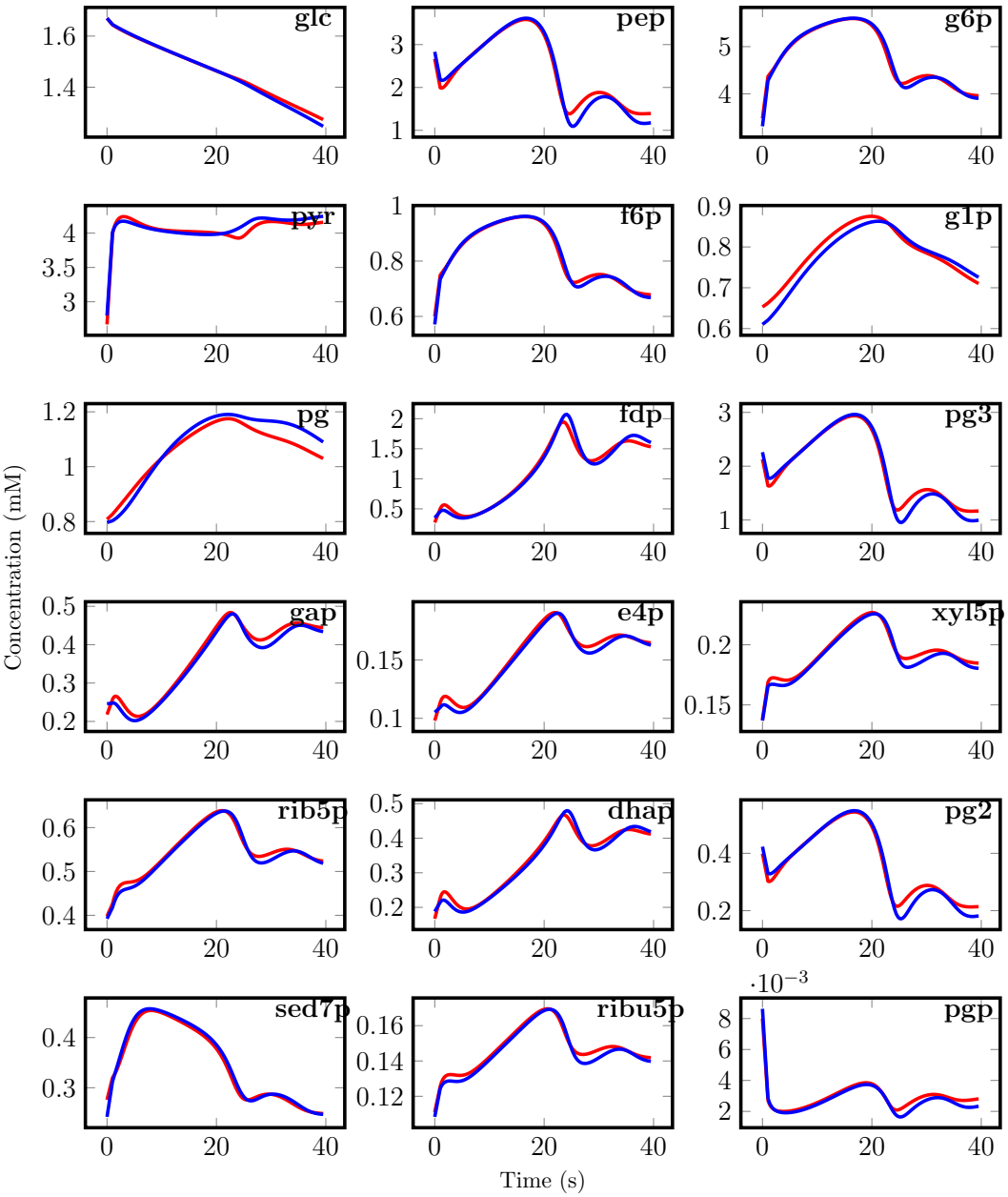


Figure 3.12: Chassagnole model versus MARL Calibrated model containing original PFK reaction. The blue line represents the MARL metabolite trajectories, while the red line represents the Chassagnole model metabolite profiles.

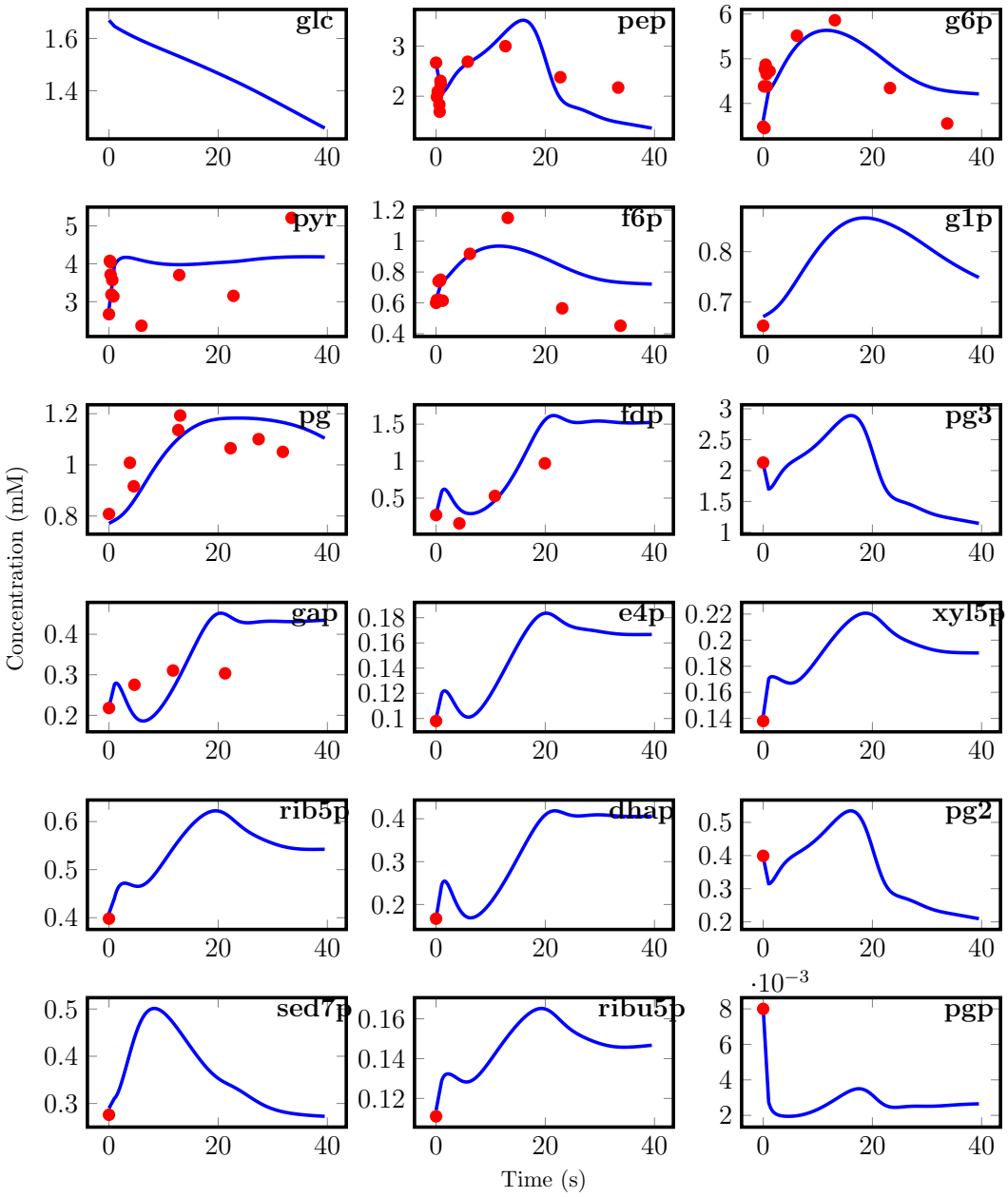


Figure 3.13: Experimental data versus MARL calibrated model.

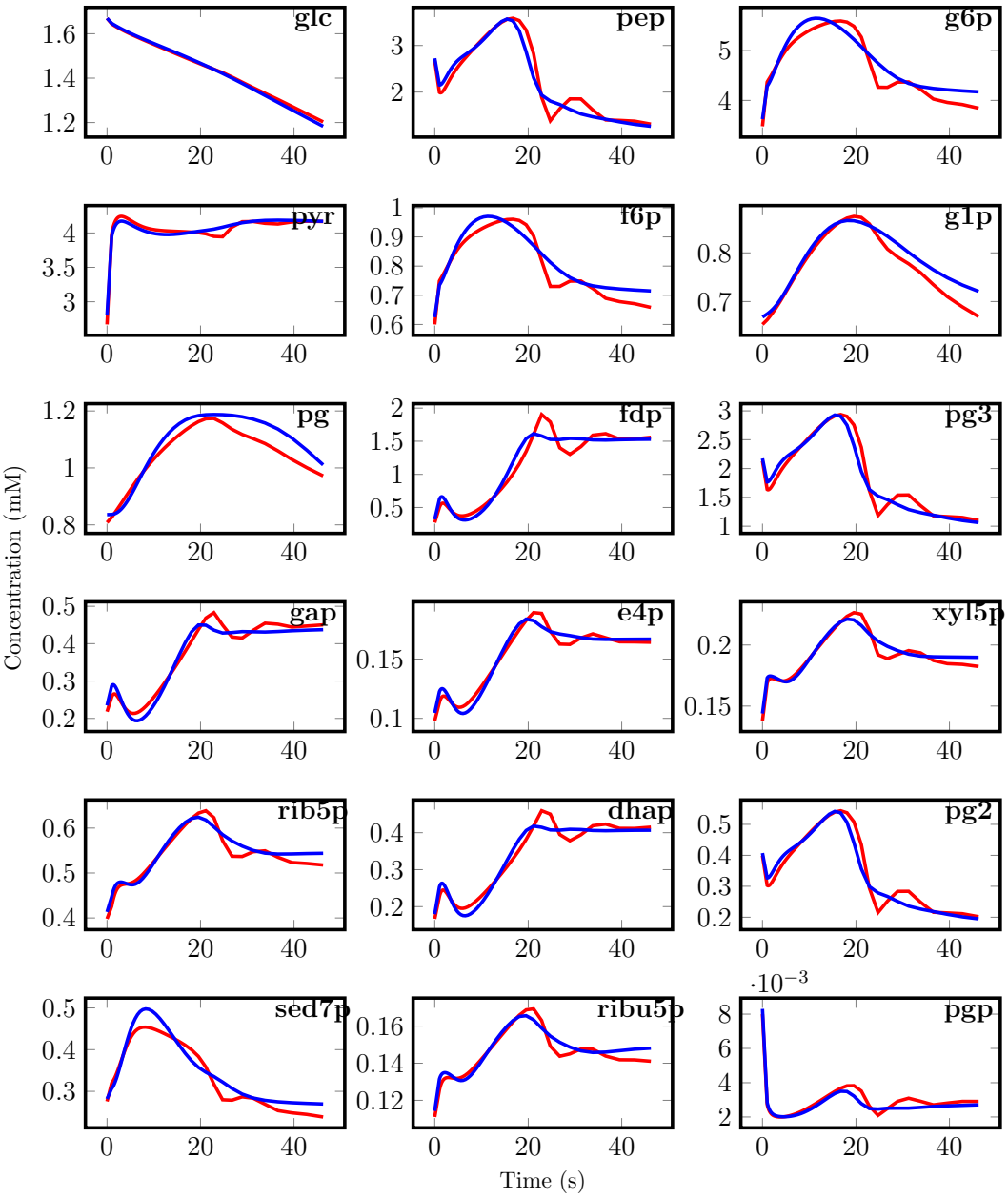


Figure 3.14: Chassagnole model versus MARL Calibrated model. The blue line represents the MARL metabolite trajectories, while the red line represents the Chassagnole model metabolite profiles.

It can be seen in Figures 3.11 and 3.12 that the MARL model containing the original PFK reaction and the Chassagnole model behave similarly in the pulse experiment and that both models possess a similar steady-state. The behavior of these models is constrained by the co-metabolites time dependent functional description. In other experimental contexts, these co-metabolites behavior would have to be acquired experimentally and fed to the model as time dependent functions. This limits the applicability of the model, despite having being utilized considering co-metabolites as constants.

When the updated version of PFK reaction devised by Zhao in [21] replaces the original PFK ARL in the previous MARL model, the system behaves as portrayed in Figures 3.13, and 3.14. The constructed models are in Appendix A.0.1.

3.4 Conclusions

In this work, the reactions of an existing model of central carbon metabolism of *Escherichia coli* were converted to the respective MARL ODE systems (by devising each interaction of the enzyme complexes and participating metabolites in the reaction). However, based on a sensitivity analysis of the original model (regarding individual parameters and reactions), it was decided to simplify some of the more complex full MA mechanisms, such as PFK, PK, among others.

The parameters of those systems were regressed to match the behavior of the corresponding ARL reaction equations of the original model, utilizing existing methods as well as new approaches devised in this work.

The MARL level of description may require less calibration data than other more data based, formalizations of kinetics, which incorporate little or no information about the underlying mechanism. Notwithstanding, these detailed descriptions possess a high number of parameters (are over-parametrized) and, therefore, the utilized optimization procedures should be coupled with identifiability analysis. Parameters whose impact is negligible or are correlated with other parameters should be fixed, to avoid failure of the optimization procedures (due to lack of convergence) or high variability in the

parameters (due to infinite sets of equally likely parameters).

All the devised calibration methods were connected with an identifiability analysis to pinpoint which parameters affected the most the regression process, and those that should be kept fixed. Existing strategies for estimating elementary rate values were also extended . The first strategy consisted in using the KA rate expression of the MARL system as proxy and regressing the parameters values in a least square setting against the reaction ARL equation.

The second strategy swapped the original ARL equation in the original model by the MARL ODE system and calibrated elementary rate equations using a Maximum Likehood estimation approach. This approach was followed in two distinct settings: (i) calibration against only available experimental data; (ii) calibration against pseudo experimental data generated by the original model. The simulated data mimicked the original pulse experiment, but captured all the metabolite profiles at equally spaced time intervals. In the first scenario, reaction rates often behaved differently (concerning the transient behavior) from the original model. In the second scenario, rates were able to present the same profiles as the original model, due to the larger constraints imposed by a higher density of experimental data.

The third approach is similar to the second one, but each reaction is represented by the CHA ARL of the mechanism. It is assumed that the the only slow steps of a reaction are the interconversion between enzyme complexes of substrate and products.

The fourth approach utilizes GE to define input functions to excite an enzyme mechanism, to provide more information about its parameters. Contrary to the other methods, this approach requires the fine tunning of experimental sampling times . In this process, integration of the MARL system may be avoided by utilizing a collocation method like defined in [23].

Simpler reaction mechanisms can be calibrated with any of these methods. However, more complex enzymatic mechanisms may be easier to calibrate with methods that require the computation of enzyme complexes expressions at steady state. These expressions can be very complex and possess a higher computational burden to compute and

evaluate (as shown on this chapter).

All these methods were coupled with an identifiability analysis, to fix robust parameters.

The Kronecker formalism served as low level representation of the system and was also extended to incorporate reactions with free form symbolic expressions. The constructed model can serve as basis to design new Metabolic Engineering strategies or to build a new set of experiments based on Optimal Design of Experiments to improve parameter accuracy [34]. This model captures a higher level of detail than the ARL model counterpart. Thus, it is expected to be valid in a larger physiological range. Nonetheless, this version of the model is coupled with the same time dependent functional descriptions of co-metabolite behavior as the original model. To utilize this model in other contexts the time profiles have to re-estimated and included into the model.

Abbreviations

ARL Aggregated rate law

ARL Mass action rate law

ALDO aldolase

DAHPS DAHP synthases

ENO Enolase

G1PAT Glucose-1-phosphate adenyltransferase

G3PDH Glycerol-3-phosphate dehydrogenase

G6PDH Glucose-6-phosphate dehydrogenase

GAPDH Glyceraldehyde-3-phosphate dehydrogenase

IleSynth Isoleucine synthesis

MetSynth Methionine synthesis

MurSynth Mureine synthesis

PFK Phosphofructokinase

PGDH 6-phosphogluconate dehydrogenase

PGI Glucose-6-phosphate isomerase

PGK Phosphoglycerate kinase

PGluMu Phosphoglycerate mutase

PDH Pyruvate dehydrogenase

PEPCxylase PEP carboxylase

PGM Phosphoglucomutase

PK Pyruvate kinase

PTS Phosphotransferase system

R5PI Ribose-phosphate isomerase

PPK Ribose-phosphate pyrophosphokinase

Ru5P Ribulose-phosphate epimerase

Synth1 Synthesis 1

Synth2 Synthesis 2

TA Transaldolase

TIS Triosephosphate isomerase

TKA Transketolase, reaction a

TKB Transketolase, reaction a

pg2 2-phosphoglycerate

pg3 3-phosphoglycerate

g6p Glucose-6-phosphate

rib5p Ribulose-5-phosphate

f6p Fructose-6-phosphate

g1p Glucose-1-phosphate

fdp Fructose-1,6-bisphosphate

dhap Dihydroxyacetone phosphate

pyr Pyruvate

pep Phosphoenolpyruvate

gap Glyceraldehyde-3-phosphate

mur Mureine

pg 6-phosphogluconate

sed7p Sedoheptulose-7-phosphate

e4p Erythrose-4-phosphate

xyl5p Xylulose-5-phosphate

pgp 1,3-diphosphosphoglycerate

ribu5p Ribose-5-phosphate

REFERENCES

[1] H. Kitano, "Systems biology: a brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002.

[2] X.-C. Zhang, A. Visala, A. Halme, and P. Linko, "Functional state modelling approach for bioprocesses: local models for aerobic yeast growth processes," *Journal of Process Control*, vol. 4, no. 3, pp. 127–134, 1994.

[3] A. Provost and G. Bastin, "Dynamic metabolic modelling under the balanced growth condition," *Journal of Process Control*, vol. 14, no. 7, pp. 717–728, 2004.

[4] K. J. Kauffman, P. Prakash, and J. S. Edwards, "Advances in flux balance analysis," *Current opinion in biotechnology*, vol. 14, no. 5, pp. 491–496, 2003.

[5] M. I. Klapa and G. Stephanopoulos, "Metabolic flux analysis," in *Bioreaction Engineering*, pp. 106–124, Springer, 2000.

[6] C. S. Henry, L. J. Broadbelt, and V. Hatzimanikatis, "Thermodynamics-based metabolic flux analysis," *Biophysical journal*, vol. 92, no. 5, pp. 1792–1805, 2007.

[7] M. Rizzi, M. Baltes, U. Theobald, and M. Reuss, "*In vivo* analysis of metabolic dynamics in saccharomyces cerevisiae: Ii. mathematical model," *Biotechnology and bioengineering*, vol. 55, no. 4, pp. 592–608, 1997.

[8] C. Chassagnole, N. Noisommit-Rizzi, J. Schmid, K. Mauch, and M. Reuss, "Dynamic modeling of the central carbon metabolism of *Escherichia coli*," *Biotechnology and Bioengineering*, vol. 79, no. 1, pp. 53–73, 2002.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[10] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[11] E. O. Voit, "Biochemical systems theory: a review," *International Scholarly Research Notices*, vol. 2013, 2013.

[12] D. Visser and J. J. Heijnen, "Dynamic simulation and metabolic re-design of a branched pathway using linlog kinetics," *Metabolic engineering*, vol. 5, no. 3, pp. 164–176, 2003.

[13] W. Liebermeister, J. Uhlendorf, and E. Klipp, "Modular rate laws for enzymatic reactions: thermodynamics, elasticities and implementation," *Bioinformatics*, vol. 26, no. 12, pp. 1528–1534, 2010.

[14] I. Schomburg, A. Chang, and D. Schomburg, "Brenda, enzyme data and metabolic information," *Nucleic acids research*, vol. 30, no. 1, pp. 47–49, 2002.

[15] U. Wittig, M. Golebiewski, R. Kania, O. Krebs, S. Mir, A. Weidemann, S. Anstein, J. Saric, and I. Rojas, "Sabio-rk: integration and curation of reaction kinetics data," in *Data integration in the life sciences*, pp. 94–103, Springer, 2006.

[16] J. F. Apgar, *Experiment design for systems biology*. PhD thesis, Massachusetts Institute of Technology, 2009.

[17] L. King, "The king-altman method for driving kinetic equations," *J Phys Chem*, vol. 60, pp. 1375–1378, 1956.

[18] W. Cleland, "The kinetics of enzyme-catalyzed reactions with two or more substrates or products: I. nomenclature and rate equations," *Biochimica et Biophysica Acta (BBA)-Specialized Section on Enzymological Subjects*, vol. 67, pp. 104–137, 1963.

[19] M. O'Neil and C. Ryan, "Grammatical evolution," in *Grammatical Evolution*, pp. 33–47, Springer, 2003.

[20] D. R. Hagen, *Parameter and topology uncertainty for optimal experimental design*. PhD thesis, Massachusetts Institute of Technology, 2014.

[21] J. Zhao, D. Ridgway, G. Broderick, A. Kovalenko, and M. Ellison, "Extraction of elementary rate constants from global network analysis of *E. coli* central metabolism," *BMC systems biology*, vol. 2, no. 1, p. 41, 2008.

[22] S. Cha, "A simple method for derivation of rate equations for enzyme-catalyzed reactions under the rapid equilibrium assumption or combined assumptions of equilibrium and steady state," *Journal of biological chemistry*, vol. 243, no. 4, pp. 820–825, 1968.

[23] K.-Y. Tsai and F.-S. Wang, "Evolutionary optimization with data collocation for reverse engineering of biological networks," *Bioinformatics*, vol. 21, no. 7, pp. 1180–1188, 2005.

[24] MATLAB, *version 7.10.0 (R2012b)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[25] G. Sin and K. V. Gernaey, "Improving the morris method for sensitivity analysis by scaling the elementary effects," *Computer Aided Chemical Engineering*, vol. 26, pp. 925–930, 2009.

[26] M. Ruano, J. Ribes, A. Seco, and J. Ferrer, "An improved sampling strategy based on trajectory design for application of the morris method to systems with many input factors," *Environmental Modelling & Software*, vol. 37, pp. 103–109, 2012.

[27] R. Albrecht, B. Buchberger, G. E. Collins, and R. Loos, *Computer algebra: symbolic and algebraic computation*, vol. 4. Springer Science & Business Media, 2013.

[28] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp, "Ecocyc: a comprehensive database resource for *Escherichia coli*," *Nucleic acids research*, vol. 33, no. suppl 1, pp. D334–D337, 2005.

[29] D. Degenring, C. Froemel, G. Dikta, and R. Takors, "Sensitivity analysis for the reduction of complex metabolism models," *Journal of Process Control*, vol. 14, no. 7, pp. 729–745, 2004.

[30] K. Z. Yao, B. M. Shaw, B. Kou, K. B. McAuley, and D. Bacon, "Modeling ethylene/butene copolymerization with multi-site catalysts: parameter estimability and experimental design," *Polymer Reaction Engineering*, vol. 11, no. 3, pp. 563–588, 2003.

[31] T. Sumner, *Sensitivity analysis in systems biology modelling and its application to a multi-scale model of blood glucose homeostasis*. PhD thesis, UCL (University College London), 2010.

[32] K. Smallbone, E. Simeonidis, D. S. Broomhead, and D. B. Kell, "Something from nothing- bridging the gap between constraint-based and kinetic modelling," *Febs Journal*, vol. 274, no. 21, pp. 5576–5585, 2007.

[33] K. Smallbone and P. Mendes, "Large-scale metabolic models: From reconstruction to differential equations," *Industrial Biotechnology*, vol. 9, no. 4, pp. 179–184, 2013.

[34] J. F. Apgar, J. E. Toettcher, D. Endy, F. M. White, and B. Tidor, "Stimulus design for model selection and validation in cell signaling," *PLoS Comput Biol*, vol. 4, no. 2, p. e30, 2008.

CHAPTER 4

EXTENSION OF DYNAMIC FLUX BALANCE ANALYSIS

Increases in computational power in conjunction with the evolution of high-throughput sequencing technologies lead to an explosion of biological data, that accompany the creation of computational methods for the semi-automatic curation of this information. As consequence the number of genome scale models of metabolism published in the literature has been steadily increasing. Several methods allow to predict a possible homoeostatic cell state (under certain assumptions) utilizing the metabolic network topology as input, as well as, the mass exchanges with the environment. These abstraction are often employed as part of optimization problems to find suitable strategies to modify or study organisms behavior under different experimental conditions. Bioprocess models (for a well mixed processes) on the other hand are often described by a set of differential equations, where the cells are depicted as a black-box entities responsible for transforming specific substrates into products. This simplified representation hampers the capability of representing all the cellular states due to the simplified representation.

Dynamic Flux Balance Analysis (DFBA) was designed to tackle this issue allowing to couple these two distinct formalisms. The link is made by coupling the cellular exchange fluxes (often updated at regular time intervals) with the bioprocess differential equations. However, DFBA only permits the simulation of batch and fed-batch systems when substrate is present in the bioreactor.

In the past, these models had already been linked to bioprocess systems by replacing the black-box representation of cells in the bioprocess model by the uptake/excretion reactions related to the target microorganism genome scale model. However, these models possess some shortcomings such as the incapacity to model the absence of substrates in the bioreactor.

In this work, an extension to DFBA allowing the simulation of the system in the

absence of substrates in the bio-reactor is elaborated. The method is applied in the construction of a bacth/fed-batch model of *Escherichia coli*, as well as, in the optimization of a feeding strategy. The system also permits the simultaneous arrangement of Metabolic Engineering strategies.

As a case study, *Escherichia coli* core stoichiometric model was utilized in conjunction with a standard bioprocess model describing a fed-batch process.

The model was calibrated with *in vivo* fermentation data, after a global sensitivity analysis to fix unimportant or non-influential parameters was carried out. Afterwards, the delineated model served as basis for the design of near optimal flux feed functions under the goal of maximizing the production of the target product at the end of the batch, while minimizing the consumption of substrate.

4.1 Introduction

In the last decades, one of the major trends of the biochemical industry has been the replacement of chemical synthesis methods by biotechnological ones. In these processes, the capacity to model the interactions of the cell with its environment plays a crucial role. Several methods have been described in the literature to address this problem. One of the most common methodologies is the description of the model by a system of ordinary differential equations (ODEs). Most of these models utilize a simplified representation of a cell (as simple black-box converter of substrates into products) without accounting for its inner workings [1].

More detailed cellular models require a more thorough description of the cellular processes and data, depending on the level of abstraction adopted.

Recently, due to the increasing availability of omics data, the number of genome scale models available in the literature has been increasing. Those models are composed by a topological and stoichiometric description of the metabolic network of a specific microorganism. Many mathematical methods take advantage of these models by assuming that the cell is in an equilibrium state. Flux Balance Analysis (FBA) casts the problem of finding one of those equilibrium states as a linear programming optimization problem, where the cell (or the system) has a specific goal (often the maximization of the biomass growth) and the reaction fluxes are within limited bounds [2].

However, these methods may not return a unique steady-state (and the set of valid steady-states satisfying optimality may be infinite). Another hurdle is the definition of the objective functions of these methods and their match to the state of the cell. Nonetheless, these methods have been used successfully to predict lethal phenotypes and cellular states in a variety of conditions including in reaction knock-out studies [3].

The conjunction of FBA with the ODE description of a bioprocess has led to the creation of Dynamic Flux Balance Analysis (DFBA) [4]. This method allows the simulation of the bioprocess system assuming the cell is always in steady-state and may change its state after a certain amount of time (often a discrete number of time inter-

vals). The FBA method is often executed in the DFBA process with the biomass growth as the objective function. Contrarily to the bioprocess ODE models, that utilize black-box descriptions of the cell, these DFBA models may be able to portray a vast number of cellular states without the need of re-fitting or even updating the model structure to represent new bioprocess states.

Understanding how cells interact with their environment and how extra cellular variations affect their state has industrial interest in conceiving strategies to simulate and optimize biological processes. Microorganisms try to keep homoeostasis and react to external environmental modifications through a web of components encompassing metabolic, genetic and enzymatic networks. The prowess to model this behavior integrated with a formal bioprocess abstraction can lead to the development of new feeding profiles in a fed-batch context, as well as the design of targeted Metabolic Engineering (ME) strategies during transient stages of the culture.

The initial DFBA formulation [4] assumed the discretization of the simulation at fixed length time intervals. Each time interval was simulated by first computing the model uptake rates based on the available compounds in the bioreactor. After fixing the uptake fluxes, FBA was utilized to maximize the cell growth rate and to predict the biochemical flux distribution. Uptake and excretion fluxes were then set in the ODE bioprocess system. Finally, the ODE system would be simulated up to the end of the defined time interval. The process would be repeated until the end of the simulation or would stop if the FBA method was infeasible. This formulation assumes a linear model and does not require any previous calibration.

In [5], the authors presented a DFBA approach called static optimization approach, with the same premises as [4], but with the possibility of utilizing non-linear rate equations for the uptake reactions.

One important limitation of DFBA is the fact that existing formulations cannot be directly applied to fed-batch fermentations. Although a few examples of this adaptation have been described ([6]), those are not applicable in situations where substrate reaches concentrations close to zero (which is often the case).

The DFBA approach presented in [5] requires the calibration of the utilized rate equation parameters. One important part of this process should incorporate identifiability analysis, to fix non-influential or non-identifiable parameters to default values. Identifiability analysis often utilizes techniques from two intertwined distinct areas: (i) Uncertainty analysis (UA) methods aim to quantify these parameters uncertainties; (ii) Sensitivity Analysis (SA) to characterize the source of parameters' volatility [7] [8];

Some of these model parameters in their defined ranges do not influence the model outputs or are non-identifiable posing problems to optimization methods when calibrating the model. Thus, it is important to apply techniques to identify and those that drive the calibration process [7].

Sensitivity methods can broadly be divided into three categories: Local methods are based in the derivatives of parameters regarding the target output variable. These values are local in nature due to the fact of being related to specific system states and contemplate only a change in a parameter at the time. Global methods, on the other hand, take into consideration the simultaneous variation of a set of model parameters, reducing nevertheless the applicability, of some techniques [8] [7] (check Chapter 2).

Contrarily to ODE systems, DFBA based systems do not allow the computation of analytical parameter derivatives regarding compounds along time. Even if these derivatives are computed numerically, they should be used judiciously due to the possibility of discontinuities in fluxes caused by the flux determination method utilized in between time intervals (such as FBA).

Thus, one approach to infer the parameter sensitivities is to utilize Global Sensitivity Analysis based methods, that utilize data from the sampling of the parameter space (considering the simultaneous variation of all of the model parameters and not being constrained by a nominal parameter set as in the previous methods), and often do not require the use of derivatives. These techniques tend to be computationally more intensive; however, they offer a global overview of the parameter sensitivity, possibly accounting for the non-linear interactions between parameters. In the scope of this work, methods such as High Dimensional Model Representation (HDMR) [9], Fourier amplitude sen-

sitivity test (FAST) [10], or the Morris Method [11] are of extreme importance (check Chapter 2 for a more detailed explanation).

HDMR is a black-box functional mapping of the input-output behavior of a system (for more information see Chapter 2) requiring low sampling effort (with polynomial complexity regarding the number of variables considered) [12]. This permits to construct Input-Ouput representations often called fully equivalent representations of a system - concerning the input space where they were calibrated. This abstraction allows to decompose the interaction and the main effects of each input on the output variables, thus giving global quantitative sensitivity measures.

The Morris Method, on the other hand, is a GSA method that allows to estimate a facade value for the global sensitivity of a model parameter. These values permit to rank parameters and to identify which parameters are non-influential for the observed behavior. Contrarily to HDMR based techniques this method does not allow to infer the interaction structure of the parameters.

This work encompasses the creation of new method based on DFBA with the prowess of simulating a bioprocess system in fed-batch conditions (even in the absence of substrate in the medium). The model is composed by a ODE based description of the bioprocess connected to a stoichiometric model. This method was applied to an *Escherichia coli* bioprocess. The ODE model parameters were regressed against available experimental data and their uncertainty was assessed by utilizing GSA methods before and after the calibration process. These analyses served to identify non-important parameters (whose value may be fixed to a base value without affecting the model output) and the interactions between parameters and variables that influence the observed variables trajectories. This model also served as basis for the delineation of (near) optimal feeding strategies.

4.2 Methodology

In this work, a novel extension of DFBA is proposed allowing the simulation of the system (the bioprocess and the participating microbial strains). As a case study, a bioprocess model of *Escherichia coli* regarding batch and fed-batch fermentations was built and linked to a stoichiometric core model [13] through exchange fluxes.

The uptake fluxes present in the bioprocess model were represented by kinetic equations portraying known information about the system. For instance, the glucose uptake rate was shaped taking into consideration possible inhibition caused by ethanol.

The influence of model parameters on the calibration process was assessed by utilizing the Morris method prior to the model calibration and afterwards. However, viable ranges for the parameters (rate equation parameters and initial concentrations of the participating species) were defined before this process began. The first application of the Morris method application served to pin-point non-influential parameters with large uncertainty. Next, the model was regressed utilizing the available experimental data, followed by a second application of the Morris method with stricter parameter bounds.

After the calibration process and the Morris method application, the identifiability of the parameters regarding these rate equations was studied at the system's level. The parameter spaces were sampled using Latin Hyper Cube Sampling Scheme and the respective system simulations were saved. The impact of each parameter in a simulation was assessed utilizing functional Pricinpal Component Analysis (PCA) weight scores as described in [14] (see Chapter 2). These scores, alongside with the simulated concentrations served as outputs and inputs, respectively, for the construction of a HDMR model. The HDMR model provides a direct way of computing the sensitivity indexes and, consequently, the total sensitivity indexes. The last metric allows to pin-point if a parameter may be deemed non-influential - a parameter is non-influential if distinct values of a parameter within a range produce the same output. Subsequently, parameters deemed non-influential by the respective total sensitivity index were fixed to their original value.

The delineated model was applied in a (near) optimal feeding design problem, whose goal is to maximize the production of a target compound along side with biomass production, while minimizing the consumption of substrate during the fed-batch fermentation. Two distinct algorithms were utilized for defining the feeding profile: (i) Differential Evolution Based: where the main goal of the algorithm is to find the feeding value at a set of interpolation points supplied by the user; (ii) Grammatical Evolution Based: whose main goal is to encounter a symbolic expression for the overall feeding profile. Due to the versatility of the method, this approach also permits to search simultaneously for ME strategies, although it is important to bear in mind that this increases the complexity of the problem.

The case study utilized for the validation of the feeding design method was the maximization of the microorganism's biomass.

4.2.1 Model Description

The bioprocess model was developed assuming an homogeneous bioreactor system with glucose, acetate, ethanol, formate, lactate, oxygen and biomass, having as basis the following ODE system:

$$\begin{cases} \dot{X} = \mu X - \frac{f_{in}}{V} X \\ \dot{Glc} = Flux_{glc} X - \frac{f_{in}}{V} Glc + inflow * X \\ \dot{Ac} = Flux_{ac} X - \frac{f_{in}}{V} Ac \\ \dot{Eth} = Flux_{eth} X - \frac{f_{in}}{V} Eth \\ \dot{Form} = Flux_{form} X - \frac{f_{in}}{V} Form \\ \dot{Lac} = Flux_{lac} X - \frac{f_{in}}{V} Lac \\ \dot{V} = f_{in} \\ \dot{O}_2 = 0 \end{cases} \tag{4.1}$$

where X is the biomass concentration, Glc is glucose, Ac is acetate, Eth is ethanol, Form

is formate, O_2 is oxygen, concentrations; V is the bioreactor volume, μ (flux associated to the stoichiometric model) is the biomass specific growth rate and terms starting with the string $Flux$ are fluxes connected to the stoichiometric model, f_{in} is the volumetric flow rate entering the bioreactor. This model represents an aerobic process where typical anaerobic fermentation products are allowed to accumulate assuming that limitations in oxygen transfer may cause periods of time of anaerobiosis.

The uptake fluxes, namely $Flux_{glc}$, $Flux_{ac}$, $Flux_{O_2}$ are applied as upper bounds in the respective uptake flux constraints and set based on the following expressions:

- $Flux_{glc} = qs$ (its computation is explained below);
- $Flux_{ac} = \frac{qacmaxAc}{Ac+acks}$, where $qacmax$ is the maximum uptake rate and $acks$ is the half saturation constant. The kinetic expression was chosen assuming that acetate maximum consumption flux varies following a Michaelis-Menten like curve;
- $Flux_{O2} = qmax_{o_2}$, where $qmax_{o_2}$ is the maximum oxygen uptake rate by the cell. This expression is not directly utilized in the ODE model, but just as a constraint in the stoichiometric model. Assuming that the bioprocess oxygen concentration does not fluctuate, the oxygen maximum consumption flux was set to a constant value;

Each expression in the ODE system can be described generically by:

$$\dot{C}_i = \sum_{j=1}^{nJ} V_j(x,\theta)X - \sum_{w=1}^{nW} V_w(x,\theta)X - \frac{\sum_{z=1}^{nZ} Fin_z}{V} C_i + \sum_{y=1}^{nY} pumpFunction_y(x,\theta) \quad (4.2)$$

where C_i represents the concentration of variable i, X is the biomass variable, $V_j(x,\theta)$ is a rate expression concerning the excretion fluxes (taking as input the system state x and the respective parameters θ), analogously $V_w(x,\theta)$ is a rate equation contemplating the uptake fluxes, nJ is the number of excretion rate expressions (that lead to the creation of C_i), analogously nW is the number of uptake rate expressions (that lead to the consumption of C_i), $\frac{\sum_{z=1}^{nZ} Fin_z}{V}$ represents the dilution term ($\sum_{z=1}^{nZ} Fin_z$ is the sum of all

the pumps input fluxes - this sum is equal to zero if the bioreactor volume is exceeded) and $pumpFunction_y(x, \theta)$ represents the amount of C_i added to the medium by pump y, while nY is the total number of pumps present in the system.

4.2.1.1 Uptake Rate Equation Expression Connected to a Feeding Pump

Considering the feeding of the compound i into the bioreactor by a feeding pump, the respective uptake rate equation $qs(x, \theta)$ will have to be modified. Firstly, expressions utilized to construct $qs(x, \theta)$ will be presented - as an example the effect of modifier compounds in the rate expression will also be shown assuming monod like kinetics. Other type of kinetic equations may also be employed with the proper modifications.

f_{in} represents the volumetric flux rate of the feeding pump regarding the compound C_i. If the current volume of the bioreactor exceeds its capacity, the feeding flux is set to zero and, consequently, the volumetric flux rate is also zero.

$$f_{in} = \begin{cases} f, & \text{if } V \leq V_{upperLimit}. \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

Assuming the uptake of compound C_i is inhibited by compound I, the inhibition term can be described by:

$$qsITerm = \frac{1}{1 + \frac{I}{qski}} \tag{4.4}$$

The qsC_iOnly expression gives the consumption rate of compound C_i, considering only the monod kinetics.

$$qsC_iOnly = \frac{qsmaxC_i}{qsks + C_i} \tag{4.5}$$

The $inflow$ accounts for the flux being fed into the bioreactor. s_{in} represents the concentration of C_i in the flux feed.

$$inflow = \frac{fin}{V}\frac{1}{X}s_{in} \tag{4.6}$$

where $inflowQs$ is the sum of the $inflow$ with $qsGlcOnly$, representing the maximum flux value that is available for the cells, corresponding to the amount of glucose entering the reactor plus what is already there.

$$inflowQs = inflow + qsC_iOnly \tag{4.7}$$

If C_i is not present in $inflowQs$ the bioreactor (its concentration is zero), then $qs(x, \theta)$ uptake rate equation is given by:

$$qs = \begin{cases} -1 \times inflow \times qsITerm, & \text{if } qsmax \geq inflow. \\ -1 \times qsmaxC_i \times qsITerm, & \text{otherwise.} \end{cases} \tag{4.8}$$

In the scenario in which C_i is available in the medium, $qs(x, \theta)$ is given by:

$$qs = \begin{cases} -1 \times (inflow \times qsITerm + qsC_iOnly \times qsITerm), & \text{if } qsmax \geq inflowQs. \\ -1 \times qsmaxC_i \times qsITerm, & \text{otherwise.} \end{cases} \tag{4.9}$$

In this model, it is assumed that the microorganism(s) consume the substrates present in the bioreactor, and simultaneously consume what is being provided instantaneously at that time by the feed pump until a given limit (qsmax) given by the maximum uptake of the cells. It is also assumed, that if there is enough substrate quantity to support the maximum C_i consumption rate, the microorganism will try to consume it, affected by the respective modifiers (activators or inhibitors of the phenomena). The system would have to be modified accordingly to support new activation or inhibition effects from other effectors.

4.2.1.2 Stoichiometric Model and Data

In this work, we utilize *Escherichia coli* core stoichiometric model [13]. The model encompasses glycolysis, penthose phosphate pathway, tricarboxylic acid cycle, glycoxylate cycle, gluconeogenesis, oxidative phosphorilation and transfer of reducing equivalents, nitrogen metabolism, and anaplerotic reactions. This model was chosen, due to its simplicity (thus, it has a lower computational burden when simulating, when contrasted with genome scale stoichiometric models), and the capability to represent the central carbon metabolism, that is linked with the bioprocess model, by the exchange fluxes. This model is also employed, instead of more complex models, due to the necessity of only estimating the metabolic fluxes in the central carbon metabolism.

Model Initial Conditions and Parameter Values

The model initial conditions are defined based on the available experimental conditions (consult section 4.2.1.2). In the feeding design problems, all the compound concentrations in the medium are set to zero (except for oxygen), while the remaining variables mimic the values of the batch experiment utilized in the calibration process.

The nominal parameter set was defined after an initial calibration of the model. Originally, all the parameters in the model, were allowed to vary between -3 and 3 in log-scale. These wide ranges were chosen to depict the uncertainty of these parameter values.

Experimental Data

Data from a 20 hours batch fermentation of wild-type *Escherichia coli* were utilized in the calibration process. The data are from [15] and were ceded by the authors. These data describes a bioprocess in aerobiosis, where the following compounds were measured: glucose, acetate and biomass (through optical density). The initial conditions of the system on these experimental data were:

- glucose : 31 mMol

- biomass : 0.1 gDw/l
- acetate : 0 mMol
- formate : 0 mMol (assumed)
- ethanol : 0 mMol (assumed)

These conditions are applied in the calibration process.

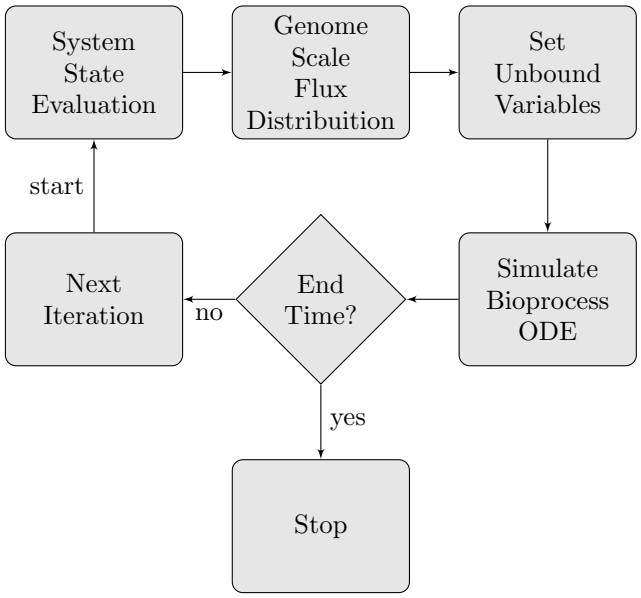
4.2.1.3 Model Simulation

The DFBA model is simulated in a similar way to the static optimization algorithm (SOA) [5] with the introduction of an extra step in the flux value computation, allowing to describe new scenarios concerning the consumption patterns of microorganisms. Assuming that the fermentation process has been described by an ODE system, whose variables are also linked to the exchange fluxes of at least one genome scale model, the system is simulated in the following way:

1. The simulation time is divided into equally spaced time steps. The length of these steps is defined by the user and is problem dependent. It is assumed in the beginning of the time step that the cell arrives at a new steady-state;
2. At each time step, firstly the uptake rate equation values are computed taking into consideration any conditional behavior or rules and the state of the system (concentrations and parameter values). These rate equation values are then fixed in the stoichiometric model.
3. The stoichiometric model is utilized in a method to predict a flux distribution. In this work we utilize Parsimonious Flux Balance Analysis (PFBA) [16]. Another method that produces flux distributions at steady-state could be employed instead;
4. The values of the unbound rates (not computed previously in step 2) are set based in the previous computation by PFBA;

5. The model is simulated utilizing a suitable ODE solver for the length of the time step. The fluxes computed in step 3 are kept constant during the execution of the time step;

Figure 4.1: Flux diagram, representing the DFBA extension simulation process.



The process is repeated until a termination criterion is met (often the end time of the simulation) as illustrated in Figure 4.1.

4.2.2 Identifiability Analysis

The identifiability analysis performed here is divided into distinct stages: (i) identification of non-influential parameters in the original model, by utilizing the Morris method with scaled elementary effects [17]; (ii) description of the interaction structure of the parameters based on the available experimental data to calibrate the model, as well model simulations, utilizing a Quasi Random Sampling HDMR (QRS-HDMR) [18].

4.2.2.1 Calibration Parameter Global Sensitivity - Morris Method With Scaled Elementary Effects

In this work, the global sensitivity indexes (the effect of a parameter on an output, including interaction effects with other parameters), were assessed by using the Morris method. The Morris method allows to identify and rank which parameters of a function y, defined in a unit hypercube divided into p equal parts (called levels) may affect the output of a system by computing a set of measures (called elementary effects (EE)), and their statistics utilizing one-step-at-a-time (OAT) sampling scheme, repeated r times.

In the context of this work, the Morris Method is utilized to infer the impact of each model parameter (without contemplating the model initial conditions explicitly), in the objective function $J(\theta)$ defined for the calibration process. The scaled elementary effects utilized in this work are defined as:

$$ScaledEE_i(\theta) = \left| \frac{J(\theta_1, ..., \theta_{i-1}, \theta_i + \delta, ..., \theta_k) - J(\theta)}{\delta} \right| \frac{\sigma\theta_i}{\sigma_J} \tag{4.10}$$

where $J(\theta)$ is the calibration objective function with input parameter set θ, $J(\theta_1, ..., \theta_{i-1}, \theta_i + \delta, ..., \theta_k)$ designates the output value, where parameter θ_i was perturbed by δ (perturbation constant), $\sigma\theta_i$ is the standard deviation of parameter θ_i, σ_J is the standard deviation of the calibration function over the whole range of parameters. For each parameter r, $scaledEEs$ are estimated (for information on the parameter ranking procedure refer to Chapter 2).

The Morris method with scaled elementary effects was applied to the depicted DFBA system, before the calibration procedure in a batch fermentation for a period of 20 hours, where the microorganism internal flux state was updated at every 0.125 hours. The method was executed with r equal to 140 and p (number of levels) equal to 10. This value was selected based on the method developed by [19] to identify the optimal r value. All the parameters were allowed to vary within the interval -3 to 3 in log-scale (in their respective units). This range was selected to include a wide range of values, some outside of biological feasibility, due to the fact of not utilizing literature based val-

ues as starting parameter values. After the calibration process, the Morris method was reapplied to the system, allowing the system parameters to vary 50% regarding the best parameter set found. It is important to bear in mind that this method was employed to identify which parameters drive the calibration process and their relative influence.

4.2.2.2 Time Series Global Sensitivities

There are several techniques to assess the sensitivity measures of time series trajectories, such as the area under the curve of relative sensitivities, derivatives or sensitivity indexes. However, most of these methods are based on the computation of the average value of a discrete set of sensitivity measures computed at specific time intervals. These procedures correspond to weighting each sensitivity index equally, what may distort the overall sensitivity index.

In this work we utilize the approach developed in [14] that weights each sensitivity index according to their importance, utilizing equidistant time points (the ones utilized in the extended DFBA formulation).

The sensitivity of each model parameter concerning each output variable was assessed by using functional Principal Component Analysis (fPCA) scores in conjunction with Quasi Random Sampling High Dimensional Model Reduction (QRS-HDMR). Both techniques are detailed in Chapter 2. In this work, these methods were utilized to identify which parameters influence the most the aforementioned DFBA time series trajectories (e.g biomass, glucose and all the remaining system variables), after the calibration process.

The sensitivity indexes were computed assuming that each parameter may change its value one order of magnitude, concerning the best parameter set found in the initial calibration process. The parameter space was sampled utilizing Latin Hyper Cube Sampling (LHCS), obtaining as outputs the variable trajectories at equidistant time points from the simulation process. Next, the set of simulations corresponding to each variable were approximated through the computation of the functional principal components and the respective scores. The principal components describe the modes of variation (vari-

ability) in the data in decreasing order. Thus, only a subset is needed to represent the data (and its variability), by a user defined threshold, thus reducing data dimensionality. As explained in [14] (using the same notation), a model output y_i, described by functional principal components $\xi(t) = (\xi_1(t), \xi_2(t), \ldots, \xi_q(t))$ can be defined as:

$$y_i(t) = \sum_{j}^{nPCs} \omega_{i,j}\xi_j(t) \tag{4.11}$$

where $nPCs$ is the number of functional principal components considered by the user, the output $y_i(t)$, represents the sum of each of the functional principal components $\xi_j(t)$, times the principal component score $\omega_{i,j}$, for model output i on component j. The score indicates the amount of principal component j contained in the output i (for details on the computation of the functional principal components consult [14]). These scores replace the output simulations, as the output of the system (the output becomes a vector of scores, one for each PC). Thus, global sensitivity methods are utilized to identify which parameters affect the selected functional principal component scores. It is important to note that each PC score has a sensitivity measure associated. These values are aggregated to generate a single sensitivity index, given by:

$$S_i^{Overall} = \sum_{n=1}^{nPCs} S_i^n V_{PC}^n \tag{4.12}$$

where $S_i^{Overall}$ is the weighted sensitivity index of parameter i, $nPCs$ is the number of selected functional principal components, S_i^n is the sensitivity index value (or fPCA score sensitivity) for functional principal component n, and V_{PC}^n is the part of the variance explained by the fPC n.

If a parameter is considered influential, then it has an effect on the score of at least one of the fPCs, thus on the observed behaviour of the output. Functional principal components were utilized in conjunction with RS-HDMR to compute the sensitivity indexes of the model parameters for each output. The HDMR computations include the extensions described in Chapter 2. The HDMR model for a fPC z score, concerning one model output is given by:

$$f(\theta)^z \approx f_0^z + \sum_{i=1}^{n} f_i^z(\theta_i) + \sum_{1 \le i \le j \le n} f_{ij}^z(\theta_i, \theta_j) \tag{4.13}$$

where θ are a set of selected model parameters under study affecting the DFBA model output i, $f(\theta)^z$ is the score value for fPC z (for one model variable), f_0 is the mean value of the scores for fPC z under all $f(\theta)$ and each successive order function represents the parameter set contribution θ to $f^z(\theta)$. For instance, f_{ij}^z contains the contribution of θ_i and θ_j for the output score of fPC z in $f^z(\theta)$. These $f(\theta)^z$ are computed for all the utilized fPCs. For each fPC the sensitivity indexes are computed (as explained in chapter 2), and their overall sensitivities values aggregated as explained in equation 4.12.

Model Regression

The goal of the regression procedure is to find a set of parameter values that make the model reproduce the experimental data. The goodness of this reproduction is assessed by a cost function $J(\theta)$. In this work, we utilize a weighted least squares, whose weight corresponds to the variance of the measurement, at a specific time. This is equivalent to maximum likelihood estimation, when it is assumed that errors are normally distributed. The weighted least squares is given by:

$$J(\theta) = \sum_{t=1}^{t} \sum_{i=1}^{n} \left(\frac{(C(t)_i - C(t)_{i,exp})^2}{\sigma_{C_{i,exp}^2}} \right) \tag{4.14}$$

where $J(\theta)$ is the cost function, $C(t)_i$ is the simulation value of variable C_i at time t, $C(t)_{i,exp}$ is the experimental measurement of variable $C_{i,exp}$. After, fixing the parameters classified as non-influential by the Morris method, the calibration process was carried utilizing an Evolutionary Computation approach. The previously presented model parameters were regressed using the Differential Evolution algorithm and the minimization of the aforementioned function $J(\theta)$.

4.2.3 Optimization Algorithms

In this work, two distinct types of evolutionary algorithms are employed: (i) Differential Evolution (DE) [20]; (ii) Grammatical Evolution (GE) [21].

4.2.3.1 Differential Evolution Configuration

The calibration process employed a variant of the DE algorithm called *DE/rand/1* that uses a binomial crossover [20]. In this case, the following scheme is followed, in every generation, for each individual i in the population:

1. Randomly select 3 individuals r_1, r_2, r_3 distinct from i;
2. Generate a trial vector based on: $\vec{t} = \vec{r}_1 + F \cdot (\vec{r}_2 - \vec{r}_3)$, where F is a weighting factor;
3. Incorporate coordinates of this vector with probability CR;
4. Evaluate the candidate and use it in the new generation, if it is at least as good as the current individual.

The DE was ran for 500 iterations with a population of 20 individuals. The F parameter was set to 0.5 and CR to 0.6.

In the context of this work, a solution represents a parametrization of a subset of model parameters defined by the user. A solution is evaluated by computing the objective function described previously in the calibration subsection.

4.2.3.2 Grammatical Evolution Configuration

The GE [21] algorithm behaves similarly to a Genetic Algorithm with an integer vector representation, as explained in Chapter 2. In this work the population is composed of 100 individuals, iterated for 500 generations and the following modification operators:

- Cut and Splice cross over: Two individuals are utilized as parents and a distinct crossover point is selected in each solution. The genes before and after that point

are swapped between those individuals, giving rise to two new solutions. This operator may give raise to individuals with distinct sizes from their ancestors.

- Random Mutation: An individual gene from a solution is swapped by a randomly chosen value (within allowed range of values defined by the user for that gene).

The individuals for the modification operation are chosen based on the Tournament selection procedure, with two individuals selected randomly with repetition from the population. The algorithm possesses elitism, thus the best individual of a generation is automatically part of the next generation population.

4.2.3.3 Feeding Strategy Design For Fed-Batch Bioprocesses

After the batch model has been calibrated it can be extended to support the simulation of a feeding strategy of a fed-batch fermentation by incorporating the $inflow$ term as previously explained.

These terms can be added to the respective compounds of the equation being fed into the bioreactor. Each individual added term represents the amount of a substance fed to the bioreactor by a pump. It is possible to add more pumps by adjoining the respective number of terms. In this work, a pump capable of providing glucose to the bioreactor will be added to the system by changing the glucose equation to:

$$\dot{Glc} = qs(x,\theta) \times X - \frac{f_{Glcin}}{V} \times Glc + inflow \tag{4.15}$$

The pump will also affect the bioreactor volume equation in the following way:

$$\dot{V} = \sum_{n=1}^{numberOfPumps} f_{compoundPump} \tag{4.16}$$

The summation $\sum_{n=1}^{numberOfPumps} f_{compoundPump}$ equals the total flux being added to the bioreactor. In the definition of this problem, it is assumed that the volume is bounded and that a pump cannot feed the system if the volume variable is in its upper bound.

The feeding profile design problem consists in finding an optimal/near optimal feeding strategy that optimizes a given objective function. In this work, it was assumed as the main goal the maximization of the bioprocess yield defined as:

$$\frac{\sum_{i=1}^{nProducts}(P_{i,tf} - P_{i,t0})}{\sum_{j=1}^{nSubstrates}\int_{t0}^{tf} S\,dt} \tag{4.17}$$

where the term $\sum_{i=1}^{nProducts}(P_{i,tf} - P_{i,t0})$ represents the summation of all target products i concentration in the bioreactor at the end of the experiment and $\sum_{j=1}^{nSubstrates}\int_{t0}^{tf} S\,dt$ is total substrate concentration consumed during the simulation (represented by the summation of the $nSubstrates$ in the bioreactor).

A test case was utilized to devise the feeding strategy: the maximization of the biomass yield. This test serves as a assessment to verify the exponential growth pattern of biomass. The feed strategy will be defined by a function of time that returns the pump flux rate. Two distinct EC techniques will be used to infer this function: (i) Grammatical Evolution [21]: by searching for an explicit expression for the feed expression; (ii) DE by optimizing a set of time point values that serve as basis for linear interpolation of the function;

Grammatical Evolution

The application of GE in this problem consists in finding a symbolic expression for the rate profile of glucose to maximize the objective function (for more information about the GE algorithm consult 2). When the rate value of the expression goes beyond the operational range of the feeding pump, the maximum or the minimum value of this scope are returned based on the nearest distance to the upper or lower bound, respectively.

The context free grammar (in Backus-Naur form [22]), utilized in this problem is given by:

$$
\begin{aligned}
S &::= f =< Op > \\
Op &::= < Value >< BinOp >< Op > | < UniOp > | < Value > \\
BinOp &::= +|-|/|* \\
UniOp &::= cos(< Value >)|sin(< Value >)|ln(< Value >) \\
Value &::= < Number > | < Number > . < Number > |Glc|X \\
Number &::= 1|2|3|4|5|6|7|8|9|0| < Number >
\end{aligned}
$$

The individual solution representation is allowed to wrap around and possess a maximum depth of 1000. The algorithm configuration is found on Section 4.2.3.2.

Differential Evolution

The feeding profile optimization utilizing DE in this context corresponds to finding a set of interpolation points having as independent variables a set of time points defined by the user. The function value is given by linear interpolation.

The same variant of the DE algorithm used before (with the same parametrization) was utilized for this problem.

4.3 Results and Discussion

In Figure 4.2, the time series trajectories of the participating species on the system after the calibrations are shown. The lines represent the system simulation predictions, while the dots correspond to experimental data. Lactate and formate were not measured in this experiment. However, the system predicts the excretion of formate. The same procedure was also tried out with distinct genome-scale models of *Escherichia coli*, such as IJO1360 [23] IJR904 [24] and similar results were observed.

Per contra, if these fermentative products (present in the model like, lactacte, formate, ethanol) are forced to zero (in the training data at each experimental point), the model becomes unable to reproduce the acetate production and consumption curve (results not shown).

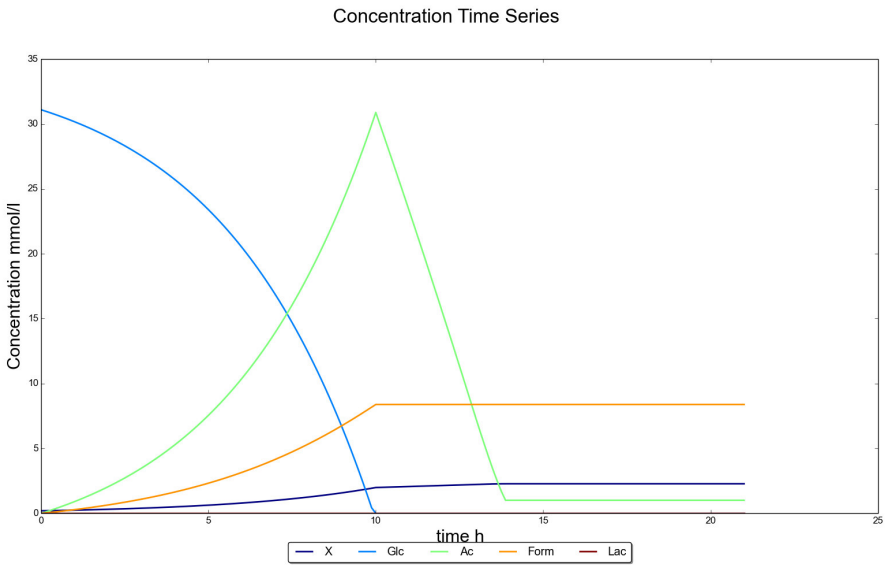


Figure 4.2: System species concentration trajectories against training data

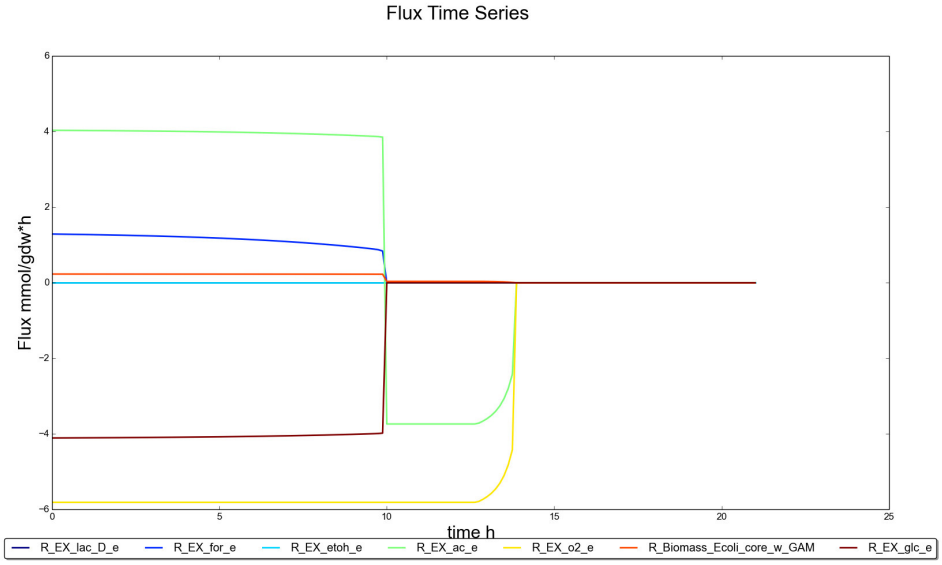


Figure 4.3: System flux trajectories

In Figure 4.3, the flux time series are shown. The system enters an dormant state around the 10th hour due to the insufficient amount of substrate in the bioreactor to sup-

port growth. This assumption has the disadvantage of forcing the complete simulation of every *in silico* experiment. In the DFBA presented in [5], the *in silico* experiment would be stopped, when the flux prediction method was infeasible (in this scenario FBA). This feature or relaxation of the problem has important consequences in the design of (near) optimal feeding profiles. Another important factor in the devised simulation method is the usage of PFBA. It was observed (results not shown) that FBA can produce abrupt changes in the flux distributions due to the high number (may be infinite) of possible valid states. When using PFBA these brusque changes tend to occur less often.

4.3.1 Calibration and Identifiability Analysis

Table 4.1: Parameter Ranking utilizing Morris Method with scaled elementary effects and r= 140 p = 10 before the calibration process

ParameterId	Rank	Value
qsmax	2	0.56
qski	4	0.14
qsks	3	0.18
qsmaxo2	1	0.73
qacmax	5	0.025
qacks	6	0.0025

Table 4.2: Parameter Ranking utilizing Morris Method with scaled elementary effects and r= 140 p = 10 after the calibration process

ParameterId	Rank	Value
qsmax	2	0.37
qski	3	0.017
qsks	6	6.9E-5
qsmaxo2	1	0.49
qacmax	5	0.0030
qacks	4	0.0023

Table 4.3: Parameter first order sensitivity indexes utilizing QRS-HDMR with simulation data, after calibration

ParameterId	X	Ac	Glc	Form
qsmax	0.41	0.36	0.77	0.81
qski	1.5E-4	5.1E-4	1.6E-5	3.4E-5
qsmaxo2	0.43	0.40	0.081	0.37
qsks	0.0010	0.0011	3.77E-5	6.74E-4
qacmax	0.0078	0.0073	0.0024	0.0030
qacks	0.0024	0.0022	3.87E-5	0.0015

Table 4.4: Most Relevant parameter second order interactions sensitivity indexes utilizing QRS-HDMR with simulation data, after calibration

ParameterId	ParameterId	X	Ac	Glc	Form
qsmax	qsmaxo2	0.0071	0.014	0.037	0.0
qski	qsks	0.0010	0.0014	5.74E-4	0.0
qsmaxo2	qacmax	0.069	0.054	0.0	0.0

The analysis of the calibration process regarding the model parameters was assessed utilizing the Morris method with scaled elementary effects. All the parameters were deemed as significant for the production of the model behavior within the bounds defined for each parameter. In Table 4.1, the scaled elementary effects and the parameter ranking before the calibration process are shown. The Morris method was executed with parameters having a larger sample space, due to uncertainty regarding their values (all the parameters were allowed to vary between -3 and 3 in log-scale). In Table 4.2, the scaled Morris elementary effects, after the calibration process (with parameters possessing 50% variability regarding the best solution found in the calibration process) are shown.

It can be seen, in both tables, that the most important parameters driving the calibration process are *qsmax* and *qo*2*max*. Nonetheless, in the first analysis (before the calibration process) *qsks* was ranked as the third most important parameter,while after the

calibration it was less important. In the first scenario, this parameter could take a wide range of values, while after the calibration the value was fixed to a small value that had a small impact on the calibration cost function. The *a priori* screening analysis (Morris method before the calibration), was followed by a correlation analysis (explained in the methods) to identify non-influential parameters. Notwithstanding, all the parameters were deemed as influential. It is important to note that parameters may be deemed less important due to the reduced number of times a phenomena occurs in a set of *in silico* experiments such as the consumption of acetate when contrasted with the consumption of glucose.

Afterwards,the factors driving the model output trajectories were assessed by utilizing the first and second order sensitivity indexes (utilizing QRS-HDMR for this computation). This method allows to designate how the factors interact and affect the output, as well as, the identifiability characteristics of the system (without distinguishing between structural or practical identifiability). A sample of system simulations were computed utilizing Latin Hyper Cube Sampling (LHCS) to sample the parameter space with 1 order magnitude variability, regarding the best parameter set found in the calibration process. Next, the solutions that were less than 12% worse, when compared to the best solution were kept. This threshold was chosen empirically, due to the capability of the system with this error level to represent the experimental behavior in training data. Subsequently, the sensitivity indexes were computed utilizing QRS-HDMR.

The sensitivity indexes computed in this problem are shown in Tables 4.3 and 4.4.

We can conclude the following from the GSA analysis of the calibration process:

- The least important parameter driving the distinct ouputs profiles is $qski$. This parameter has the lowest first sensitivity index value in all output variables. Nonetheless, it is confounded for biomass and acetate with $qsks$, as it can be observed in Table 4.4.

- The parameters that drive the estimation procedure are the ones that also define the behavior the outputs after the calibration as it can be seen in Tables 4.1, 4.3

and 4.4. It is important to take into account the selection procedure, utilized in the sampling process for the computation of the global sensitivity indexes (as explained previously);

- Analysis of the second order sensitivity indexes in the first three output variables in the Table 4.4, reveals that there is an interaction between $qsmax$, $qsmaxo2$ and $qacmax$;

- Formate profile is only affected by first order effects. However, the sum of the first order sensitivities has a value larger than one (thus, the sensitivity index value has not converged). This can be explained due to the small subset of selected simulations covering the excretion of formate.

It also is important to bear in mind that the utilized objective function for the calibration process is not monotone regarding the model parameters and only methods such as the ones utilized (variance based methods) should be employed to characterize its parameter structure.

Lactate and ethanol do not appear in the table due to fact of their concentrations being always zero in all the carried simulations.

From the data in the previous tables, we can conclude that parameters (assuming we can measure all variables at equidistant time points at 0.125 hours) $qsmax$ and $qo2max$ can be computed from the available experimental data due to the small differences concerning the respective first and total sensitivity indices.

The parameter set composed by $qacks$, $qski$ and $qacmax$ may not be successfully identified due to the presence of higher order interactions of the same magnitude or larger than the main effects, that confound the parameter values. These parameters may become identifiable with suitable experimental data that excites the system in specific ways.

After the identifiability analysis and the calibration of the model parameters, the feeding profile design algorithms with the purpose of maximizing the micro-organism biomass, while minimizing the amount of substrate utilized in the process,were exe-

cuted. Both optimization algorithms (DE and GE) described in the methods section were utilized. In this work, the original DFBA formulation was relaxed to allow the absence of growth in the cell (almost like setting the microorganism in a dormant state, where all the fluxes have the value zero). It is curious to note that the cell will often tend to feeding profiles where the absence of growth is not present. Without this relaxation when the flux distribution algorithm returns an infeasible result, the optimization algorithms tend to get stuck in local optima (or at least in a worse solution than the one found considering the relaxation). Typically, the algorithm will pass through intermediary states (that may compromise the absence of growth) until they arrive to the best solution found.

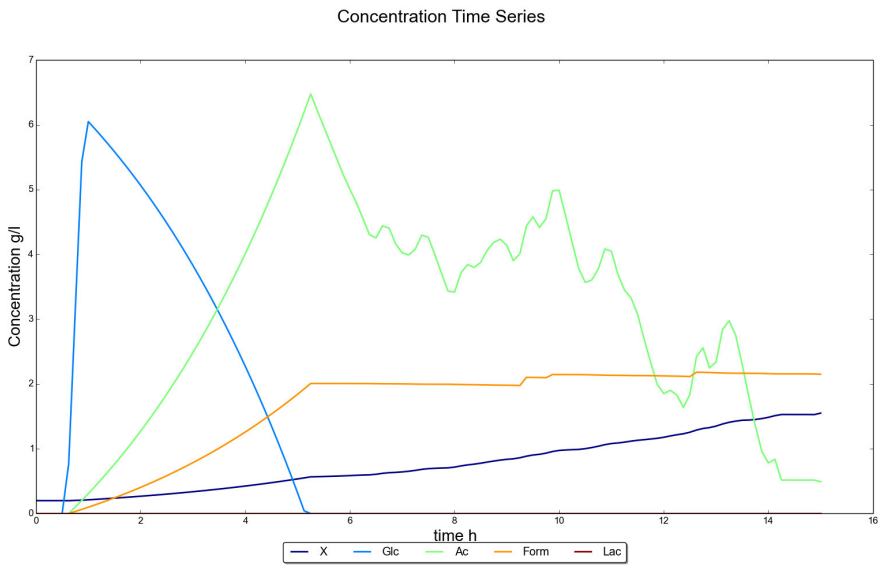


Figure 4.4: Best Differential Evolution Solution Found - Concentrations

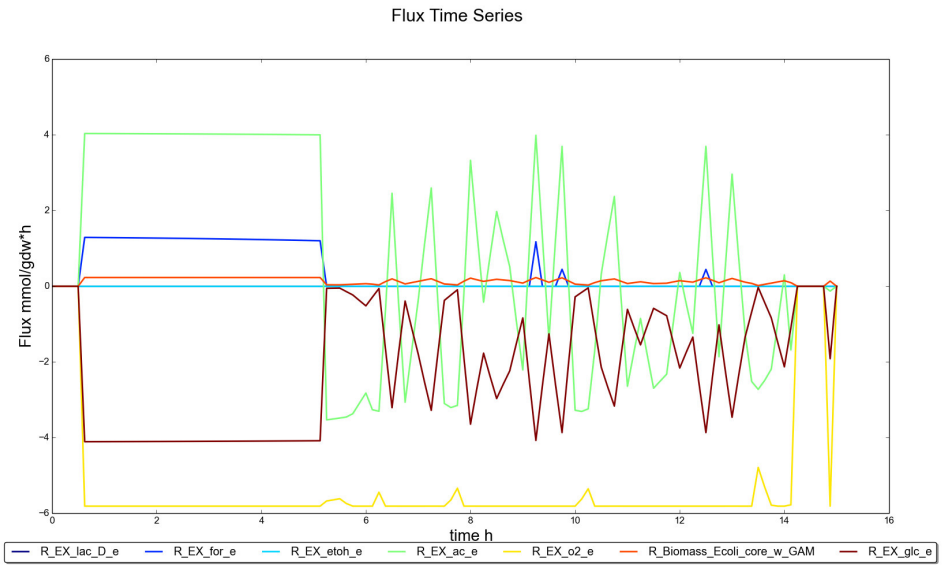


Figure 4.5: Best Differential Evolution Solution Found - Fluxes

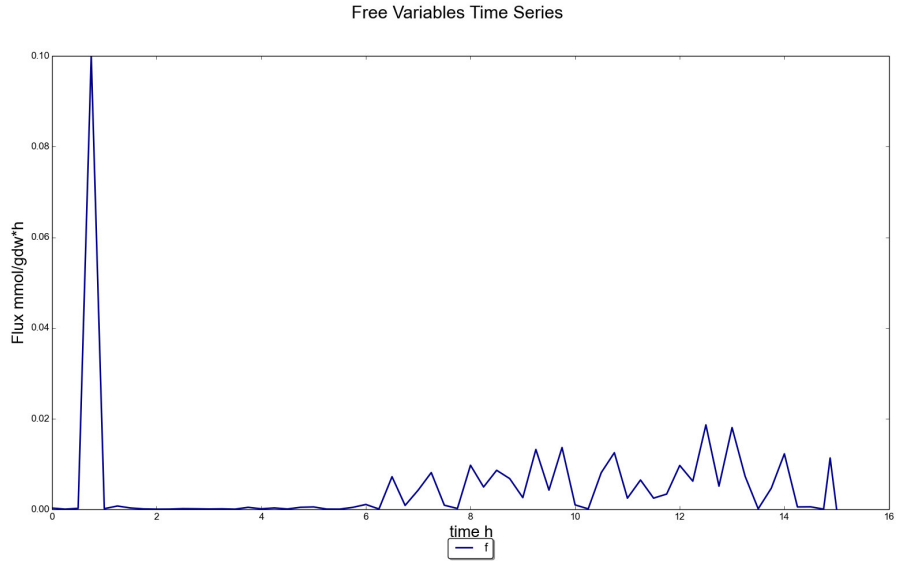


Figure 4.6: Best Differential Evolution Solution Found - Feed Profile with objective function value of 0.063

In Figures 4.4, 4.5, and 4.6 the feeding profiles generated by DE, are shown. This

algorithm has several difficulties in converging to a good solution (only with iterative tuning of the best solution found, it is possible to reach to a solution as good as GE). Often, the algorithm will find a local optimum, where the problem is treated almost like a batch fermentation, as it can be seen in Figure 4.4. In this solution, the algorithm chooses to introduce a large amount of substrate in the bioreactor, at the beginning of the simulation, followed by a period where only a small amount of flux feed is added to the bioreactor. This feeding profile, leads to the excretion of by-products, such as acetate and formate, what indicates a waste of carbon that could be directly to biomass, if the cell had the metabolic capacity to deal with the amount of glucose in the bioreactor at those times (excretion of by-products). Acetate is also consumed at the end part of the simulation, what is least efficient (growth wise), than the consumption of glucose.

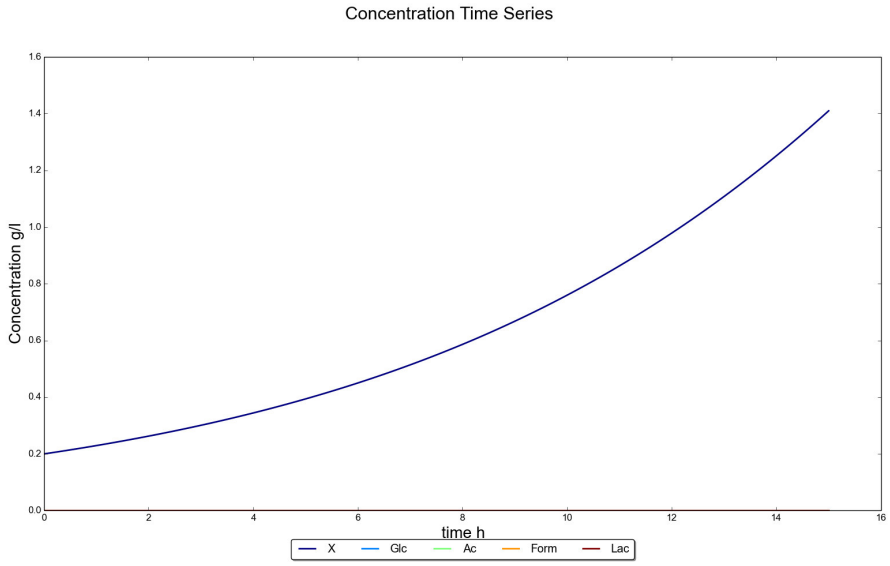


Figure 4.7: Best Grammatical Evolution Solution Found - Concentrations

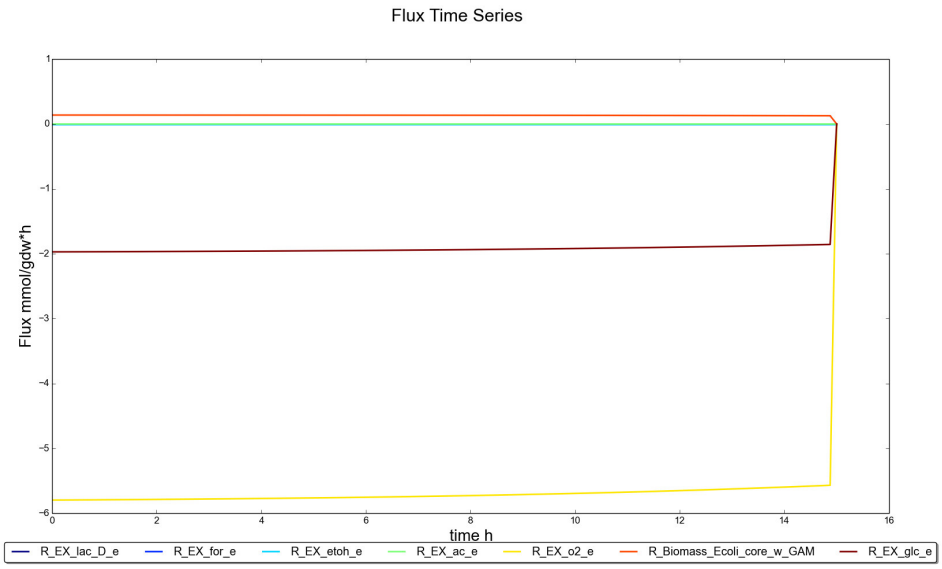


Figure 4.8: Best Grammatical Evolution Solution Found - Fluxes

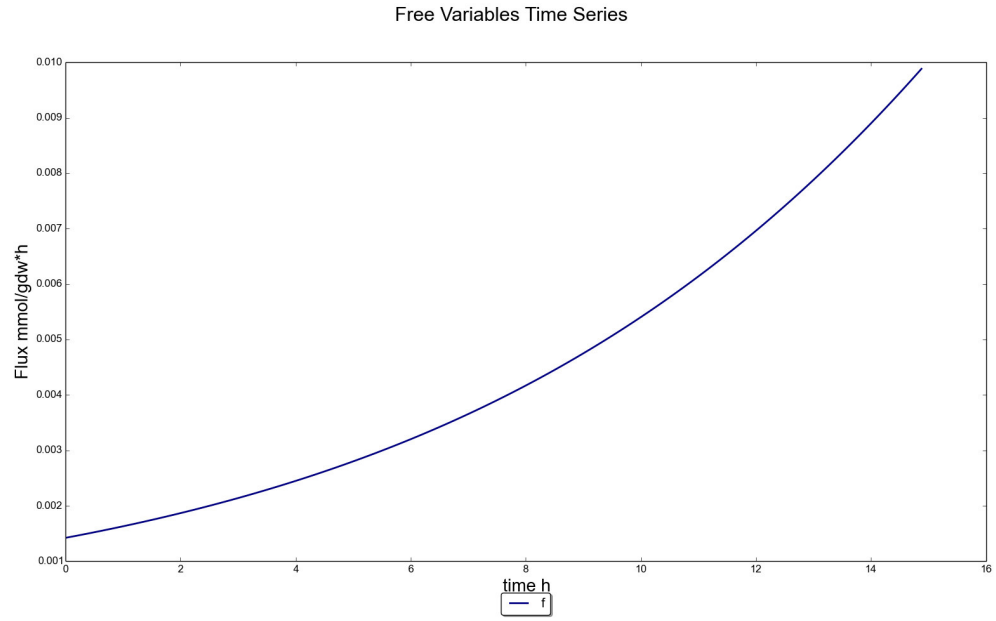


Figure 4.9: Best Grammatical Evolution Solution Found - Feed Profile with objective function value of 0.067

In Figures 4.7, 4.8, and 4.9, the time series of the concentrations, fluxes and feed profile, regarding the best GE solution found, are shown. In equation 4.18 the best symbolic expression devised by the GE algorithm is presented.

$$f = Glc + \frac{X}{\frac{7}{e^{3.0}}} \tag{4.18}$$

In this solution (due to the feed profile), the cell is redirecting all available carbon to biomass, in the most efficient way possible, by setting the value of the glucose flux near its critical point (defined as flux bound , where excess carbon has to redirected to other exchange fluxes, such as acetate). Contrarily to the DE the GE feed profile is continuous over the simulation time. This affects the ODE simulation part of the method, where DE may possess several discontinuous flux feed levels,and cannot adjust the flux feed to the exact cell needs. Thus, GE does not have to deal with the number of collocation points of interpolation, what alleviates the convergence of the method (often leading to better solutions). On the other hand, DE possesses difficulties in converging to a good solution, and has to be iteratively fine tuned (often leading to poorer quality solutions than GE).

The method possesses a caveat, when the distribution calculation method returns a flux distribution from a set of possible valid flux distributions - the obtained result becomes solver dependent. One way to alleviate this hurdle is to utilize a method, such as Geometric FBA [25], that returns the same flux distribution independently from the solver.

4.4 Conclusion

This work contemplated the extension of Dynamic Flux Balance Analysis supporting the absence of substrate in the bioreactor by relaxing the need of a valid result when computing the cell flux distribution and by incorporating a new formulation regarding the substrate consumption by the cells.

The method was tested by first calibrating a model describing a bioprocess ODE system and the interaction with an *Escherichia coli* stoichiometric model. The calibration process was preceded by two global sensitivity analysis: firstly, the influence of the model parameters in the calibration cost function was assessed by ranking the parameters utilizing the Morris Method and several extensions described in the literature (described in the methods section), before and the after the calibration process; next, the identification of the underlying identifiability structure was computed by constructing the first and second order sensitivity indexes using QRS-HDMR as base model.

Afterwards, the proposed method was utilized to design fed-batch feeding profiles to attain maximum yields utilizing GE or DE. DE tends, to converge to sub-optimal solutions, contrarily to GE.

This method allows to improve and design new feed strategies, thus having implications at improving current industrial bioprocesses that can be described mathematically by this formulation. The GSA methods allow to pinpoint possible future modifications to the model as well identifying the parameters driving the calibration process (before executing the model calibration). This method also allows to take advantage of the growing number of genome-scale models described in the literature.

However, when employing the devised method it is important to bear in mind that, if the method used to compute the flux distribution possesses several possible solutions, the obtained results will be solver dependent.

4.5 Acknowledgements

Paulo Vilaça participated actively as an co-author in the discussion and method development.

Sónia Carneiro participated in the discussion and supplied the experimental data.

REFERENCES

[1] G. Bastin and D. Dochain, "On-line estimation and adaptive control of bioreactors: Elsevier, amsterdam, 1990 (isbn 0-444-88430-0). xiv+ 379 pp. price us 146.25 dfl. 285.00," *Analytica Chimica Acta*, vol. 243, p. 324, 1991.

[2] K. J. Kauffman, P. Prakash, and J. S. Edwards, "Advances in flux balance analysis," *Current opinion in biotechnology*, vol. 14, no. 5, pp. 491–496, 2003.

[3] N. Ishii, K. Nakahigashi, T. Baba, M. Robert, T. Soga, A. Kanai, T. Hirasawa, M. Naba, K. Hirai, A. Hoque, *et al.*, "Multiple high-throughput analyses monitor the response of *E. coli* to perturbations," *Science*, vol. 316, no. 5824, pp. 593–597, 2007.

[4] A. Varma and B. O. Palsson, "Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *E. coli* w3110.," *Applied and environmental microbiology*, vol. 60, no. 10, pp. 3724–3731, 1994.

[5] R. Mahadevan, J. S. Edwards, and F. J. Doyle, "Dynamic flux balance analysis of diauxic growth in *E. coli*," *Biophysical journal*, vol. 83, no. 3, pp. 1331–1340, 2002.

[6] J. L. Hjersted and M. A. Henson, "Optimization of fed-batch *Saccharomyces cerevisiae* fermentation using dynamic flux balance models," *Biotechnology progress*, vol. 22, no. 5, pp. 1239–1248, 2006.

[7] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[8] R. C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*, vol. 12. SIAM, 2013.

[9] G. Li, S.-W. Wang, H. Rabitz, S. Wang, and P. Jaffé, "Global uncertainty assessments by high dimensional model representations (hdmr)," *Chemical Engineering Science*, vol. 57, no. 21, pp. 4445–4460, 2002.

[10] G. J. McRae, J. W. Tilden, and J. H. Seinfeld, "Global sensitivity analysis—a computational implementation of the fourier amplitude sensitivity test (fast)," *Computers & Chemical Engineering*, vol. 6, no. 1, pp. 15–25, 1982.

[11] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[12] H. Rabitz, Ö. F. Aliş, J. Shorter, and K. Shim, "Efficient input—output model representations," *Computer Physics Communications*, vol. 117, no. 1, pp. 11–20, 1999.

[13] J. D. Orth, R. M. Fleming, and B. Ø. Palsson, "Reconstruction and use of microbial metabolic networks: the core *E. coli* metabolic model as an educational guide," *EcoSal Plus*, vol. 4, no. 1, 2010.

[14] T. Sumner, E. Shephard, and I. Bogle, "A methodology for global-sensitivity analysis of time-dependent outputs in systems biology modelling," *Journal of The Royal Society Interface*, vol. 9, no. 74, pp. 2156–2166, 2012.

[15] S. Carneiro, "A systems biology approach for the characterization of metabolic bottlenecks in recombinant protein production processes," 2010.

[16] N. E. Lewis, K. K. Hixson, T. M. Conrad, J. A. Lerman, P. Charusanti, A. D. Polpitiya, J. N. Adkins, G. Schramm, S. O. Purvine, D. Lopez-Ferrer, *et al.*, "Omic data from evolved *E. coli* are consistent with computed optimal growth from genome-scale models," *Molecular systems biology*, vol. 6, no. 1, 2010.

[17] G. Sin and K. V. Gernaey, "Improving the morris method for sensitivity analysis by scaling the elementary effects," *Computer Aided Chemical Engineering*, vol. 26, pp. 925–930, 2009.

[18] G. Li, S.-W. Wang, and H. Rabitz, "Practical approaches to construct rs-hdmr component functions," *The Journal of Physical Chemistry A*, vol. 106, no. 37, pp. 8721–8733, 2002.

[19] M. Ruano, J. Ribes, A. Seco, and J. Ferrer, "An improved sampling strategy based on trajectory design for application of the morris method to systems with many input factors," *Environmental Modelling & Software*, vol. 37, pp. 103–109, 2012.

[20] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[21] C. Ryan, J. Collins, and M. O. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *Genetic Programming*, pp. 83–96, Springer, 1998.

[22] B. Augmented, "for syntax specifications: Abnf (rfc 2234)," 1997.

[23] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. Ø. Palsson, "A comprehensive genome-scale reconstruction of escherichia coli metabolism—2011," *Molecular systems biology*, vol. 7, no. 1, 2011.

[24] J. L. Reed, T. D. Vo, C. H. Schilling, B. O. Palsson, *et al.*, "An expanded genome-scale model of *E. coli* k-12 (ijr904 gsm/gpr)," *Genome Biol*, vol. 4, no. 9, p. R54, 2003.

[25] K. Smallbone and E. Simeonidis, "Flux balance analysis: a geometric perspective," *Journal of theoretical biology*, vol. 258, no. 2, pp. 311–315, 2009.

CHAPTER 5

EVOLUTIONARY COMPUTATION FOR PREDICTING OPTIMAL REACTION KNOCKOUTS AND ENZYME MODULATION STRATEGIES

One of the main purposes of Metabolic Engineering is the quantitative prediction of cell behaviour under selected genetic modifications. These methods can then be used to support adequate strain optimization algorithms in a outer layer. The purpose of the present study is to explore methods in which dynamical models provide for phenotype simulation methods, that will be used as a basis for strain optimization algorithms to indicate enzyme under/over expression or deletion of a few reactions as to maximize the production of compounds with industrial interest. This work details the developed optimization algorithms, based on Evolutionary Computation approaches, to enhance the production of a target metabolite by finding an adequate set of reaction deletions or by changing the levels of expression of a set of enzymes. To properly evaluate the strains, the ratio of the flux value associated with the target metabolite divided by the wild-type counterpart was employed as a fitness function. The devised algorithms were applied to the maximization of Serine production by Escherichia coli, using a dynamic kinetic model of the central carbon metabolism. In this case study, the proposed algorithms reached a set of solutions with higher quality, as compared to the ones described in the literature using distinct optimization techniques

5.1 Introduction

Progress in molecular biology technologies permitted uncovering new molecular interactions aiding in the better characterization of cells. Modeling a cell based on the understanding of the interplay of its constituents, in connection with information from different omics, is the purpose of Systems Biology (SB) as advocated by Kitano [1].

The application of engineering concepts to SB provides valuable insights, helping to consolidate ongoing efforts in Biotechnology. Of particular interest, in the scope of this work, is Metabolic Engineering (ME). This discipline is concerned with the understanding and use of metabolic pathway modifications, using biological models under an engineering perspective to attain a specific industrial objective [2].

There has been a trend in industry to replace chemical synthesis techniques by biotechnological processes, due to environmental and sustainability concerns. Optimization of microbial strains has an important role in this scenario, due to increases in bioprocess productivity and, consequently, in profitability. Generally, the metabolism of wild-type microorganisms is geared to its survival and reproduction, without engaging in the production of compounds outside this scope. Thus, the metabolism has to be modified in order to meet the desired industrial outcome, typically the overproduction of a target compound.

Until recently, in bioprocess engineering, cells were modeled as black box entities responsible for consuming substrates and producing certain compounds, ignoring the underlying biological mechanisms. The genetic improvement of microorganisms has been driven by selective pressure based on empirical principles to obtain organisms with desired characteristics.

More recently, rational approaches for ME have been proposed, where researchers attempt to build mechanistic whole cell models to elucidate and provide tools for studying metabolic responses under different environments and perturbations. However, these still face hurdles such as the lack of knowledge about the reaction kinetics and the cellular responses to specific external perturbations. Nonetheless, ME has paved the way

to induce cells to over-synthesize target products, to engineer new metabolic pathways and to control the production of a set of metabolites of interest.

The prediction of metabolic states has been accomplished mainly by the use of genome-scale stoichiometric models and constraint based phenotype simulation methods. These are developed based on a microorganism's specific biochemical network, using mass balances and reaction flux constraints derived from biophysical principles or empirical observations

Several stoichiometric genome scale models have been published in the literature for microorganisms such as *Escherichia coli* [3] and *Saccharomyces cerevisiae* [4]. Typically, these models do not contain kinetic and/or regulatory information. Even so, it is possible to predict cellular behavior under certain assumptions (e.g. pseudo steady-state). From a ME point of view, these models allow to investigate the response of a metabolic network to specific genetic manipulations and/or environmental conditions.

There are several simulation methods that can be employed to estimate the microorganism flux distribution such as Flux Balance Analysis (FBA) [5], minimization of metabolic adjustment (MOMA) [6] and Regulatory on/off minimization of metabolic flux changes(ROOM) [7]. Each of these methods returns a unique optimal solution from the solution space, but in many cases several optimal solutions may exist and there is no information concerning which of those is indeed used by the cell. Thus, it is hard to identify the cell's true state [8]. Also, the employed objective functions may not represent the biological reality and other objectives for the cell can be considered instead [9].

Despite the described limitations, these methods can provide useful insights for ME. Several tools have been developed in the last years to calculate the best set of reaction (or gene) deletions or levels of expression of enzyme sets to attain a specific objective. Within this context, the problem of finding a gene/ reaction knockout set belongs to the class of combinatorial optimization [10], while the reaction down/up regulation task is included in the numerical optimization class. It is not feasible to test all gene/ reaction deletion combinations or enzyme expression level values using a brute force approach in a reasonable amount of time.

OptKnock [11] provides an alternative for the reaction deletion task, based on a MILP formulation, finding an optimal set of reactions to delete. However, it is constrained to linear objective functions and it cannot be applied with large networks due to the NP complexity of the problem [12].

OptGene [10] tackles this problem using Evolutionary Algorithms (EAs), where solutions are encoded in a specific representation scheme, in conjunction with FBA to estimate the effect of certain sets of reaction deletions. This method gives no guarantees of finding the best global reaction deletion set, but often provides (near) optimal solutions in a reasonable time being also more flexible in terms of the definition of the fitness functions. In recent work, other variants of Evolutionary Computation (EC) approaches such as set-based representation EAs and Simulated Annealing have been proposed and evaluated [13]. Also, methods that try to estimate the best under/over expression levels for a set of enzymes have been proposed, namely OptReg [14] that is based on a MILP formulation and more recently EAs [15].

An important shortcoming of all these methods based on constraint-based approaches is the absence of dynamic features concerning the metabolic state, not allowing to cope with enzyme kinetics and regulation. Therefore, the obtained results do not portray these effects and are bound to be incomplete.

One approach to overcome these hurdles is to use dynamic models. These models are usually based in ordinary differential equations and produce a more detailed description of cellular systems by capturing transient behavior. This type of models mimic better the phenomena observed *in vivo* in microbial strains than its purely stoichiometric counterparts. These mathematical abstractions may also allow obtaining a specific steady state from an initial set of conditions.

On the down side, they require detailed enzyme kinetic information that is often incomplete and spread across several databases. This gives rise to inconsistencies due to the unavailability of experimental data and methodology standardization concerning the estimation of kinetic parameters. Another hurdle is the imprecise knowledge of the mechanistic rate laws underlying several reactions. It is important to bear in mind that

it is not usually possible to measure all cellular compounds precisely in order to build the respective kinetics. On the whole, these models account for a small part of the metabolism. These obstacles can be attenuated by utilizing kinetic law approximations [16].

Despite the limitations, and due to the use of kinetic information, dynamic models are able to represent enzyme interactions not possible with steady-state models, such as metabolic inhibition. These models are also better suited to simulate the effects of enzyme expression level changes.

Therefore, and in spite of the lack of information to build large-scale dynamic models, a few attempts have been made regarding their use in ME applications. In [17] a Mixed Integer Non-Linear Progamming (MINLP) method for finding optimal modulation strategies was developed. The main limitations of this method are computational tractability and the generation of optimal sets of modifications [18].

In [19] the problem of finding the best set of enzyme expression levels and reaction knockouts using a dynamic model of central carbon metabolism of *Eschericia coli* [20] was addressed. A MILP formulation and a generalized linearization of the kinetic model were used to find a ME strategy. However, like in Optknock [11] the effort to solve a MILP problem increases exponentially with the size of the problem at hand. This method also assumes flux and concentration bounds around the reference state, to control the error of the linerarized model regarding the original model.

In [21], the problem of finding the best set of enzyme expression levels using the aforementioned model was addressed. Simulated Annealing [22] was used to search the enzyme set space, while a sequential quadratic programming method estimated the respective enzyme expression levels, forcing the objective function and the constraints to be continuous in the considered ranges and of class C^2. This method assumes a value for the overall maximum allowed metabolite changes at steady-state and also that overall system enzyme levels remain constant within a constant value proportional to the number of modifications. In this work this constraint will not be used due to its specificity and lack of experimental data to corroborate it in Escherichia coli. This

restriction may also limit the algorithm generalization capabilities.

5.1.1 Aims and overview of the approach

This work entails the development of EC approaches to find a set of metabolic modifications, such as reactions knockouts and reaction up/down regulation that will optimize the production of a metabolite with an industrial interest, utilizing as a basis for simulation dynamic models composed of ordinary differential equations (ODEs).

One aim of this method is to provide a proof of concept of a scalable approach able to deal with larger scale dynamic metabolic models than the ones that currently exist. The existing methods are not able to cope with the definition of ME strategies in larger dynamic metabolic models, due to the combinatorial increase in the number of possible strategies. Also, they do not take into consideration invalid solutions that may contain valid building blocks for the optimal solution. It is important to bear in mind that it is impossible to scan the whole state space, when it has a high dimensionality. A brute force approach is not feasible for the enzyme level modulation task and it becomes unpractical in the reaction deletion scenario as the number of modifications increases.

Also, most of the current methods deal with the parallel optimization of enzyme and knockout expressions by employing a Mixed Integer Non-Linear Programming methods [17] that are unable to solve problems with hundreds of equations, or rely on the approximation of the non-linear dynamic model around a reference state (usually a steady state) and a posteriori use a MILP formulation [19]. The approximation of the non-linear dynamic model around a reference state also enforces the use of reaction and metabolite ranges around the reference state that may exclude valid solutions of interest.

The present approach deals with these shortfalls by using the original non-linear model without doing any approximation and by searching ME strategies by means of EAs that adapt the solution size. Thus, this method does not need to assume a range of flux and metabolite values where solutions are considered valid. The assumptions in the current method are made in a reaction basis when defining the discrete or continuous

range for modifiable reactions.

In this situation, dynamic models are used to generate a single steady-state solution without the need of specifying further assumptions such as in the cases of FBA, MOMA, or ROOM for stoichiometric models. Another advantage of using these models in ME applications is the straightforward implementation of over/under expression of enzymes as ME strategies.

Two tasks are used to test the devised optimization techniques, whose purpose is to maximize the production of a metabolite at steady-state: (i) *reaction deletion* - the objective is to discover the best set of reaction deletions (knockouts). The ideal number of reactions to remove is also determined simultaneously; and, (ii) *reaction up/down regulation* - the main goal is to find the best set of enzymes to tweak and the respective level of expression concerning the base values present in the original model.

In this work, a novel encoding scheme is proposed that will adress both tasks, allowing the algorithm to choose the best ME strategy given a permitted set of constraints, as well as the number of modifications. The solution decoding affects the simulation of the dynamic model by multiplying the v_{max} parameter of the reaction rate law by the decoded enzyme modulation level contained in the solution's genome. This corresponds to a change in the total enzyme concentration assuming that v_{max} is directly proportional to it. In the reaction deletion case, the v_{max} parameter is multiplied by zero, therefore constraining the reaction's flux to 0.

The design of the algorithm also allows the discretization of the enzyme modulation value into a set of pre-defined ranges. In a wet lab setting it is often not possible to fine tune the exact enzyme expression levels as returned by the algorithm. Thus, this discretization may allow a more flexible representation of what may be achieved *in vitro*. This representation provides for the simultaneous optimization of discrete and continuous enzyme levels. The developed method also allows to incorporate non-modifiable reactions (reactions that cannot be tweaked by the algorithm or have to respect specific constraints, like for example directionality or flux intervals).

As a basis for phenotype simulation, a metabolic dynamical model of selected path-

ways of *Eschericia coli* will be used, based on ordinary differential equations, namely, the mechanistic model of the central carbon metabolism [20] consisting of mass balance equations for glycolysis and for the pentose-phosphate pathway.

The aforementioned tasks are used in a case study related to the maximization of Serine production, allowing to contrast the obtained results to the ones published in [19]. Co-metabolite concentrations were assumed to be constant as in the previous study. Nowadays, Serine plays a major role in several industrial applications. Serine is used in cosmetic and food industries and is produced by fermentative routes [23]. In this case study, it is not apparent how to find the best set of genetic modifications to enhance the production of Serine due to the high number of interacting reactions.

5.2 Methods

5.2.1 Mechanistic model of the central carbon metabolism

The mechanistic model of the central carbon metabolism [20] encompasses the phosphotransferase system, glycolysis and the pentose-phosphate pathway. This model is curated and available from Biomodels [24]. In Figure 5.1, a schematic representation of the reaction network is shown. The mass balances take the following form:

$$\frac{dX}{dt} = S\,V - \mu X \tag{5.1}$$

where X represents the vector of metabolite concentrations, μ is the specific growth rate, S is the stoichiometric coefficient matrix and V is the reaction rate vector. The equation for extra-cellular glucose has the following form:

$$\frac{d[GlcExt]}{dt} = D([GlcFeed] - [GlcExt]) + fPulse - \frac{[bio]v_{PTS}}{\rho_{bio}} \tag{5.2}$$

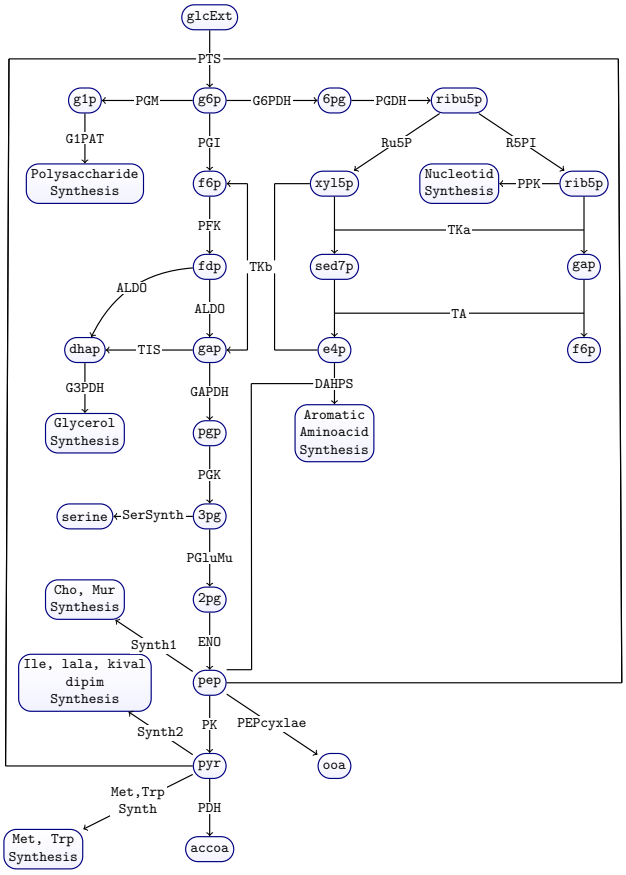


Figure 5.1: *Escherichia coli* central carbon metabolism network.

where $[GlcExt]$ represents the external glucose concentration, $[GlcFeed]$ is the concentration of glucose in the feed, $fPulse$ is a function allowing to introduce glucose pulses, $[bio]$ is the biomass concentration, ρ_{bio} is the biomass density and v_{PTS} is the flux through the phosphotransferase system reaction.

The reaction fluxes at steady-state are described by:

$$v_i^0 = v_{iMax} f_i(X_i^0, P_i^0) \tag{5.3}$$

where the superscript ^{0} denotes a variable at steady-state, v_i^0 is the rate of reaction i at steady-state, v_{iMax} is the maximum reaction rate and $f_i(X_i^0, P_i^0)$ is a function of X_i^0

metabolite concentrations at steady-state that participate in the reaction in conjunction with a set of parameters P_i^0. Thus, the v_{iMax} for each reaction is computed by the following equation (as described in [20]):

$$v_{iMax} = \frac{v_i^0}{f_i(X_i^0, P_i^0)} \tag{5.4}$$

5.2.2 Objective function formulation

The Reaction deletion and Enzyme over/under expression problems can be stated as the maximization of $\frac{vMutant_j^0}{vWildType_j^0}$, where $vMutant_j^0$ and $vWildType_j^0$ represent the flux for the target reaction j at steady-state in the mutant and in the wild-type strains, respectively.

5.2.3 Solution evaluation

To assign a fitness value for each solution suggested by the evolutionary method, the following algorithm was used:

1. Perform the model modifications by decoding the solution being evaluated, described in the next section;
2. Simulate the modified model, by adding the constraints from the solution decoded and performing the numerical integration of the ODEs in the model in a given time range;
3. Verify whether metabolite concentrations do not change significantly in a given time range encompassing the end of the simulation. If this condition is met, the system is considered to be in steady-state.
4. If the previous step is completed with success, the ratio of the solution target flux by the wild-type strain target flux value is returned. Otherwise, zero is returned.

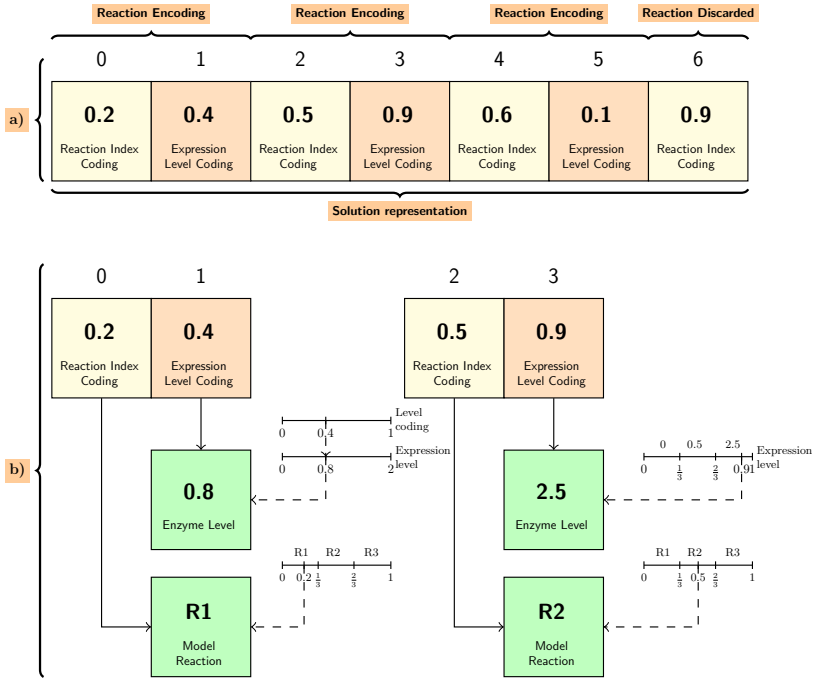


Figure 5.2: Enzyme expression level solution decoding example. In a) a solution encoding for a model with reaction set $\{R1, R2, R3\}$ is shown. In b) the decoding process for the first two reactions is illustrated.

5.2.4 Solution encoding

In this work, a novel variable size representation for inferring enzyme expression levels was developed. This representation allows searching simultaneously for the set of enzymes to modify and the respective expression level. The expression level can be a real number in a specific interval or a set of discrete values defined by the user. In the proposed representation, solutions are quite simple, being represented as vectors of real numbers with values between 0 and 1.

When considering enzyme expression levels optimization, the values in an even position are mapped to a reaction index, while the values in the following odd position encode the enzyme expression level for that reaction, in a continuous or discrete interval. Each reaction may have different expression level modulation ranges.

In Figure 5.2, the solution decoding process is illustrated with an example consid-

ering a very simple model with three reactions R1, R2 and R3. In a), it is possible to observe that each consecutive pair of elements encodes a reaction index and its enzyme level modulation. The reaction in position six is discarded because it does not possess an enzyme modulation part.

In b), the decoding process of the first two reactions is shown. The reaction index encoded at position zero is mapped to reaction R1, as follows: the interval $[0, 1]$ is divided into three equal spaced sub-intervals (the number of reactions in the model) being each interval mapped to a reaction. The interval that contains the encoded value maps to the specific model reaction. In the example, the value 0.2 is contained in the interval $[0, \frac{1}{3}]$, that is mapped to reaction R1. Position one encoded the enzyme level for that reaction (R1). This reaction was defined as varying in the continuous interval $[0, 2]$. In this scenario, the expression level coding value 0.4 is multiplied by the interval length 2 giving the expression level equal to 0.8. Note that in this case, the lower limit of both intervals is zero and therefore the mapping is easier; in general, a mapping to an interval $[a, b]$ is obtained by multiplying the encoded value by $b - a$ and adding a.

The reaction index of the next modulation in position 2 is calculated as in the previous case, corresponding to the mapping of 0.5 to reaction R2. In this case, it is assumed that reaction R2 modulation can only vary in a discrete set of values $\{0, 0.5, 2.5\}$. In this case, the mapping of the enzyme modulation level occurs in an analogous way to the reaction index mapping and, therefore, the value 0.9 at position 3 is mapped to a modulation of 2.5. If a solution has several occurrences of the same index, only the last one is considered.

The same representation can be used for representing reaction deletions (knockouts). In this case, all expression level coding positions encode a discrete set with the value zero for the modulation level.

5.2.5 Reproduction operators

For reproduction purposes within the EA, the following operators are used:

- *Random mutation*: replaces an element of the vector by another, randomly generated in the allowed range.
- *Cut and splice crossover*: A distinct crossover point is selected in both parents and the genes before and after that point are swapped giving rise to two new individuals. This operator has the capacity to modify the length of the resulting offspring.

In the proposed EA the operators have the following probabilities of being selected to generate new solutions from the selected parents: the mutation operator has 10% probability of being chosen while the crossover operator has 90% probability.

5.2.6 Experimental setup

In the first step of the evaluation function, the time course simulation is computed for the time interval $[0, 1E6]$ seconds. The system is considered in steady-state if the metabolite concentration change is inferior to 5% in the interval $[1E4, 1E6]$ seconds.

The enzyme up/down regulation allows reaction fluxes to vary by a multiple in the linear interval $[0, 2]$. The upper bound value was chosen based on the employed values by [19] with the linearized models. This reaction modulation range needs to be imposed in order to model the experimental capacity and the physiological reality inside the cell.

The algorithms are executed with an incremental number of restricted modifications from one up to six. In the case study, the algorithms for each problem are executed 30 times. In the knockout task the algorithm is run for 250 iterations, while in the enzyme over/under expression the algorithm is executed for 500 iterations, values that allow the convergence of the EA. Both algorithms employ the following configuration:

- *Population size*: 100 individuals;
- *Population initialization*: individuals are generated randomly with size varying between 1 and 100;
- *Elitism value*: 1 individual (the best) is always kept;

Table 5.1: Knockout task - best solutions

#Modifications	Algorithm	Modifications	Fitness (v_j/v_{j0}^0)	Mean Fitness ±95% Confidence Interval
	EAK	PEPC	**1.149**	$1.149 \pm 8.082 \times 10^{-17}$
1	VLS	DHAPS	1.057	–
	VLL	PEPC	**1.149**	–
	EAK	PEPC DHAPS	**1.254**	$1.254 \pm 8.082 \times 10^{-17}$
2	VLS	DHAPS G1PAT	1.073	–
	VLL	PEPC PK	1.226	–
	EAK	PEPC DHAPS PGM	**1.352**	$1.352 \pm 1.765 \times 10^{-5}$
3	VLS	DHAPS Syn1 Syn2	1.092	–
	VLL	PEPC PK Syn1	1.250	–
	EAK	PEPC DHAPS PGM PPK	**1.387**	1.380 ± 0.00449
4	VLS	DHAPS G1PAT PK PGI	1.124	–
	VLL	PEPC PK G1PAT Syn1	1.273	–
	EAK	PEPC DHAPS MURS PGM PPK	**1.389**	1.386 ± 0.00242
5	VLS	DHAPS G1PAT PK G3PDH PGI	1.124	–
	VLL	PEPC PK Syn1 PPK TRPS	1.262	–
	EAK	PEPC DHAPS Syn1 PK PPK PGI	**1.394**	1.387 ± 0.00280
6	VLS	DHAPS G1PAT PK G3PDH PGI METS	1.124	–
	VLL	PEPC PK Syn1 PPK TRPS METS	1.262	–

- *Number of selected individuals for reproduction*: 50 individuals;
- *Number of reinserted individuals in the population*: 49 individuals;
- *Selection operator*: Tournament selection with three individuals randomly selected, where the fittest is selected.

5.2.7 Implementation

Regarding the implementation, the software for the proposed tasks was developed using the Java, Scala, and Matlab languages. The following libraries were utilized: JECoLi, a library for EAs developed by the authors [25] and JSBML [26] a java library allowing to parse SBML encoded files. Differential equations were simulated using the solver ODE15s from Matlab. The source code is released under the GPLv3 license and is available from `http://darwin.di.uminho.pt/Software/EADynamic`.

5.3 Results and Discussion

The best solutions obtained with the proposed EA are displayed in Tables 5.1 and 5.2. These solutions are contrasted to the ones found in the literature [19], namely the ones

Table 5.2: Enzyme Modulation task - best solutions

#Modifications	Algorithm	Modifications	Fitness (v_j/v^0_{j0})	Mean Fitness ±95% Confidence Interval
1	EAE	(2.0)Sersynth	**1.876**	1.802 ± 0.0761
	VLS	(2.0)Sersynth	**1.876**	–
	VLL	(2.0)Sersynth	**1.876**	–
2	EAE	(1.99)Sersynth (0.0033)PGluMu	**2.413**	2.189 ± 0.0119
	VLS	(2.0)Sersynth (0)PK	2.115	–
	VLL	(2.0)Sersynth (2.0)PTS	1.876	–
3	EAE	(1.99)Sersynth (1.92)GAPDH (0.0032)PGluMu	**2.582**	2.385 ± 0.0251
	VLS	(2.0)Sersynth (1.94)GAPDH (1.57)PFK	2.001	–
	VLL	(2.0)Sersynth (0)PEPC (1.84)PTS	2.191	–
4	EAE	(1.99)Sersynth (0.043)TKA (0.0032)PGluMu	**2.639**	2.475 ± 0.0222
	VLL	(2.0)Sersynth (2.0)PTS (0)PEPC (2.0)GAPDH	2.369	–
5	EAE	(1.99)Sersynth (0.015)R5PI (0.015)PEPC (1.99)GAPDH (0.015)PK	**2.661**	2.529 ± 0.0254
	VLL	(2.0)Sersynth (1.94)PTS (0)PEPC (2.0)GAPDH (0)PK	2.532	–
6	EAE	(1.99)Sersynth (0.0035)PEPC (1.86)GAPDH (0.0035)PGluMu (0.0035)R5PI (1.79)TRPS	**2.705**	2.553 ± 0.0251
	VLL	(2.0)Sersynth (1.38)PTS (0)PEPC (1.90)GAPDH (0)PK (0)DHAPS	2.671	–

resulting from a linearized approximation of the non-linear model of central carbon metabolism of *Escherichia coli* around a steady-state. These solutions are also constrained by flux and concentration bounds to reduce the likelihood of not portraying the behavior of the original model. All the fitness values (v_j/v^0_{j0}) concern the non-linearized version of the model.

In both tables, EAK and EAE represent the data regarding the solutions found with devised EA (for knockout and enzyme level optimization, respectively), while VLS and VLL are related to the application of the method developed in [19]. In VLS, the enzyme expression levels (e^0_i) at steady state in the linearized model are restricted by the inequality $0.5e^0_i \leq e^0_i \leq 2e^0_i$, while metabolite concentrations at steady-state (x^0_i) are constrained by $0.5x^0_i \leq x^0_i \leq 1.5x^0_i$. In VLL, the enzyme modulation levels in the linearized model are constrained by $0.5e^0_i \leq e^0_i \leq 2e^0_i$ and the metabolite concentrations by $0.5x^0_i \leq x^0_i \leq 10x^0_i$.

The developed EA overcomes these restrictions by integrating the non-linear model and by assuming that the model depicts adequately the subjacent reality. Nonetheless, it is important to note that even the original model may not be valid in all range of metabolite concentrations due to the absence of data regarding those states when the

model was fitted. Comparing the results obtained by the proposed EA with the ones in [19], it is possible to check that they show equal or, in most cases, higher fitness values.

In all the studied scenarios the first proposed transformation is also part of the underlying proposed modifications. The first modification is the one that implies a larger flux gain in the Serine synthesis flux. However, reaction knockouts are geared towards reactions leading to an increase in the concentration of compounds that contribute to Serine formation, whereas in the enzyme modulation, the algorithm tends to maximize the flux that leads directly to the production of Serine (in this case the serineSynth reaction). Notwithstanding, as the number of reaction modification increases, the marginal serine synthesis flux gain usually tends to decrease. This fact can be observed in the present case studies as well as in [19].

In addition, even without the constraints that restrict most the variation of fluxes and concentrations, solutions tend to knockout reactions directly related with drains, as it can be observed in Table 5.1. In VLS and VLL knockout solutions, the first two modifications will normally imply reactions that drain compounds out of the network. These reactions are selected because they imply a smaller change in the overall metabolite concentrations, with an increase of the Serine production. Contrarily to what would be empirically expected, the EA only deletes reactions not directly related with drain reactions (PK, PGI) with 6 modifications. This fact is owed to the absence of constraints in the EA algorithm regarding flux and metabolite concentration constraints that may limit what are considered valid solutions.

In all knockout algorithms, as the number of allowed modifications increases, some of the previously utilized reactions are swapped by a not apparent set of reactions that cause an increase in the target flux. This fact can be observed in table 5.1. For instance, with five knockouts, the best solution found by the EA is composed by the PEPC, DHAPS, MURS, PGM and PPK reactions, while with six modifications the PGM and MURS reactions are changed by Syn1, PGI and PK. This swap of reaction produces an approximated 0.35% increase in the target flux.

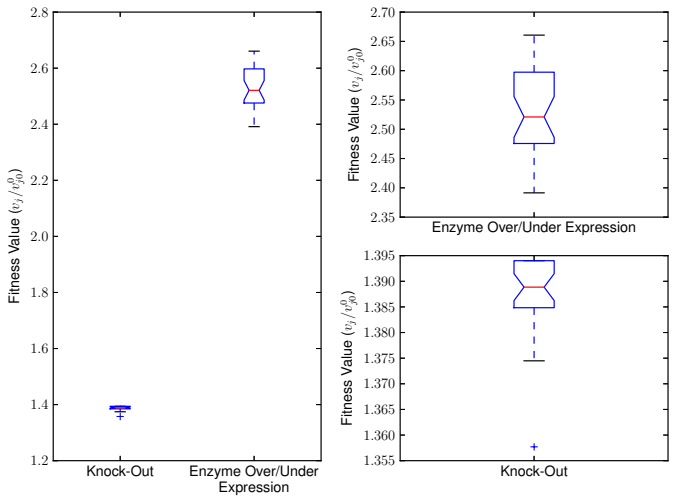


Figure 5.3: Boxplots concerning the best solutions with six modifications found in the knock-out and enzyme modulation tasks, regarding the Serine maximization case study.

In the devised EA knockout algorithm up until five modifications, the reactions of the solution tend to converge to the same solution in all runs. With three modifications there is a twelve-fold change in variability regarding the previous cases. This increase in variability can be explained by the increase in the search space.

In Figure 5.3, it is possible to observe that the knock-out algorithm tends to converge to a set of solutions with lower variability than the enzyme over/under expression counter part. This fact may be explained by the larger search space of the enzyme modulation task and the number of solutions with similar values. It is also noticeable that the enzyme over/under expression task requires more iterations to reduce the variability in the best solutions. These results cannot be extrapolated to other models or scenarios and depend on the objective function, constraints and the underlying metabolic model.

5.4 Conclusion

This work encompassed the development of algorithms to design *in silico* improved microbial strains for the production of industrial relevant compounds. These algorithms

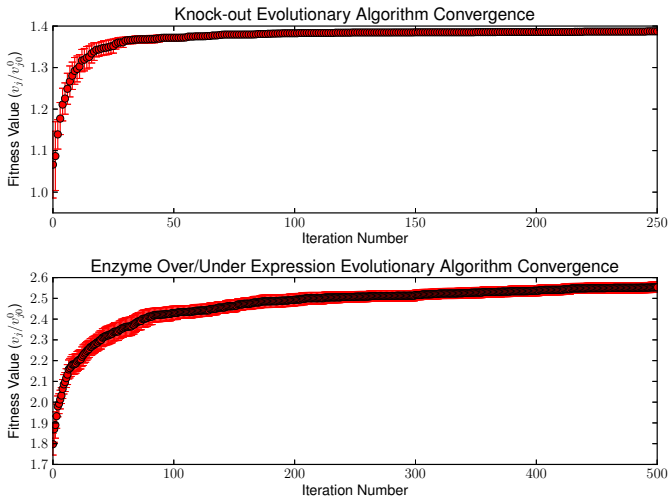


Figure 5.4: Knock-out and enzyme modulation evolutionary algorithms convergence with 95% confidence bounds, concerning the best solutions found with six modifications during the 30 runs of the algorithms in the Serine maximization case study.

achieved these modifications by finding the near/best set of reaction deletions to remove from a model and/or to infer the optimum expression levels for the enzymes in the model (or a predefined subset). A dynamic ordinary differential model describing the central carbon metabolism of *Escherichia coli* was used as basis for the simulation of the devised ME strategies. These models are capable of describing regulatory effects in the metabolism not possible to represent with steady-state models.

The best solutions returned by the devised method outperformed the ones in [19] due to the fact that no approximations of the model were needed, except in the Enzyme modulation task with six modifications. Solutions were computed allowing the metabolite concentrations and the fluxes to vary with no bound restrictions. During the execution of the algorithms a set of reaction modifications that might lead to an invalid steady-state were not immediately discarded. Thus, a subset of these reactions could serve as building blocks for better and valid solutions.

In future work, the remaining issues to be tackled are the validation of the work with other real-world case studies and also the integration of the developed software in a

user-friendly software platform such as Optflux [27]. The utilization of multi-objective optimization algorithms [28] is also an expected extension to the current methods.

REFERENCES

[1] H. Kitano, "Systems biology: a brief overview," *Science*, vol. 295, no. 5560, p. 1662, 2002.

[2] G. Stephanopoulos, A. Aristidou, J. Nielsen, and J. Nielsen, *Metabolic engineering: principles and methodologies*. Academic Pr, 1998.

[3] J. Reed, T. Vo, C. Schilling, and B. Palsson, "An expanded genome-scale model of escherichia coli k-12 (ijr904 gsm/gpr)," *Genome Biol*, vol. 4, no. 9, p. R54, 2003.

[4] J. Forster, I. Famili, P. Fu, B. Palsson, and J. Nielsen, "Genome-scale reconstruction of the saccharomyces cerevisiae metabolic network," *Genome research*, vol. 13, no. 2, p. 244, 2003.

[5] K. Kauffman, P. Prakash, and J. Edwards, "Advances in flux balance analysis," *Current opinion in biotechnology*, vol. 14, no. 5, pp. 491–496, 2003.

[6] D. Segre, D. Vitkup, and G. Church, "Analysis of optimality in natural and perturbed metabolic networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 23, p. 15112, 2002.

[7] T. Shlomi, O. Berkman, and E. Ruppin, "Regulatory on/off minimization of metabolic flux changes after genetic perturbations," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 21, p. 7695, 2005.

[8] J. Reed and B. Palsson, "Thirteen years of building constraint-based in silico models of escherichia coli," *Journal of Bacteriology*, vol. 185, no. 9, p. 2692, 2003.

[9] R. Schuetz, L. Kuepfer, and U. Sauer, "Systematic evaluation of objective functions for predicting intracellular fluxes in escherichia coli," *Molecular systems biology*, vol. 3, no. 1, 2007.

[10] K. Patil, I. Rocha, J. F
"orster, and J. Nielsen, "Evolutionary programming as a platform for in silico metabolic engineering," *BMC bioinformatics*, vol. 6, no. 1, p. 308, 2005.

[11] A. Burgard, P. Pharkya, and C. Maranas, "Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization," *Biotechnology and bioengineering*, vol. 84, no. 6, pp. 647–657, 2003.

[12] B. Papp and E. Simeonidis, "Flux balance analysis and its applications," *BMC Systems Biology*, vol. 1, no. Suppl 1, p. P77, 2007.

[13] M. Rocha, P. Maia, R. Mendes, J. Pinto, E. Ferreira, J. Nielsen, K. Patil, and I. Rocha, "Natural computation meta-heuristics for the in silico optimization of microbial strains," *BMC bioinformatics*, vol. 9, no. 1, p. 499, 2008.

[14] P. Pharkya and C. Maranas, "An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems," *Metabolic engineering*, vol. 8, no. 1, pp. 1–13, 2006.

[15] E. Gonçalves, R. Pereira, I. Rocha, and M. Rocha, "Optimization approaches for the in silico discovery of optimal targets for gene over/underexpression," *Journal of Computational Biology*, vol. 19, no. 2, pp. 102–114, 2012.

[16] F. Wang, C. Ko, and E. Voit, "Kinetic modeling using s-systems and lin-log approaches," *Biochemical engineering journal*, vol. 33, no. 3, pp. 238–247, 2007.

[17] J. Dean and G. Dervakos, "Design of process-compatible biological agents," *Computers & chemical engineering*, vol. 20, pp. S67–S72, 1996.

[18] J. Dean and G. Dervakos, "Redesigning metabolic networks using mathematical programming," *Biotechnology and bioengineering*, vol. 58, no. 2-3, pp. 267–271, 2000.

[19] F. Vital-Lopez, A. Armaou, E. Nikolaev, and C. Maranas, "A computational procedure for optimal engineering interventions using kinetic models of metabolism," *Biotechnology progress*, vol. 22, no. 6, pp. 1507–1517, 2006.

[20] C. Chassagnole, N. Noisommit-Rizzi, J. Schmid, K. Mauch, and M. Reuss, "Dynamic modeling of the central carbon," *Biotechnol Bioeng*, vol. 79, pp. 53–73, 2002.

[21] E. Nikolaev, "The elucidation of metabolic pathways and their improvements using stable optimization of large-scale kinetic models of cellular systems," *Metabolic engineering*, vol. 12, no. 1, pp. 26–38, 2010.

[22] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, p. 671, 1983.

[23] M. Sadovnikova and V. M. Belikov, "Industrial applications of amino-acids," *Russian Chemical Reviews*, vol. 47, no. 2, p. 199, 2007.

[24] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, *et al.*, "Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems," *Nucleic acids research*, vol. 34, no. suppl 1, pp. D689–D691, 2006.

[25] P. Evangelista, P. Maia, and M. Rocha, "Implementing metaheuristic optimization algorithms with jecoli," in *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pp. 505–510, IEEE, 2009.

[26] A. Dräger, N. Rodriguez, M. Dumousseau, A. Dörr, C. Wrzodek, N. Le Novère, A. Zell, and M. Hucka, "Jsbml: a flexible java library for working with sbml," *Bioinformatics*, vol. 27, no. 15, pp. 2167–2168, 2011.

[27] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. P. Pinto, J. Nielsen, K. R. Patil, E. C. Ferreira, and M. Rocha, "Optflux: an open-source software platform for in silico metabolic engineering," *BMC systems biology*, vol. 4, no. 1, p. 45, 2010.

[28] P. Maia, I. Rocha, E. C. Ferreira, and M. Rocha, "Evaluating evolutionary multiobjective algorithms for the in silico optimization of mutant strains," in *BioInformatics and BioEngineering, 2008. BIBE 2008. 8th IEEE International Conference on*, pp. 1–6, IEEE, 2008.

CHAPTER 6

SOFTWARE FOR BIOCHEMICAL MODEL DESIGN AND OPTIMIZATION

The work developed during the different tasks of this thesis required the construction of customized software tools, not available as a single software package. In this chapter, the overall functionalities of the constructed software are described, being provided an overview of the available methods. The developed software contemplates a model specification language, along with other domain specific languages (DSL) to generate and transform models based on Mass Action Rate Laws (MARLs) and aggregated rate laws (ARLs). The implemented methods are also seamlessly integrated by a DSL, that allows the exchange of information between distinct algorithms, utilizing files as the communication medium, thus permitting a flexible approach in a cluster computational setting.

6.1 Software Overview

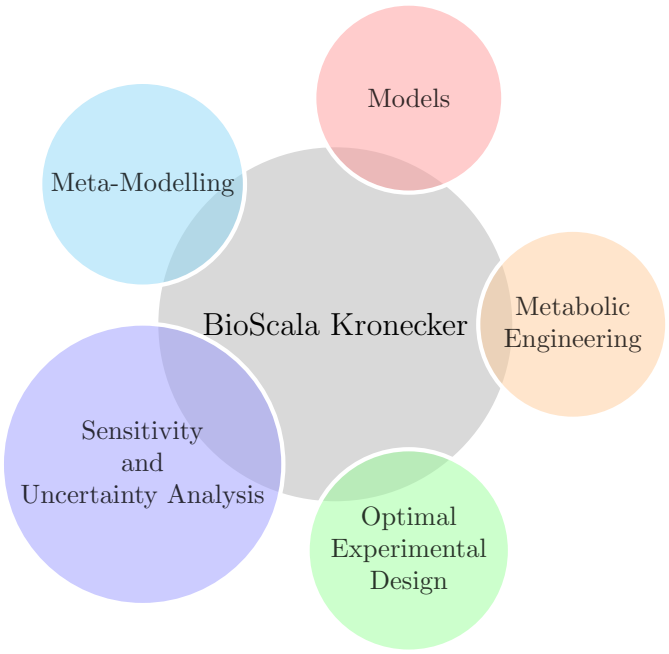


Figure 6.1: Main areas covered by the BioScala Kronecker Software.

The developed software can be represented by the layered structure shown in Figure 6.1. The name of the software is a direct reference to the BioKronecker Matlab package, created and developed at MIT in Bruce Tidor's Group by Joshua Apgar and David Hagen. The Kronecker formalism was reimplemented in the Scala programming language, based on the implementation performed by Joshua Apgar in [1] and David Hagen in [2].

This work provided the extension of the software, to incorporate symbolic expression derivatives, simplification, as well as the creation of more versatile structured representations of the underlying concepts in a software engineering perspective, due to the higher expressiveness level of the Scala programming language, when contrasted with Matlab.

This implementation also takes advantage of Matlab toolkits (namely the optimization package), by utilizing the JA builder package from [3], to integrate external prim-

itives. Numerical codes concerning simulation and sensitivity analysis were implemented in Matlab, due to the matrix integration of sparse and dense codes, as well as the ode15s solver. The Jecoli library [4] was utilized, when evolutionary algorithms were required. It is important to bear in mind that the central layer is connected to all the remaining layers, while the remaining layers are often independent from each other.

The core layer encompasses six distinct levels:

- Models - Encapsulates all the information concerning biochemical model representations. The developed software has the capability of representing the following types of models:
  - Stoichiometric models, characterized by the biochemical network structure;
  - Dynamic ordinary differential equation (ODE) based systems. The underlying representation is based on the Kronecker formalism, extended to accommodate free form symbolic expressions;
  - A special representation was also developed to represent S-system based models.

  This layer also encapsulates the following functionalities:
  - Reaction Generator - The software package encompasses two domain specific languages, described more thoroughly in the following sections, to simplify the generation of mechanistic descriptions of reactions based in elementary reactions, further into Modular Kinetics.
  - Experiment - Contains the necessary configuration information to perform the available simulation and optimization methods;
  - Calibration - Allows the regression of model parameters utilizing frequentist and Bayesian techniques;
- Meta-modeling - Accommodates formalisms that map the input-output relation of a system, without taking into account its physical description. In the software, there are three representations: Quasi Random Sampling High Dimensional

Model Reduction (QRS-HDMR) [5], feed foward neural networks [6] and Extreme Learning Machines [7] (see Chapter 2);

- Sensitivity and Uncertainty Analysis - Includes global and local methods to assess the uncertainty associated to model parameters. The Software package includes the Morris method [8] (with several extensions [9], [10]) for parameter screening, the Sobol method [11], HDMR based methods (with extensions [12], [13]), Nested Sampling [14] as alternative to Metropolis-Hastings algorithm [15], and Local sensitivities (see Chapter 2);

- Optimal Experimental Design - Allows to compute the most informative experiments, often using as basis local methods based on the Fisher Information Matrix (FIM).

- Optimization Algorithms - Covers all the interfaces needed to execute an optimization problem in Matlab (such as simulated annealing, fmincon, fminsearch, patternsearch), as well as any Jecoli algorithm based on Evolutionary Computation;

- Metabolic Engineering Layer - Embraces all the devised methods in the previous chapters for modulating enzyme expression levels and reaction knock-outs.

6.2 Software Structure

Due to the large computational demands and amounts of data generated by some of the implemented methods, the distinct procedures communicate using files. This permits their use in a cluster setting. A domain specific language (DSL) was designed, to configure the cooperation between the implemented methods. This language is structured as follows:

- Jobs: A job represents a specific method, and the respective configuration options. Each job returns a floating point value, after finishing up the execution. This value

may serve as a basis to decide which job will be executed next;

- Script: Represents a state-machine (containing a set of jobs), where jobs are the nodes, and the directed arcs between jobs represent the jobs that will be executed next, depending on a user defined boolean condition. Global variables and pre-processor macros that affect the jobs configurations may also be defined.

6.2.1 Job Definition Language

A DSL was created to connect the outputs and inputs of multiple jobs, in a algorithmic fashion, based on user defined logic. Each job has a unique identification, and is divided into at least three mandatory parts:

- Problem Configuration: where the problem specific configurations are specified;
- Job directories: specifies the root directory of the job, as well as the output directory (where outputs produced during the job execution should be saved);
- Conditions: defines what job should be executed based on a boolean condition.

Optionally, the following blocks can also be defined:

- Optimization Algorithm: if the job defines an optimization problem, the user must supply the optimization algorithm and the specific configuration in this block;
- Job Writers: entities responsible to save information about the current job (for instance, to save a model transformation or an algorithm result);
- Optimization Writers: Analogously to Job writers, these objects are responsible for saving specific information about an optimization algorithm execution.

As an example, in this language, a job to calibrate a model to experimental data could be defined as:

```
Job globalCalibrationSA {
Problem ML «
experimentFile:"experimentalData/timeSeriesFitting.ss"
modelFile:"${partialBaseOutputDir}/mainGlobal${reactionId}.model"
variance:0.2
maxTime:3
useModelParameters:false
useLHCS:true
»

OptimizationAlgorithm SANumeric {
   NumberOfIterations:300
        NumberOfFunctionEvaluations:300
        ObjectiveFunctionTolerance:1E-16
}

baseDirectory:"${baseDir}"
   baseOutputDirectory:"${baseOutputDir}"

   SolutionWriter MLModelWriter «
      baseFilename:"main${reactionId}.OptimizationSA.model"
   »

   SolutionWriter MLExperimentWriter «
      baseFilename:"initialDEExperiment"
      modelFile:"initalDEModel"
   »

   AlgorithmWriter StatisticsWriter «
```

```
      baseFilename:"initialDEStatistic"
   »

   Condition «true::globalCalibrationHDE»
}
```

In the above job definition, the previously mentioned mandatory and optional blocks are shown. It is important to note that strings enclosed in ${ } are global variables, and can take integer, floating point and string values (the language is dynamic and has strong typing). These variables are defined in the job script file, before any job definitions. A variable may refer to another variable, if it has already been defined. For example, the global variables in a script file may be defined as:

```
reactionId="DAHPS"
baseDir = "/home/jobs/model"
partialBaseOutputDir = "Result/Calibration/${reactionId}"
baseOutputDir = "${baseDir}/${partialBaseOutputDir}"
```

These variables can also be modified in the Condition block of a job definition. This block has tree parts: (i) a boolean condition, if true executes the code block (explained next);(ii) the code block contains assignment instructions, that modify the variable values;(iii) the identification of the next job to be executed. In the above Job definition, there is only one boolean condition, without any code block, and a jump to the globalCalibrationHDE job. An example of a more elaborate condition block is given below:

```
Condition «cCounter < cCounterLimit:
                cCounter = cCounter + 1;
                cLevel = cCounter*cLevelStep
                :lhsKARegressionCorrelationModelGeneration
              true::correlationFixModelParameters»
```

In this scenario, the condition block has two possible paths (a job can define any number of possible execution paths):(i) if the cCounter variable is less than the cCounterLimit, then cCounter will be incremented by one unit, while cLevel is assigned the value of the expression $cCounter * cLevelStep$, and after the execution jumps to job lhsKARegressionCorrelationModelGeneration;(ii) if the previous boolean condition fails, the next job to be executed will be correlationFixModelParameters. In these condition blocks, value is a reserved keyword that contains a value returned by the Job. These values serve as termination codes that may influence the job execution flow.

6.2.2 Available Jobs

The following list, represents the main jobs available, on the developed software package:

- Simulation: Permits the simulation of a defined system during a pre-specified set of time points;
- King-Altman formula generation: Taking an elementary reaction description of a system, it generates the corresponding King-Altman formula (check Chapter 2);
- King-Altman calibration: The model parameters are calibrated using a specified optimization algorithm, the original ARL formula, the corresponding King-Altman (KA) formula and the grid of points specified by the user. This method does not use integration (consult Chapter 2);
- King-Altman calibration simulation: When the elementary reaction cannot be converted to the corresponding King-Altman formula due to its dimension, the calibration is done against the simulation of the ARL original system. In this job, the user supplies as inputs the original ARL system, the elementary reaction system description and the time points for each simulation, as well as the set of initial conditions (check Chapter 2);
- CHA ARL generation (consult Chapter 2)

- CHA calibration: Analogous to King-Altman calibration but employing the CHA method instead (check Chapter 2);
- Calibration of model parameters against experimental data: A system is calibrated against experimental data utilizing Maximum Likelihood estimation (assuming that all model parameters possess a normal distribution). The program supports metabolite and flux data. As in the KA calibration scenario, the user defines the utilized optimization algorithm and options (specific code was implemented allowing the use of derivatives $\frac{dx}{d\theta}$ even for systems with hundreds of parameters);
- Functional PCA of simulations (check Chapter 2);
- Optimal Design of Experiments based on Fisher Information Matrix (check Chapter 2);
- Morris method with extensions - Morris method with extensions proposed in [16] and [10]. The extensions are related to the use of scaled elementary effects by the ratio of the output standard deviation by the corresponding parameter standard deviation, functioning as an approximation of sigma normalized derivatives. This method allows the ranking of model parameters utilizing low computational time (when compared to Sobol and HDMR based methods) and works as a proxy for the Total sensitivity indexes;
- Latin Hyper Cube Sampling;
- Sobol Sequence Sampling;
- Monte Carlo Filter (Regionalized Sensitivity Analysis)
- Sobol method GSA [11];
- Random Sampling High Dimension Model Representation for GSA [17];

- Neural Networks/Extreme Machine Learning Machines for representing rate equations;
- Extended DFBA (able to deal with the absence of substrate in the bioreactor) - Chapter 4;
- Extended DFBA Calibration (check Chapter 4);
- Extended DFBA Feeding signal design (check Chapter 4);
- Metabolic Engineerig (Enzyme Modulation/Reaction Knockouts) utilizing the techniques of Chapter 5.

6.3 Model Descritption Language

The developed models during this work are specified in a specially designed language, accommodating the needs for the simulation and the optimization methods. The model specification file is divided into the following segments:

- Functions: where arbitrary mathematical expressions, such as rate laws are specified;
- Parameters: where named mathematical expressions are presented. The value of the parameters may be updated at the ODE solver time step if a mathematical function that utilizes a species in the model is utilized. The parameters also permit the optimization of the function argument values. Thus, it is possible to optimize any mathematical expression. The parameters also support the instantiation of Neural networks or Extreme Learning Machines if the function call NN(networkConfigurationFile) is supplied ;
- Compartments: This section contains the dimensionality of the compartment, the set of species and inputs, and the recursive inner compartments definition. The inputs are defined as the parameters, with the exception of an equal sign between

the identifier and the expression, as well as the termination of the line with the character ';';

- Reactions: Describes the reaction schemes in terms of the elementary steps or ARL. Each reaction is characterized by a macro reaction, and the respective decomposition steps.

For instance, consider a simple model containing a MARL reaction for a Michaelis-Menten type of reaction:

$$E + S \underset{k_1}{\overset{k_0}{\rightleftarrows}} ES \overset{k_2}{\rightarrow} E + P \tag{6.1}$$

where E is the enzyme, S the substrate, P the product and the k_i represent the elementary rates. This system could be described in the developed format as:

```
[Functions]

def k1Fun(a,b):
     a*b
end

[GlobalParameters]

rK1 = k1Fun(a = 2,b = 1) (a: (*3),b: [-5,1])

[Compartments]

compartment outside {
    dimension = volume
    value = 1
```

```
    species {
    }

    inputs {nonUsedInput = 1;}

    compartment cell {
    dimension = volume
    value = 1
    species {
           S 0.5
           P 0.5
        }

        inputs {}
    }
}

[Reactions]


vE: S  --- P {
    modifiers {}

    enzymes {
    cell {
       E 1  (*1E-3)
       ES 0 [-5,-1]
       }
    }
```

```
    S + E <-> ES >@10 <@rK1 (*1E6,) [0,6]---[0.8]
    S -> E + P @ 3
}
```

In the first block [Functions], mathematical functions are defined with the keyword **def**, followed by a list of comma separated parameters in parentheses, terminated with the character ':'. Afterwards, any valid mathematical expression can be introduced. A function definition is closed with the keyword **end**. An unlimited number of function definitions can be introduced.

The block [GlobalParameters] contains parameter definitions. In the example above, the parameter rK1 represents a function call to function k1Fun, with arguments a equal to 1 and b equal to 1. The tuple (a: (*3),b: [-5,1]) states that the parameter a, when participating in an optimization, should have its value fixed to 3, while parameter b is allowed to vary between the bounds -5 and 1 in log scale. In the function call, the arguments are passed by value, and unnamed arguments are also supported.

The block [Compartments], specifies the system compartmentalization. Each compartment has a dimension, that can take any of the strings: volume (corresponds to 3D), area (corresponds to 2D) and point (corresponds to 1D). A compartment may only be contained inside a higher or equal dimension compartment. The value attribute defines the value of the corresponding dimension. Species concentrations are rescaled when reactions interact between two compartment with distinct dimensions. The compartment blocks are defined recursively, being the inner compartments defined inside the outer compartment. Each compartment specifies the species and the inputs, common to all the reactions. The input with name nonUsedInput, whose value is one, was cast as an example.

Species are defined by an identifier, and an initial concentration value. Optionally, bounds and optimization values can be defined, analogously to what was done in the function parameter block.

Afterwards, the [Reaction] block is delineated. Each reaction in this block is characterized by a macro reaction, with an identity, reactants, products and the micro level reaction description. The macro description is given by:

```
vE: S  --- P{...}
```

while the micro description is defined as:

```
modifiers {}

enzymes {
   cell {
     E 1  (*1E-3)
     ES 0 [-5,-1]
   }
}

S + E <-> ES >@10 <@rK1 (*1E6,) [0,6]---[0.8]
ES -> E + P @ 3
```

where modifiers are a set of species, separated by space, or a line feed, and represent the effectors that interact with the reaction, (while their concentrations are not modified by it), enzymes blocks define a set of species only related to the reaction and the respective compartment, where they occur. Finally, the elementary reactions are specified.

When a definition characterizes a forward and backward reaction , such as:

```
S + E <-> ES >@10 <@rK1 (*1E6,) [0,6]---[0.8]
```

the forward rate is defined after the symbol >@ (in this example, has the value 10), analogously the backward rate is defined after the symbol <@ (in this example, this rate

has the value of the parameter rk1). The term (*1E6,), asserts that in an optimization, the forward rate should be fixed to the value 1E6, while the backward rate is free to vary within the predefined bounds. The bound [0,6] defines the forward reaction bounds in an optimization problem, and similarly the bound [0,8] describes the bounds of the backward reaction. All the bounds are in log-scale. If a reaction has no bounds defined, the default value of [0,10], will be assumed.

The function and parameters follow the same naming conventions, regarding parameter and function names, as the Java language specification. It is important to bear in mind that all the entities of the model must possess a distinct name. It is also important to note that the devised format supports comments similarly to Java code. Thus, strings in between '/*' and '*/', or after '//', are ignored.

The same reaction scheme represented by an ARL could be described by:

```
[Functions]
def mmFun(vmax,km):
    (vmax*S)/(km + S)
end

[GlobalParameters]

rMM = mmFun(vmax = 2,km = 1)

[Compartments]

compartment outside {
    dimension = volume
    value = 1

    species {
```

```
    }

    inputs {}

    compartment cell {
        dimension = volume
        value = 1
        species {
         S 0.5
         P 0.5
        }

        inputs {}
     }
}

[Reactions]


vEMM: S  --- P {
    modifiers {}

    enzymes {
       cell {
       }
    }

   S -> @rMM
   -> P @rMM
```

```
}
```

6.4 Mechanistic Elementary Reaction Generator Language

A specific domain language was developed to help to describe in a compact way complicated elementary reaction mechanisms (often containing thousands of elementary reactions). This compact description permits to describe any uni-molecular and bimolecular elementary reaction system, using a small fraction of the total number of elementary reactions in the original reaction mechanism.

The devised language divides the reaction scheme into blocks or sets of elementary reactions (uni or bimolecular reactions). Each block characterizes a specific aspect of a general reaction mechanism, and often the participating species have a unitary stoichiometry. These blocks serve as generators of more complex schemes by eliciting which species are formed and up to which stoichiometry. Constraints can be enforced to limit which reactions that are generated. For example, below is a compressed reaction scheme or block for a reaction:

```
[("pep","pep",4),("atp","atp",4)](true){

pep + E <-> E_pep
atp + E <-> E_atp
}
```

The list [("pep","pep",4)] contains a set of tuples. The first position of each tuple represents the species in the block reaction scheme; the second position is the new name given to the species after the generation process; the third position is the maximum stoichiometry attained by that specific compound. All the possible combinations of species are generated and applied to each reaction in the scheme. After, in between

parenthesis are the conditions that may reduce the valid space of elementary reactions. For instance, the reactions $pep + E <-> E_pep$ applying the aforementioned reaction scheme, would generate the following set of elementary reactions:

```
pep0 + E -> E_pep1
pep1 + E_pep1 -> E_pep2
pep2 + E_pep2 -> E_pep3
pep3 + E_pep3 -> E_pep4

pep0 + E <- E_pep1
pep1 + E_pep1 <- E_pep2
pep2 + E_pep2 <- E_pep3
pep3 + E_pep3 <- E_pep4
```

If the enzyme had only four binding sites, for both substrates (pep and atp), the reaction scheme would be described by:

```
[("pep","pep",4),("atp","atp",4)](pep + atp < 5){
pep + E <-> E_pep
atp + E <-> E_atp
atp + E_pep <-> E_pep_atp
pep + E_atp <-> E_pep_atp
}
```

In this scenario, the condition states that the number of bound molecules of pep and atp has to be less than five. In this case, the reaction $pep + E_atp <-> E_pep_atp$ would produce the following output:

```
pep + E_pep0_atp0 -> E_pep1_atp0
pep + E_pep0_atp1 -> E_pep1_atp1
pep + E_pep0_atp2 -> E_pep1_atp2
```

```
pep + E_pep0_atp3 -> E_pep1_atp3
...
pep + E_pep1_atp0 -> E_pep_atp0
...
pep + E_pep2_atp0 -> E_pep3_atp1
...
pep + E_pep3_atp0 -> E_pep4_atp0
```

6.5 JECoLi Update

The JECoLi library was also updated with new optimization algorithms, such as Hybrid Differential Evolution [18], and the capability to execute Matlab codes (utilizing JA Builder coupled with scala code).

6.6 Availability

The described software was developed in Matlab [3], Java [19] and Scala [20] programming languages, and is available as an open source package, without including the proprietary version of the JA builder Matlab package from Mathworks. Some parts were also written in Python, which will be converted to Scala in the future.

REFERENCES

[1] J. F. Apgar, *Experiment design for systems biology*. PhD thesis, Massachusetts Institute of Technology, 2009.

[2] D. R. Hagen, *Parameter and topology uncertainty for optimal experimental design*. PhD thesis, Massachusetts Institute of Technology, 2014.

[3] MATLAB, *version R2012b*. Natick, Massachusetts: The MathWorks Inc., 2012.

[4] P. Evangelista, J. Pinho, E. Gonçalves, P. Maia, J. L. Sobral, and M. Rocha, "A software platform for evolutionary computation with pluggable parallelism and quality assurance," in *Artificial Intelligence Applications and Innovations*, pp. 45–50, Springer, 2011.

[5] G. Li, S.-W. Wang, and H. Rabitz, "Practical approaches to construct rs-hdmr component functions," *The Journal of Physical Chemistry A*, vol. 106, no. 37, pp. 8721–8733, 2002.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[7] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[8] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[9] F. Campolongo, J. Cariboni, and A. Saltelli, "An effective screening design for sensitivity analysis of large models," *Environmental modelling & software*, vol. 22, no. 10, pp. 1509–1518, 2007.

[10] G. Sin and K. V. Gernaey, "Improving the morris method for sensitivity analysis by scaling the elementary effects," *Computer Aided Chemical Engineering*, vol. 26, pp. 925–930, 2009.

[11] I. M. Sobol, *A primer for the Monte Carlo method*. CRC press, 1994.

[12] T. Ziehn and A. S. Tomlin, "A global sensitivity study of sulfur chemistry in a premixed methane flame model using hdmr," *International Journal of Chemical Kinetics*, vol. 40, no. 11, pp. 742–753, 2008.

[13] T. Ziehn and A. Tomlin, "Global sensitivity analysis of a 3d street canyon model—part i: The development of high dimensional model representations," *Atmospheric Environment*, vol. 42, no. 8, pp. 1857–1873, 2008.

[14] J. Skilling, "Nested sampling," *Bayesian inference and maximum entropy methods in science and engineering*, vol. 735, pp. 395–405, 2004.

[15] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[16] M. Ruano, J. Ribes, A. Seco, and J. Ferrer, "An improved sampling strategy based on trajectory design for application of the morris method to systems with many input factors," *Environmental Modelling & Software*, vol. 37, pp. 103–109, 2012.

[17] G. Li, S.-W. Wang, H. Rabitz, S. Wang, and P. Jaffé, "Global uncertainty assessments by high dimensional model representations (hdmr)," *Chemical Engineering Science*, vol. 57, no. 21, pp. 4445–4460, 2002.

[18] J.-P. Chiou and F.-S. Wang, "Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process," *Computers & Chemical Engineering*, vol. 23, no. 9, pp. 1277–1291, 1999.

[19] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley, *The Java Language Specification*. Pearson Education, 2014.

[20] M. Odersky and al., "An Overview of the Scala Programming Language," Tech. Rep. IC/2004/64, EPFL, Lausanne, Switzerland, 2004.

CHAPTER 7

CONCLUSIONS

In this chapter, a general overview of the work accomplished during this thesis and the respective contributions are presented. Relevant topics are also explored regarding possible future work.

7.1 Thesis Summary

Escherichia coli is one of the most utilized and studied microorganisms in industry, as well as in academia due to the easiness of cultivation and genetic modification. There is also a wealth of scientific information available in literature and on the web in public databases. The systematization of this information in a mathematical formalism concerning mechanistic enzymatic reactions of central carbon metabolism may provide a blueprint to be utilized for the rational metabolic engineering of new strains. The main objective of this work was to construct a mechanistic model of central carbon metabolism, to optimize the production of industrial relevant compounds such as Serine. Each reaction on this system was modeled based on elementary reaction steps of the enzymatic reaction. This description allows capturing elementary reaction rates that are encased in ARL parameters. Not all cellular details could be retained and assumptions had to be made such as proton concentration. Enzyme concentration and enzyme total concentrations are kept constant through the experimentation.

Several methods were utilized to calibrate each MARL individually, to the corresponding ARL of the Chassagnole model [1]:(i) The King-Altman based method described in [2]; (ii) Cha based method developed in this work;(iii) Design of Experiments method devised in this work;(iv) Partial calibration developed in this work (v) Global calibration stategy, based in [3];(vi) Symbolic Derivative formula that is equivalent to the KA method calibration (however, it can be extended in the future to operate with non-steady enzyme concentrations, utilizing the method developed by [4]).

These calibration problems were solved utilizing a hybrid optimization approach, using the Hybrid Differential Evolution followed by the fmincon algorithm from matlab with active-set algorithm. The idea behind this heuristic is based on the the premise that the evolutionary based algorithm will guide the optimization to a good area of the search space, while the gradient based algorithm will reach an optima. Due to the non-linearity in the model reactions, it is not possible to guarantee that the generated parameters lie in a global optima. Nonetheless, several runs of the algorithm reached to similar solutions

concerning the goodness of the fit. These methods were coupled with identifiability analysis to detect non-influential parameters.

Kronecker formalism was extended to incorporate symbolic free form rate equations. One application of this extension is the construction of hybrid models that contemplate MARLs with other types of ARLs.

From a ME standpoint, genetic modifications can be searched through a brute force strategy, when working with discrete modifications (reaction deletions or enzyme expressions at predefined levels). Another way of roaming the search space is by utilizing an evolutionary algorithm. In this work a novel variable size encoding scheme permitting the modulation of discrete or continuous enzyme expression levels as well as reaction deletions were developed. This scheme also allowed the determination of the number of modifications. During its execution solutions that violated constraints were not automatically discarded serving as building blocks for valid solutions. The devised optimization was tested in Chassagnole model [1] with the goal of maximizing the Serine flux. The obtained results were contrasted against existing published data, obtained with other algorithms [5], [6].

To take advantage of existing genome scale models of metabolism and the dynamic descriptions of genome scale models, a novel extension of DFBA was proposed. This development relaxes the need of the cell being viable during all the simulation period (viable in the sense of having a valid flux distribution), and the instantaneous consumption of substrates, supplied by n feeding pumps. These extensions to the original method allow to simulate and optimize feeding profiles. In this work, global sensitivity methods were utilized to assess the identifiability of the parameters as well as their impact on the output trajectories of the model variables. The current method also supports the simultaneous optimization of metabolic engineering strategies (conjugating the bioprocess model simultaneously with the genome scale model).

During this thesis an open-source software framework was built using Java [7] and Scala [8] programming languages interfaced with Matlab [9] (for optimization and simulation of Kronecker models) and JEcoLi [10] for the utilization of evolutionary algo-

rithms. This software includes all computational methods described in this work.

A new text model format that allows a convenient description of kronecker models as well as free form kinetics were conceived. A converter of this format to SBML [11] and kronecker package model files was also implemented. This model may be extended in the future to include other cellular sub-systems that regulate enzymatic expression such as signaling and gene regulatory networks.

7.2 Future Work

In the context of the work developed during this thesis, the author proposes the following follow-through research paths:

- The utilization of global sensitivity metrics as guiding support for optimization algorithms to devise new ME strategies. Steps have been taken in this direction with sampling of Metabolic Control Coefficients, with models with associated parameter uncertainties [12]. However, sensitivity indexes (computed with variance based methods) may also be a suitable alternative;
- Application of Indirect optimization methods [12] to construct ME, having as basis MARL models;
- Construction of hybrid non-parametric models to implicitly incorporate parts of the metabolism with missing information;
- Application of non-steady enzyme concentration expressions, derived by Chou [4] in the developed symbolic Derivative method to calibrate MARL;
- Calibration of a model with multiple data sets from different sources.

REFERENCES

[1] C. Chassagnole, N. Noisommit-Rizzi, J. W. Schmid, K. Mauch, and M. Reuss, "Dynamic modeling of the central carbon metabolism of escherichia coli," *Biotechnology and Bioengineering*, vol. 79, no. 1, pp. 53–73, 2002.

[2] J. F. Apgar, *Experiment design for systems biology*. PhD thesis, Massachusetts Institute of Technology, 2009.

[3] J. Zhao, D. Ridgway, G. Broderick, A. Kovalenko, and M. Ellison, "Extraction of elementary rate constants from global network analysis of e. coli central metabolism," *BMC systems biology*, vol. 2, no. 1, p. 41, 2008.

[4] K.-C. Chou, "Graphic rules in steady and non-steady state enzyme kinetics.," *Journal of Biological Chemistry*, vol. 264, no. 20, pp. 12074–12079, 1989.

[5] F. G. Vital-Lopez, A. Armaou, E. V. Nikolaev, and C. D. Maranas, "A computational procedure for optimal engineering interventions using kinetic models of metabolism," *Biotechnology progress*, vol. 22, no. 6, pp. 1507–1517, 2006.

[6] E. V. Nikolaev, "The elucidation of metabolic pathways and their improvements using stable optimization of large-scale kinetic models of cellular systems," *Metabolic engineering*, vol. 12, no. 1, pp. 26–38, 2010.

[7] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley, *The Java Language Specification*. Pearson Education, 2014.

[8] M. Odersky and al., "An Overview of the Scala Programming Language," Tech. Rep. IC/2004/64, EPFL, Lausanne, Switzerland, 2004.

[9] MATLAB, *version R2012b*. Natick, Massachusetts: The MathWorks Inc., 2012.

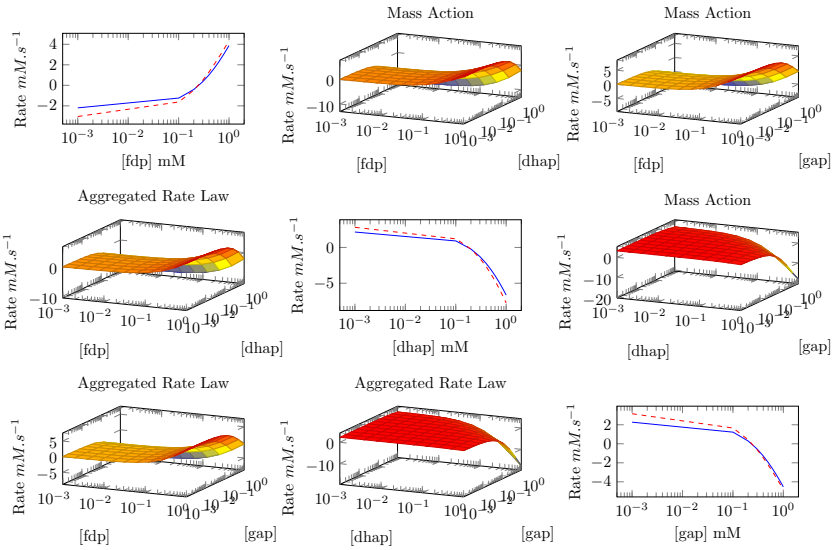
[10] P. Evangelista, J. Pinho, E. Gonçalves, P. Maia, J. L. Sobral, and M. Rocha, "A software platform for evolutionary computation with pluggable parallelism and quality assurance," in *Artificial Intelligence Applications and Innovations*, pp. 45–50, Springer, 2011.

[11] 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, *et al.*, "The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

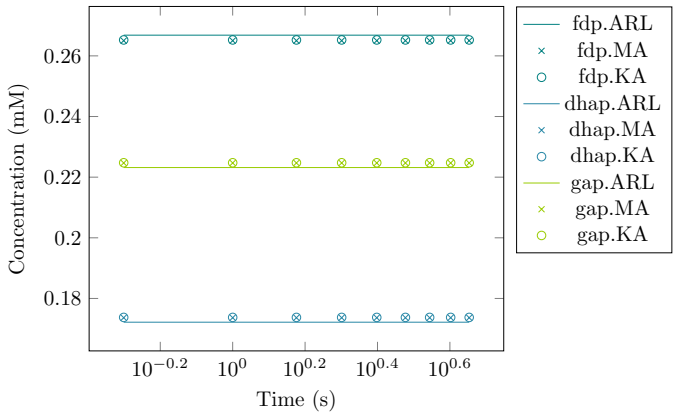
[12] L. Wang, I. Birol, and V. Hatzimanikatis, "Metabolic control analysis under uncertainty: framework development and case studies," *Biophysical Journal*, vol. 87, no. 6, pp. 3750–3763, 2004.

APPENDIX A

MECANISTIC MODEL OF ESCHERICHIA COLI

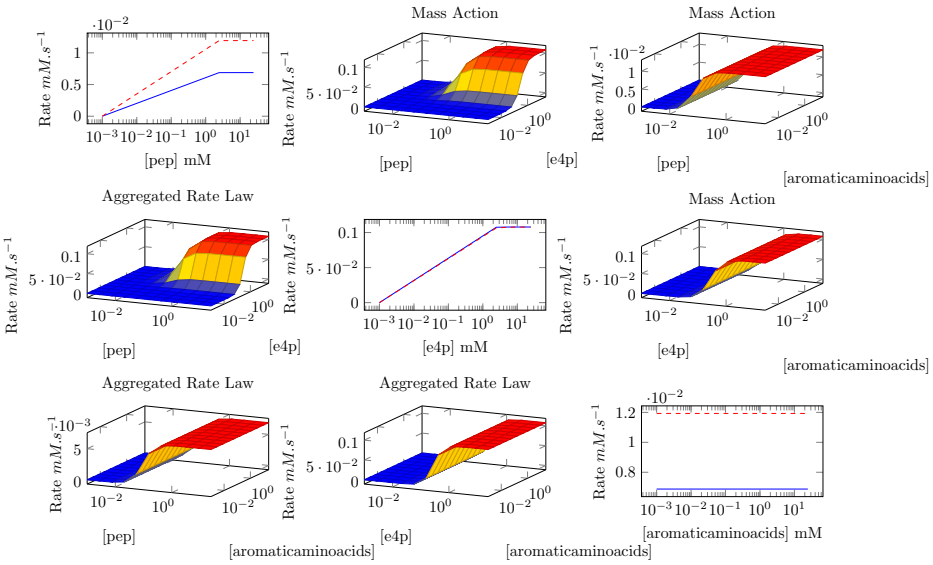


(a) Rate vs Compound graph.

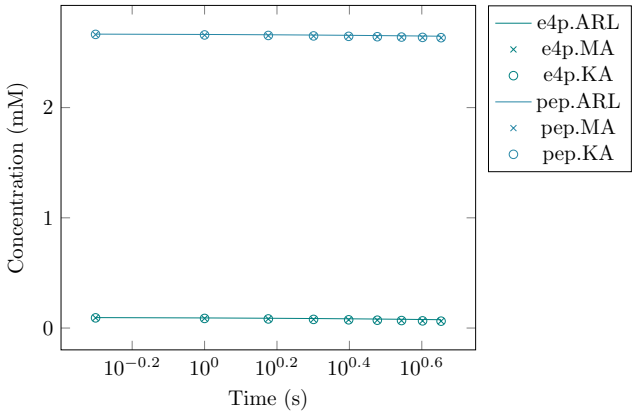


(b) Simulation from system initial conditions.

Figure A.1: Calibration of ALDO Mass action enzymatic system to Aggregated Rate Law Mechanism.

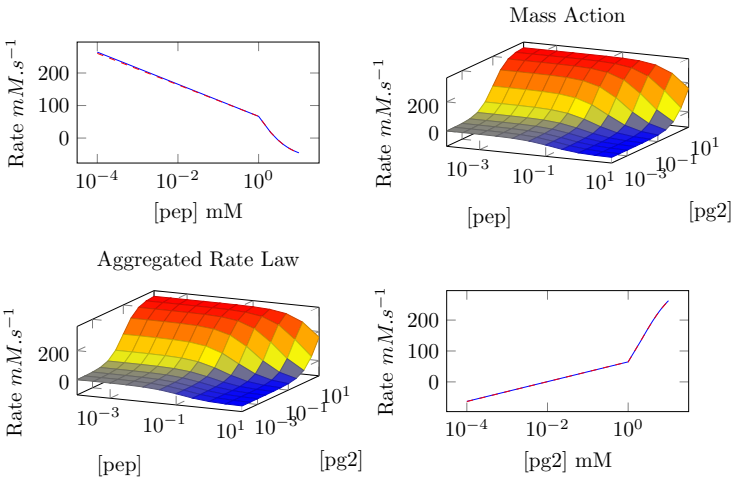


(a) Rate vs Compound graph.

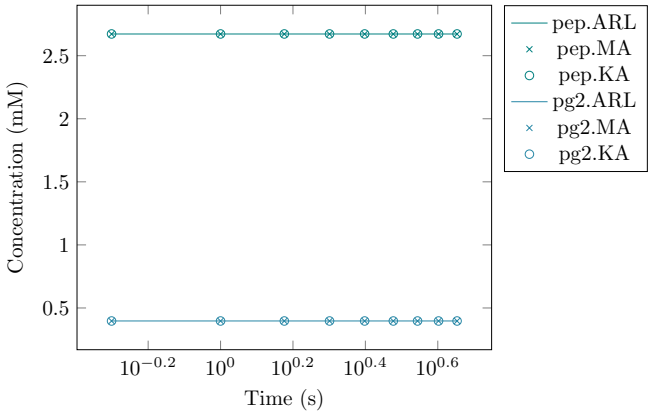


(b) Simulation from system initial conditions.

Figure A.2: Calibration of DHAPS Mass action enzymatic system to Aggregated Rate Law Mechanism.

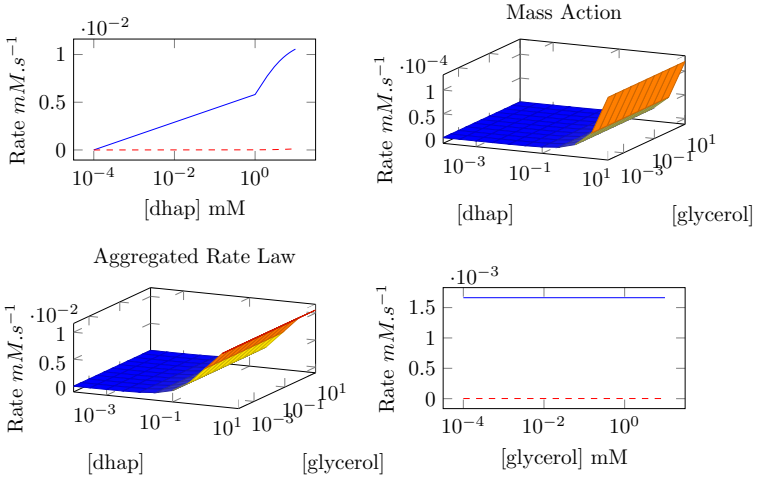


(a) Rate vs Compound graph.

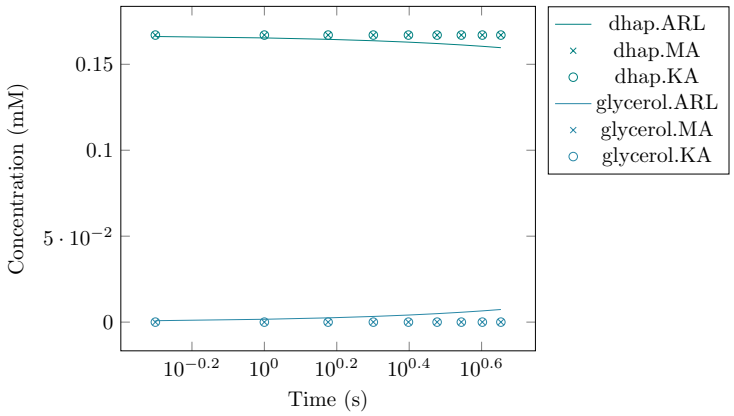


(b) Simulation from system initial conditions.

Figure A.3: Calibration of ENO Mass action enzymatic system to Aggregated Rate Law Mechanism.

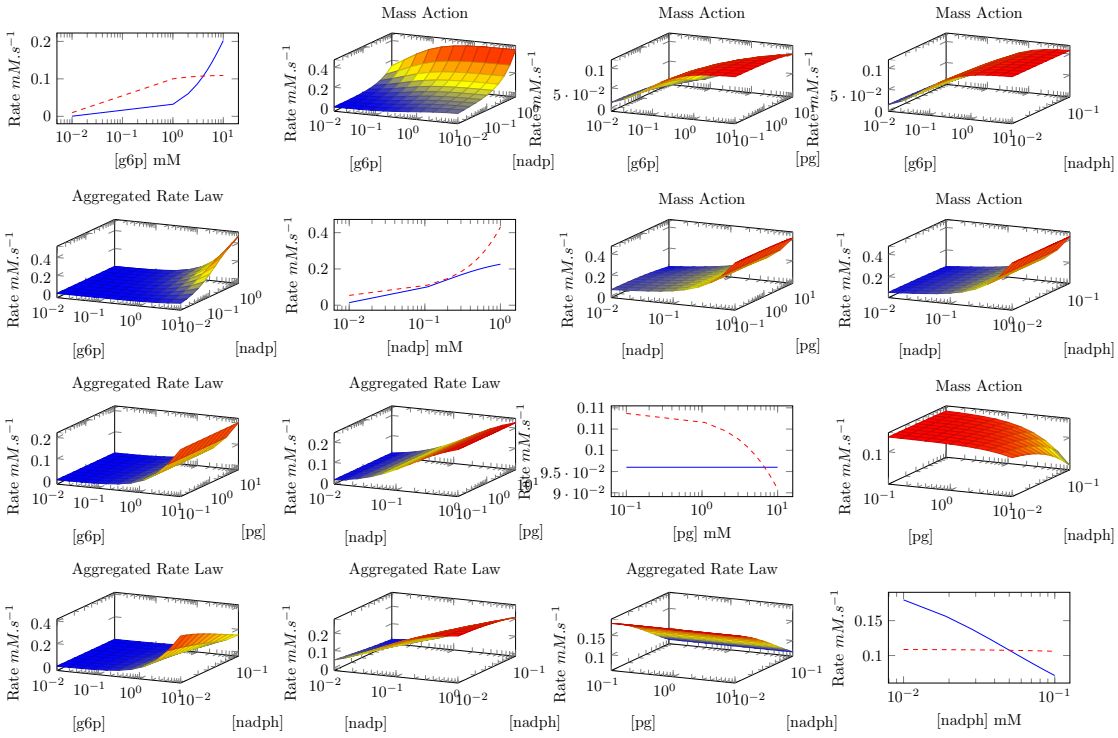


(a) Rate vs Compound graph.

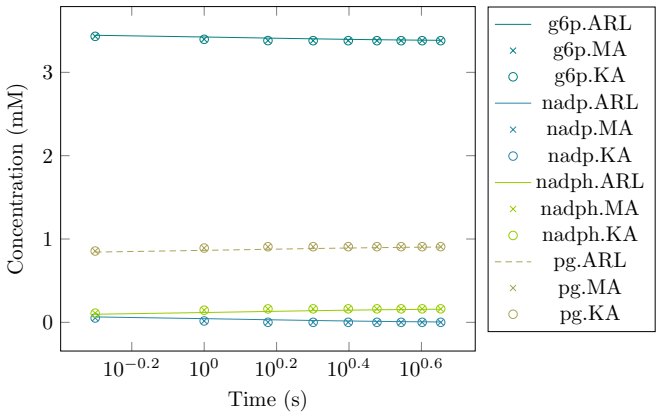


(b) Simulation from system initial conditions.

Figure A.4: Calibration of G3PDH Mass action enzymatic system to Aggregated Rate Law Mechanism.

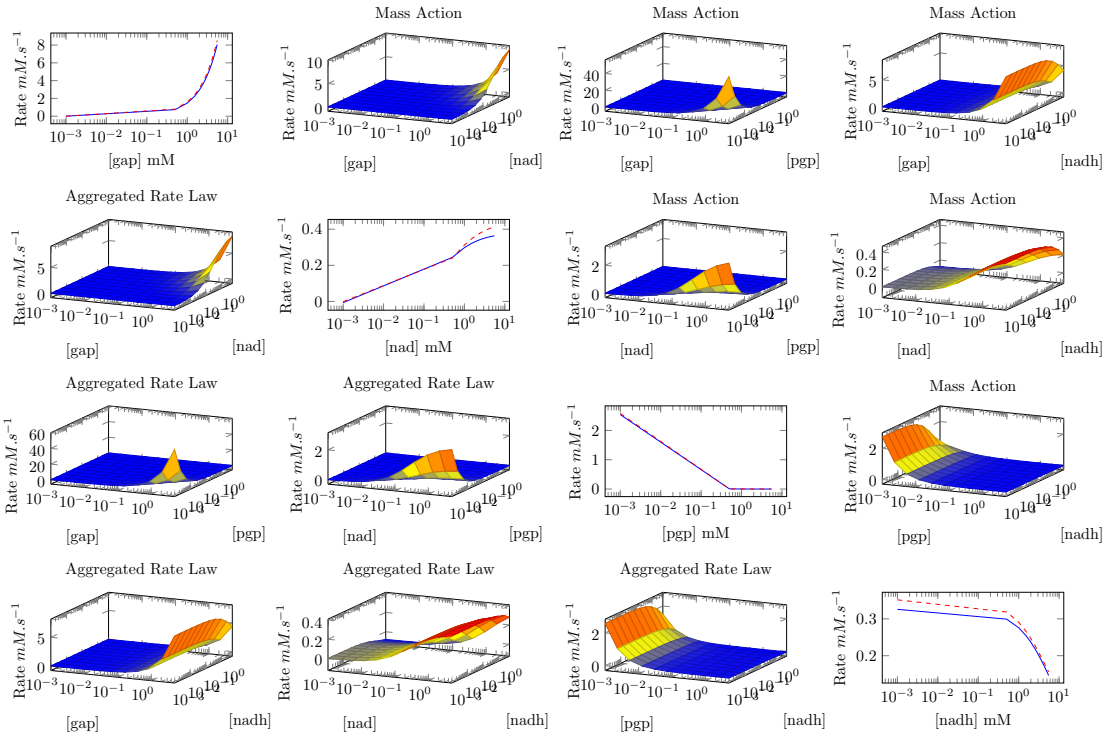


(a) Rate vs Compound graph.

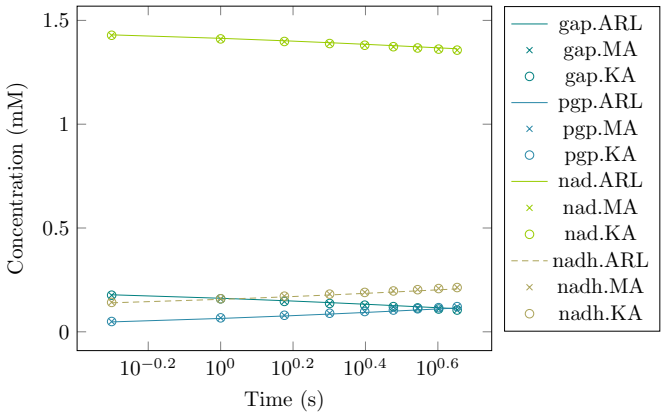


(b) Simulation from system initial conditions.

Figure A.5: Calibration of G6PDH Mass action enzymatic system to Aggregated Rate Law Mechanism.

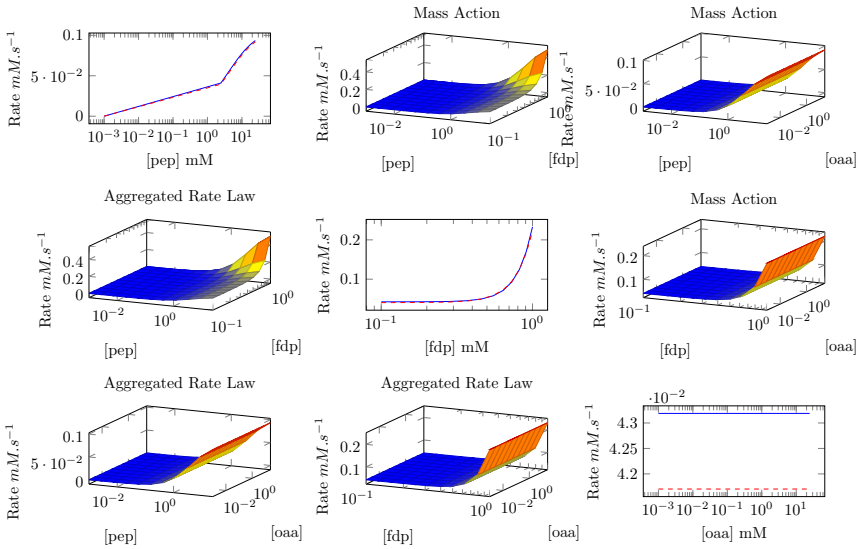


(a) Rate vs Compound graph.

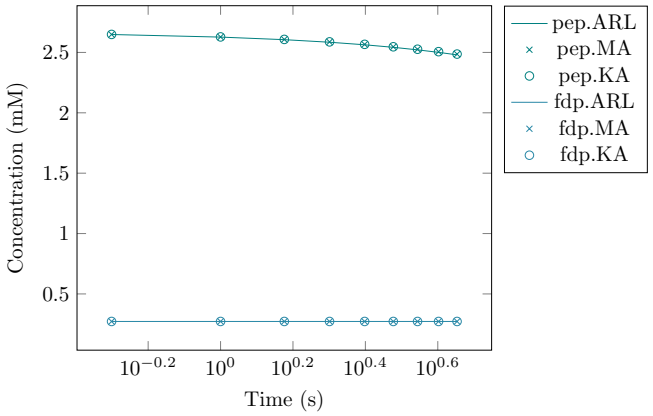


(b) Simulation from system initial conditions.

Figure A.6: Calibration of GAPDH Mass action enzymatic system to Aggregated Rate Law Mechanism.

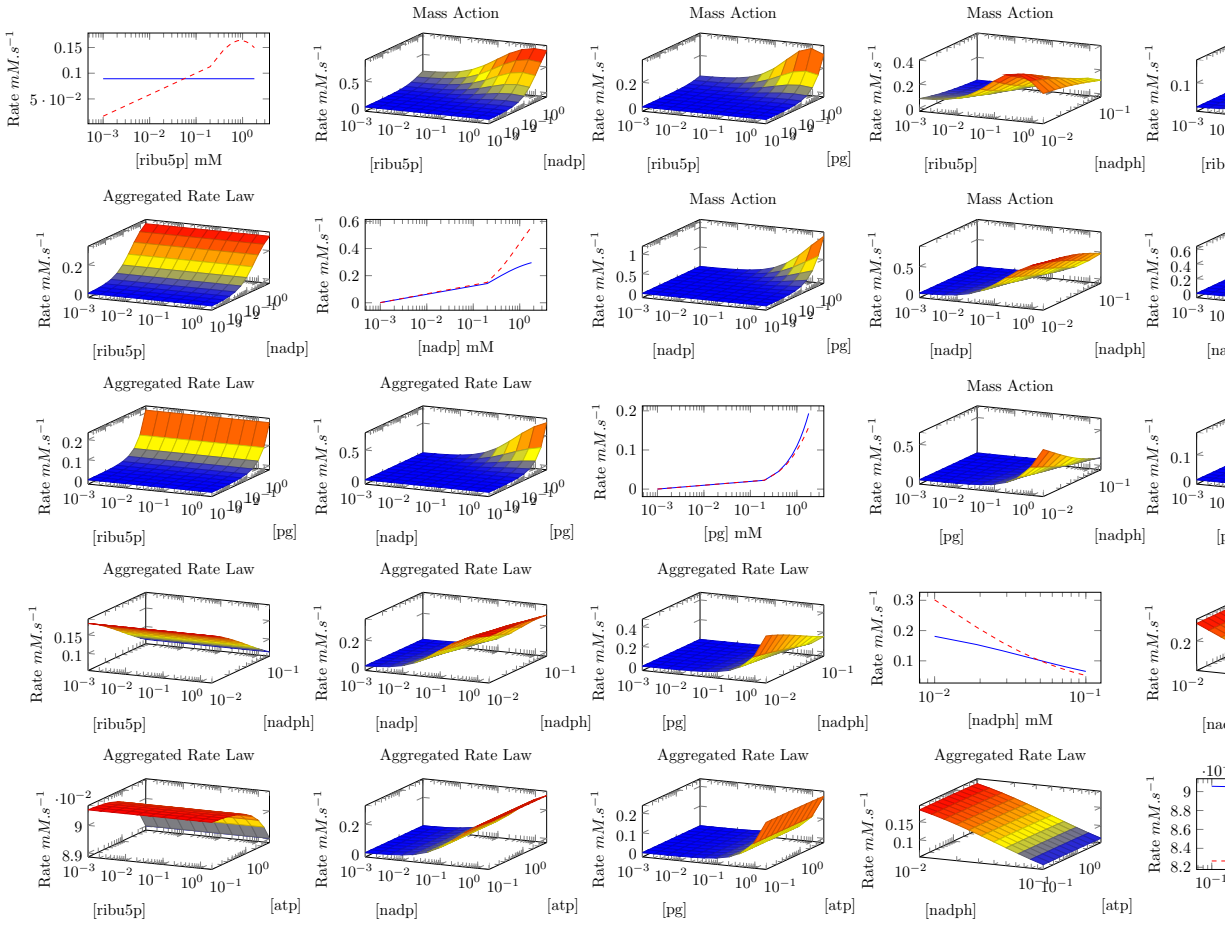


(a) Rate vs Compound graph.

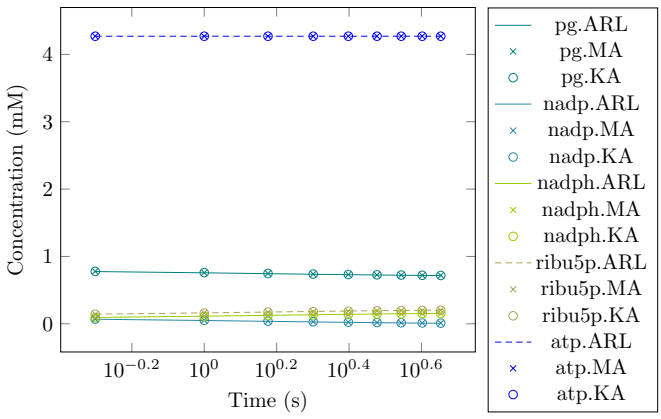


(b) Simulation from system initial conditions.

Figure A.7: Calibration of PEPCxylase Mass action enzymatic system to Aggregated Rate Law Mechanism.

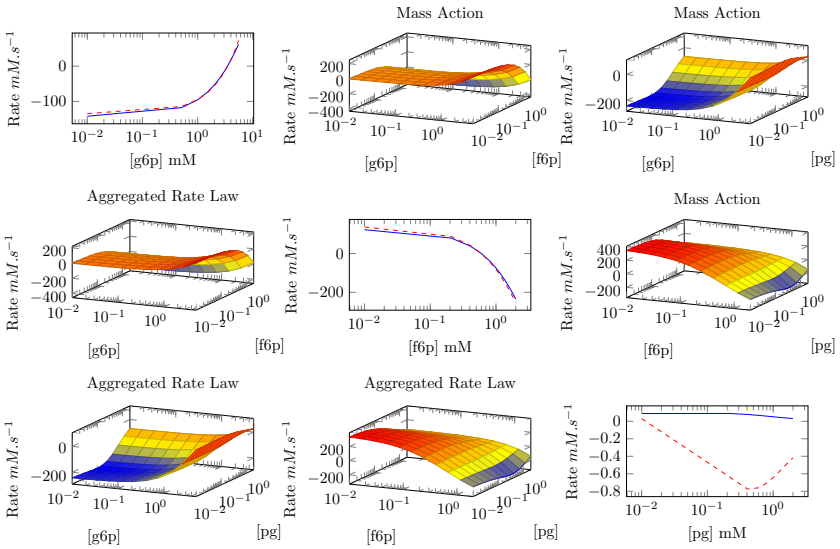


(a) Rate vs Compound graph.

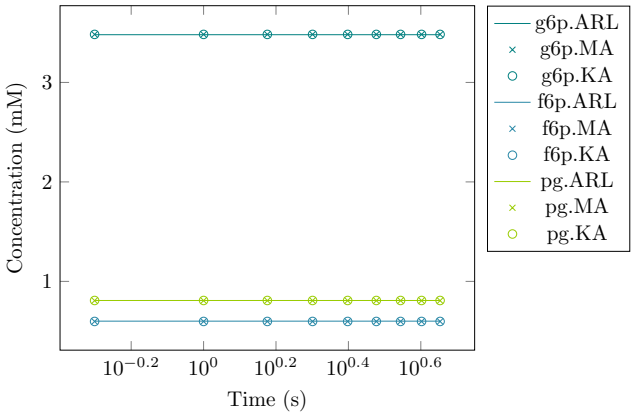


(b) Simulation from system initial conditions.

Figure A.8: Calibration of PGDH Mass action enzymatic system to Aggregated Rate Law Mechanism.

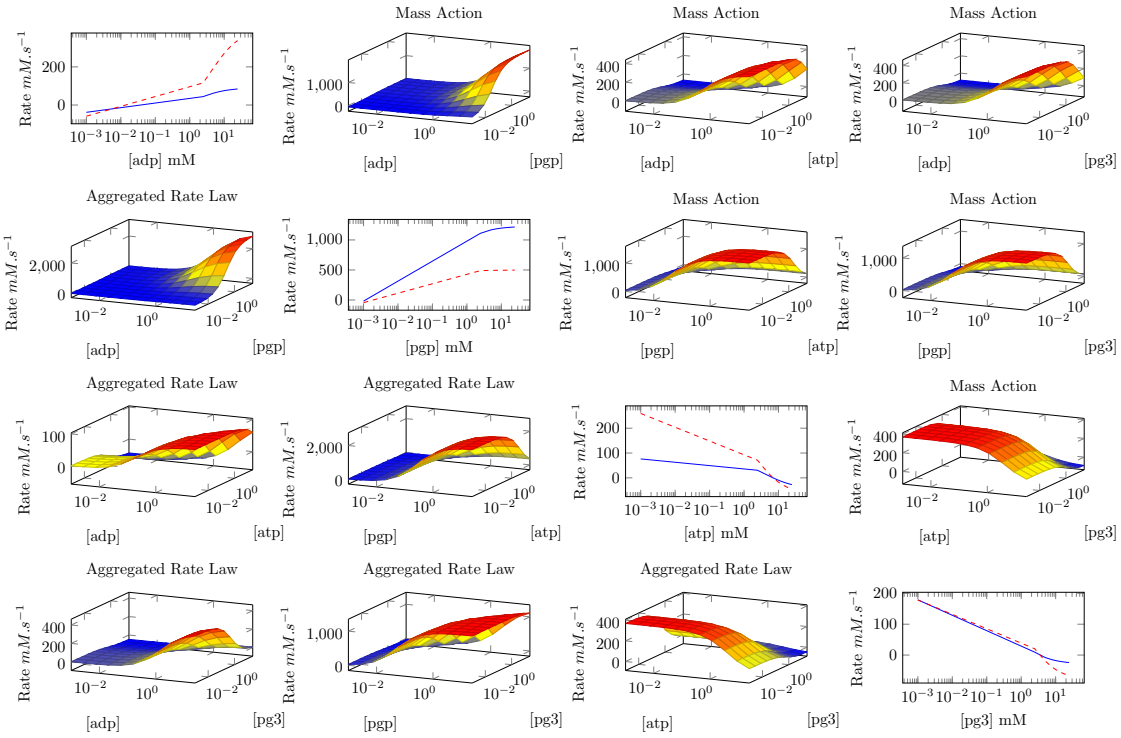


(a) Rate vs Compound graph.

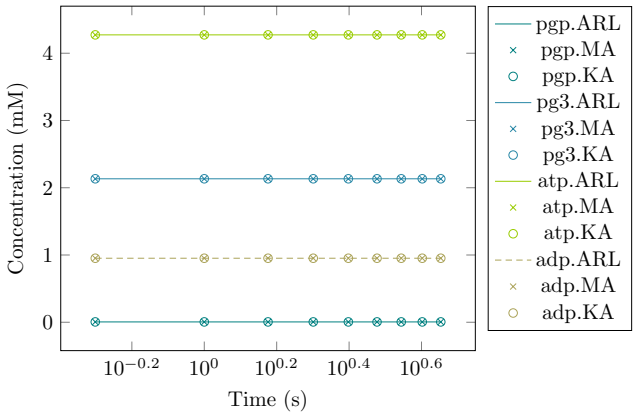


(b) Simulation from system initial conditions.

Figure A.9: Calibration of PGI Mass action enzymatic system to Aggregated Rate Law Mechanism.

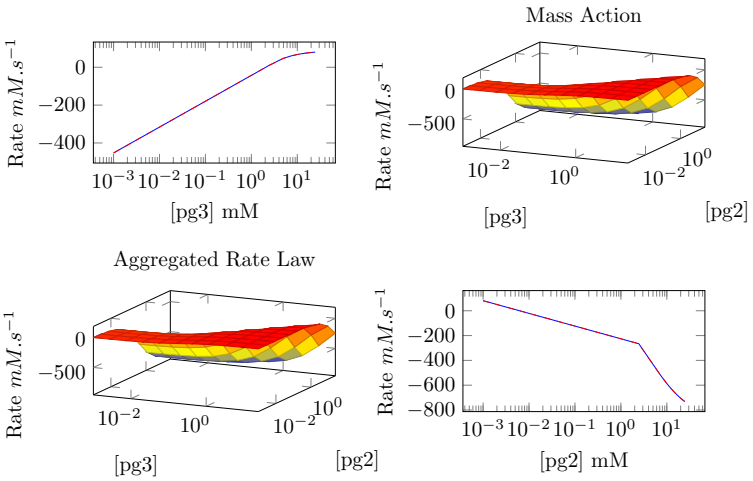


(a) Rate vs Compound graph.

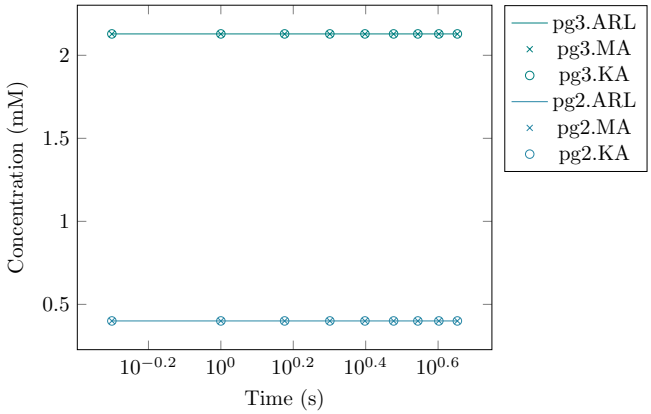


(b) Simulation from system initial conditions.

Figure A.10: Calibration of PGK Mass action enzymatic system to Aggregated Rate Law Mechanism.

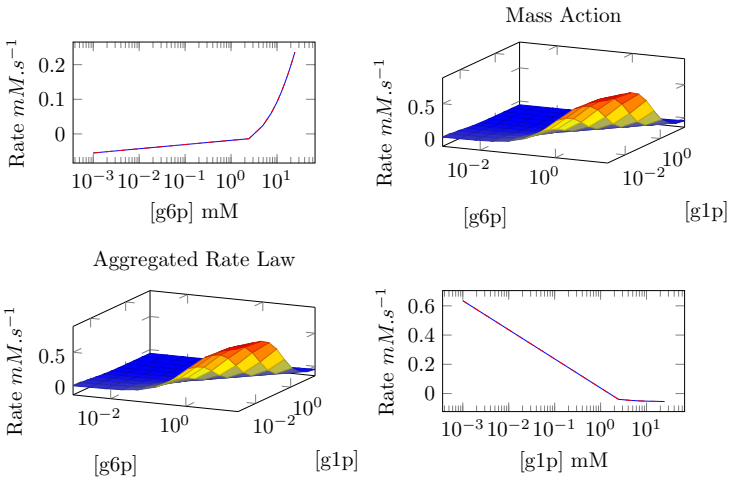


(a) Rate vs Compound graph.

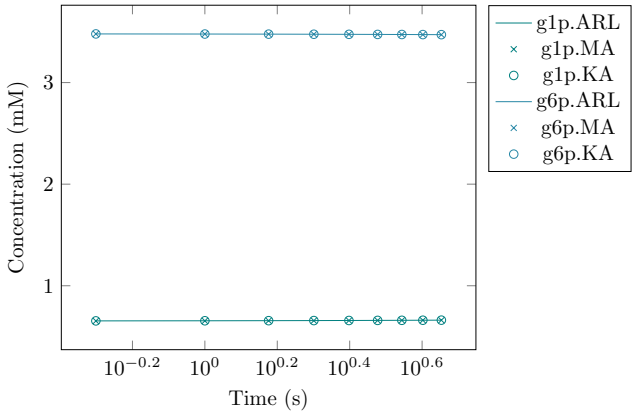


(b) Simulation from system initial conditions.

Figure A.11: Calibration of PGluMu Mass action enzymatic system to Aggregated Rate Law Mechanism.

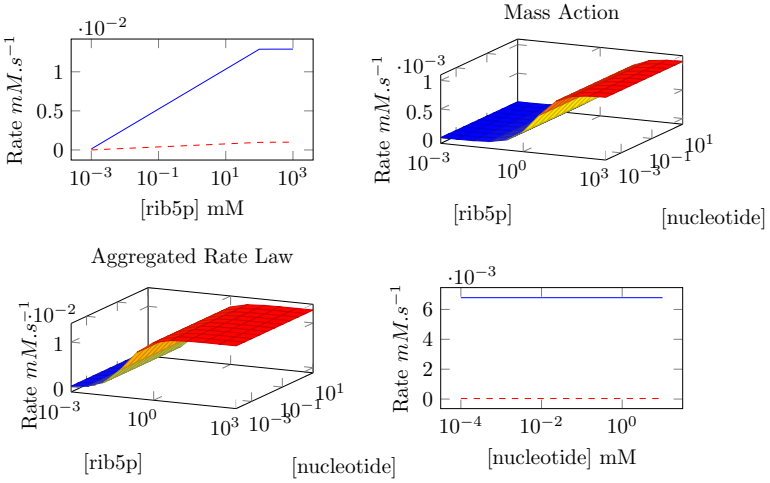


(a) Rate vs Compound graph.

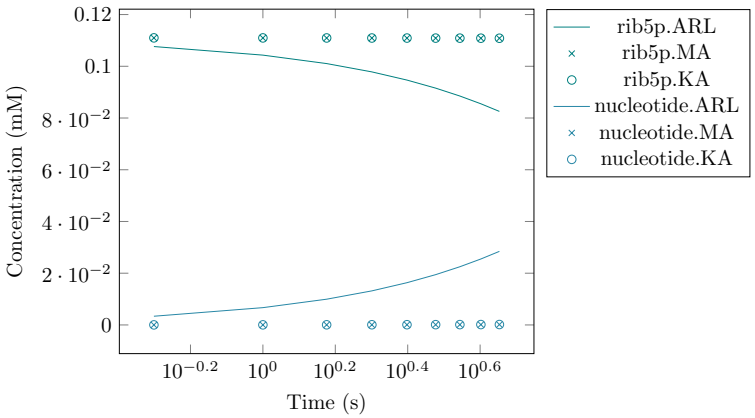


(b) Simulation from system initial conditions.

Figure A.12: Calibration of PGM Mass action enzymatic system to Aggregated Rate Law Mechanism.

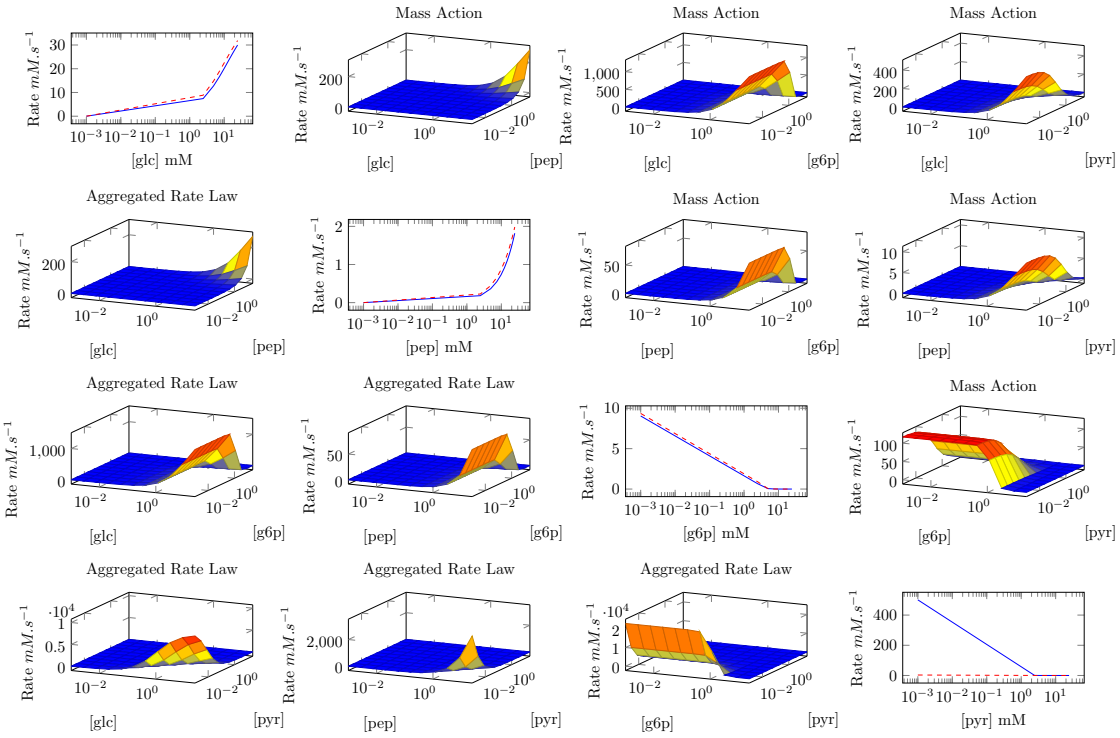


(a) Rate vs Compound graph.

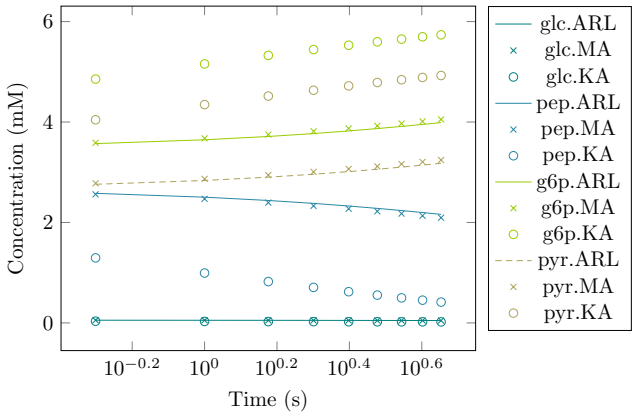


(b) Simulation from system initial conditions.

Figure A.13: Calibration of PPK Mass action enzymatic system to Aggregated Rate Law Mechanism.

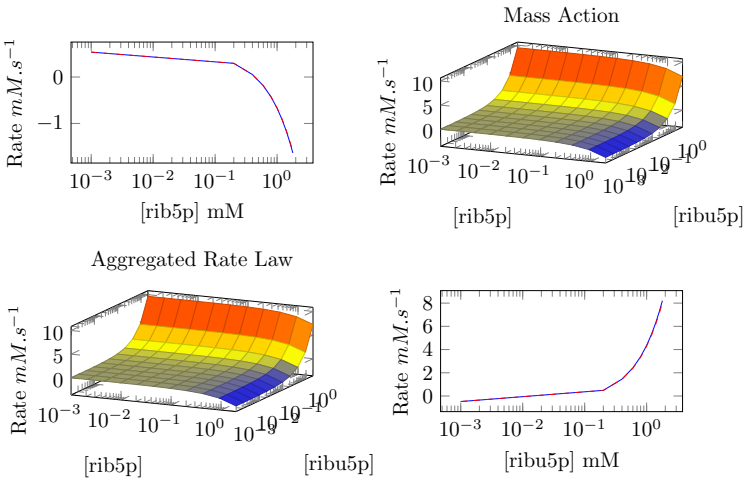


(a) Rate vs Compound graph.

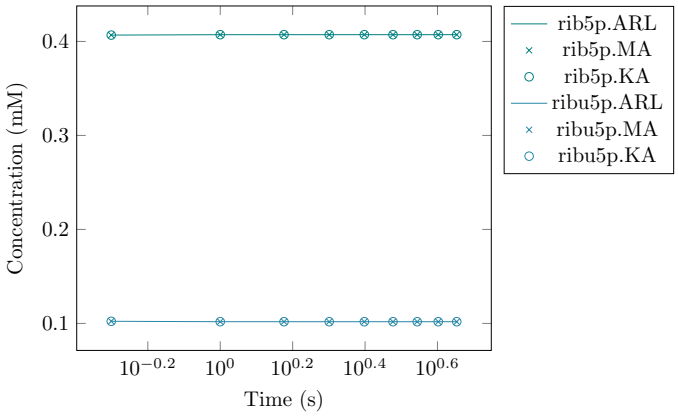


(b) Simulation from system initial conditions.

Figure A.14: Calibration of PTS Mass action enzymatic system to Aggregated Rate Law Mechanism.

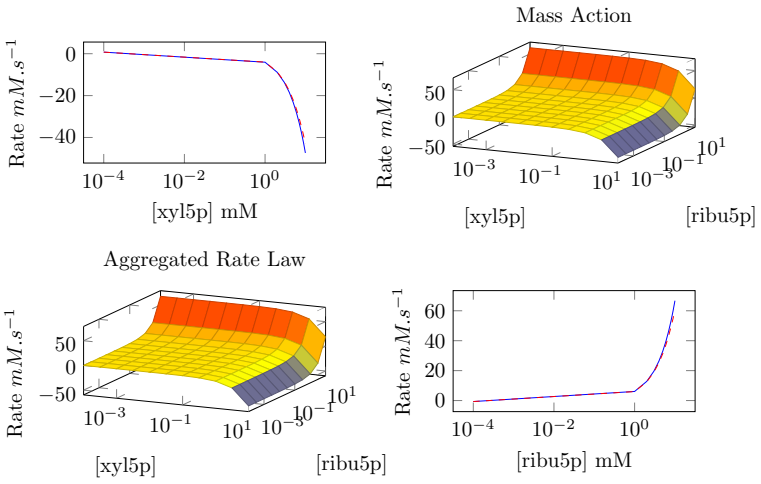


(a) Rate vs Compound graph.

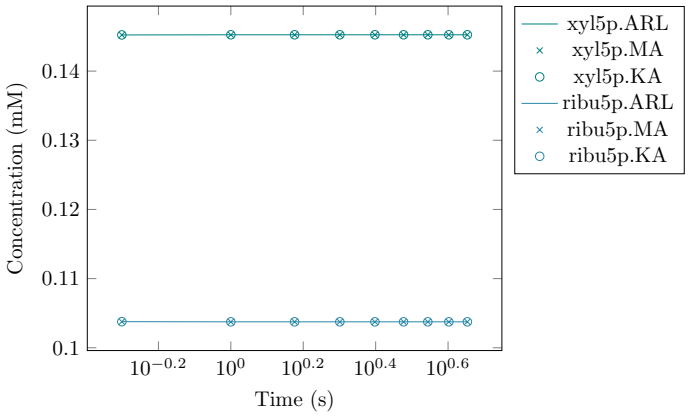


(b) Simulation from system initial conditions.

Figure A.15: Calibration of R5PI Mass action enzymatic system to Aggregated Rate Law Mechanism.

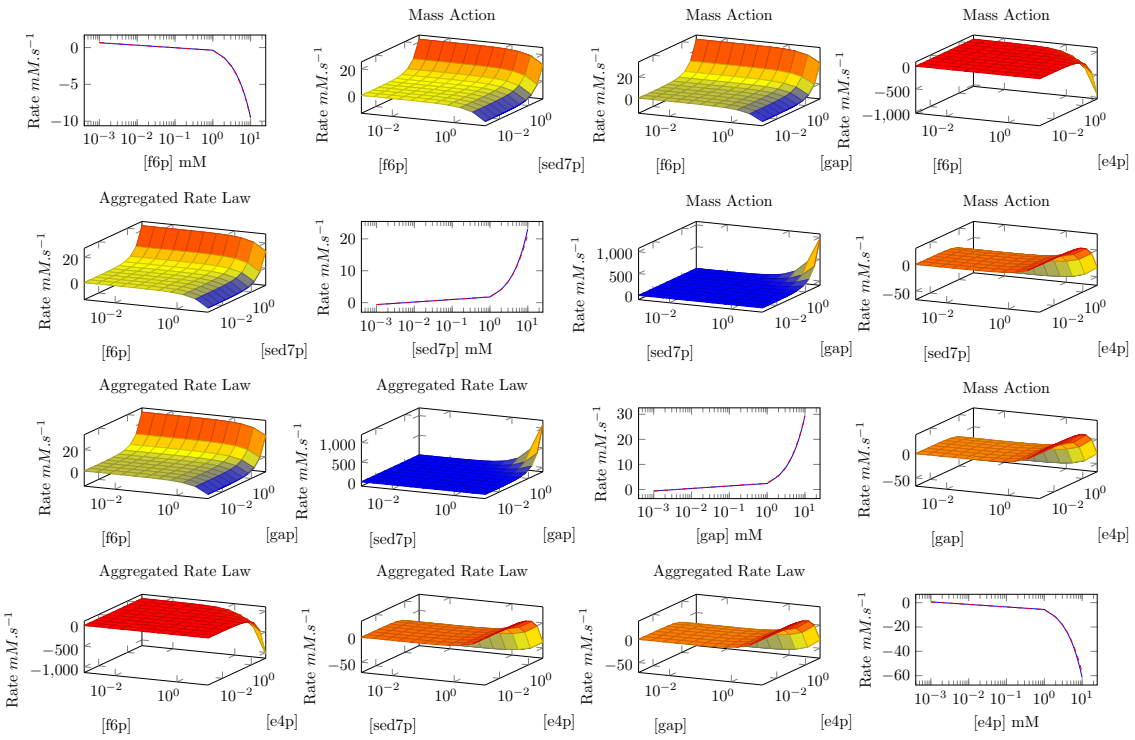


(a) Rate vs Compound graph.

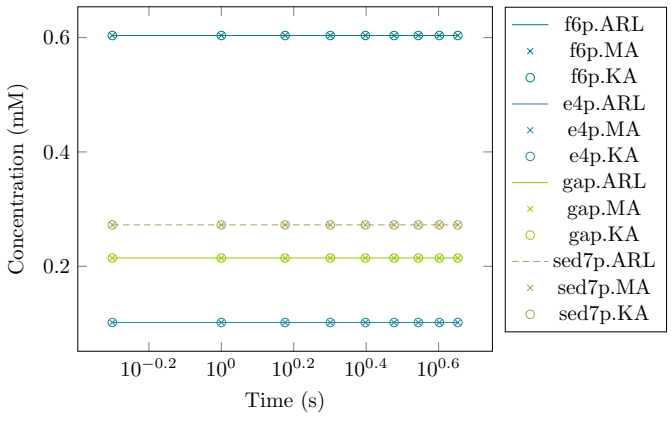


(b) Simulation from system initial conditions.

Figure A.16: Calibration of RU5P Mass action enzymatic system to Aggregated Rate Law Mechanism.

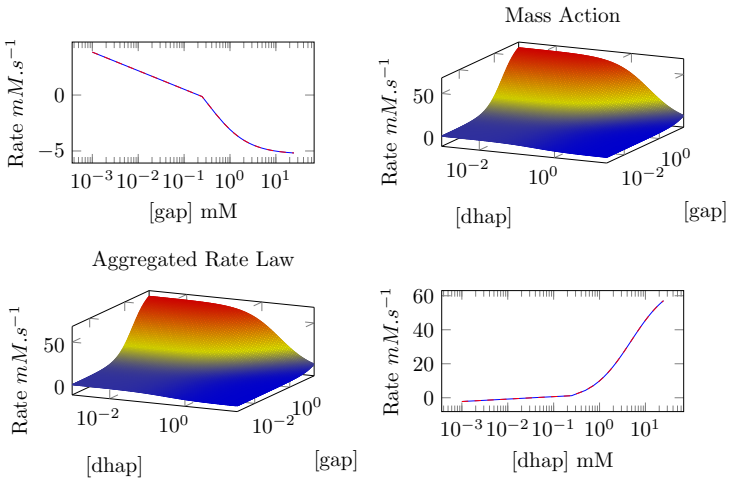


(a) Rate vs Compound graph.

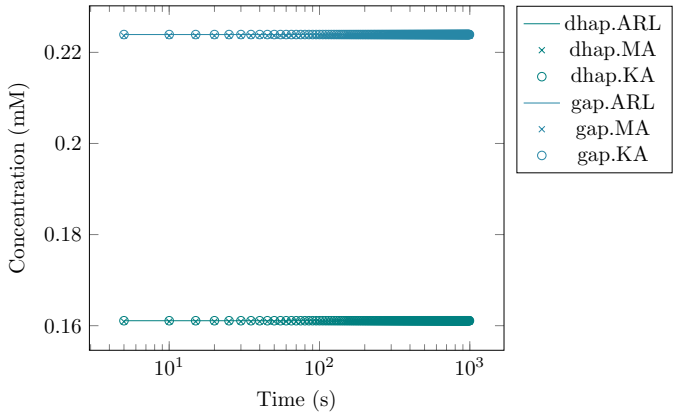


(b) Simulation from system initial conditions.

Figure A.17: Calibration of TA Mass action enzymatic system to Aggregated Rate Law Mechanism.

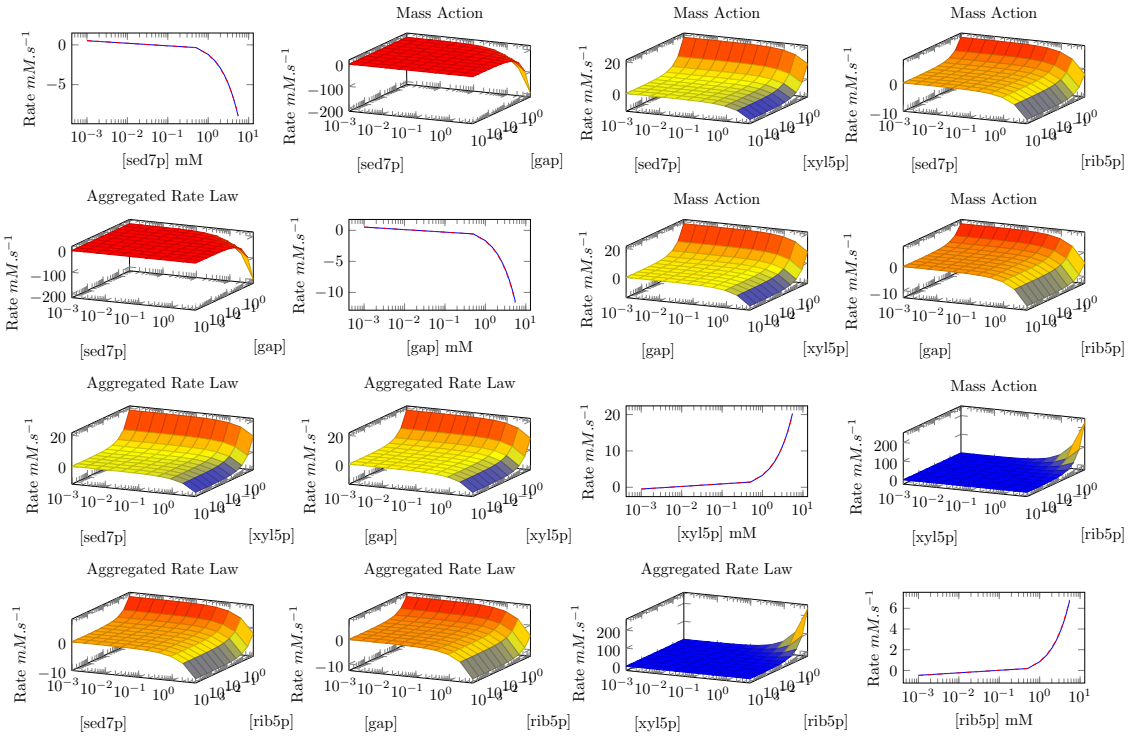


(a) Rate vs Compound graph.

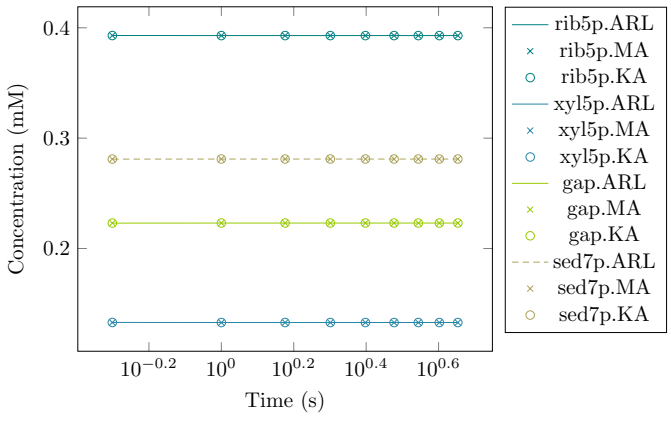


(b) Simulation from system initial conditions.

Figure A.18: Calibration of TIS Mass action enzymatic system to Aggregated Rate Law Mechanism.

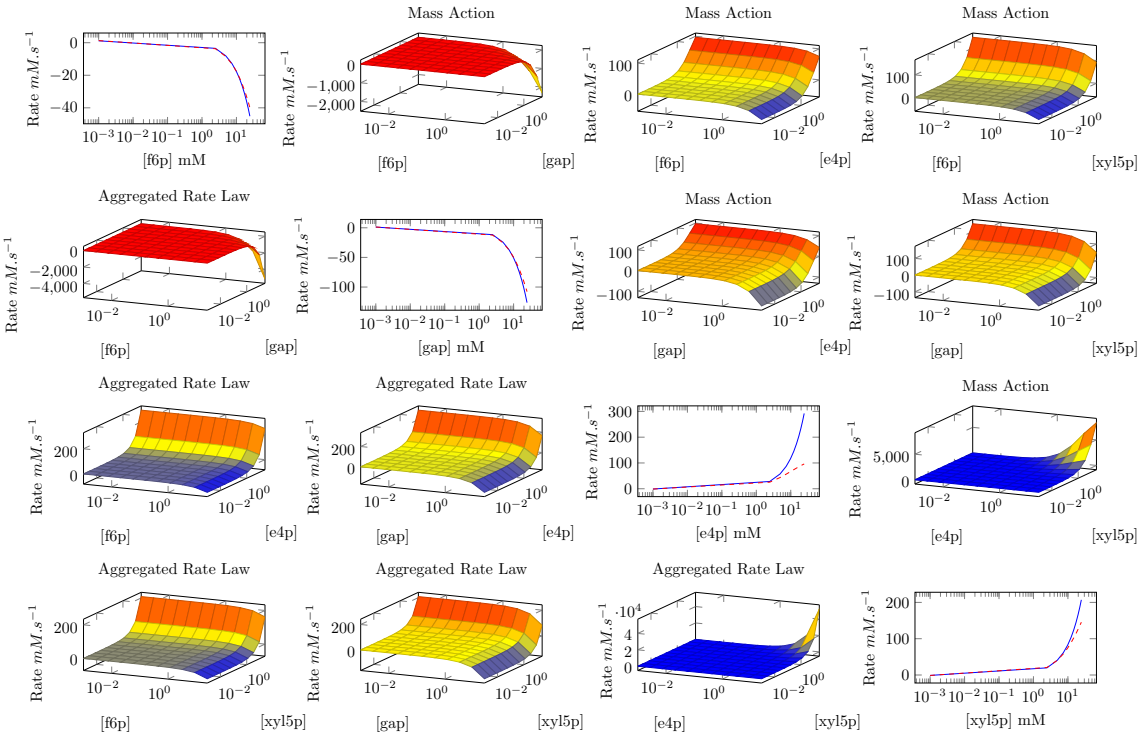


(a) Rate vs Compound graph.

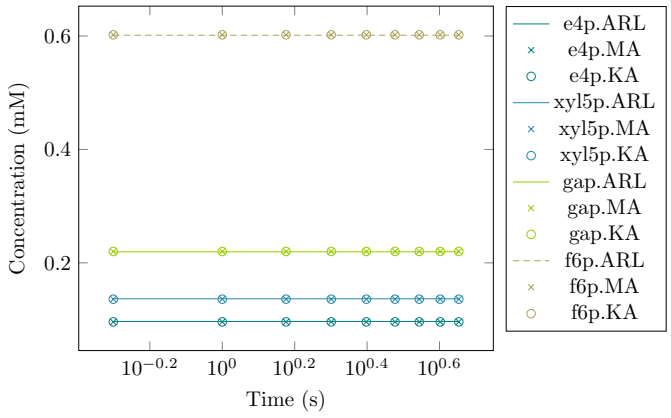


(b) Simulation from system initial conditions.

Figure A.19: Calibration of TKA Mass action enzymatic system to Aggregated Rate Law Mechanism.

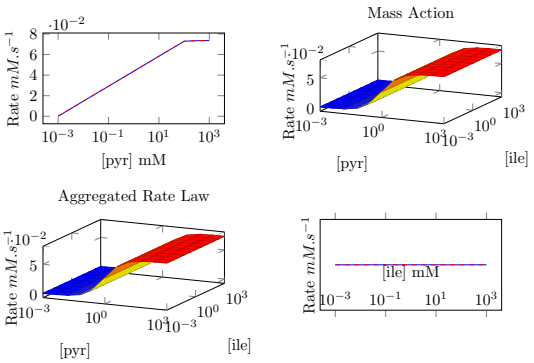


(a) Rate vs Compound graph.

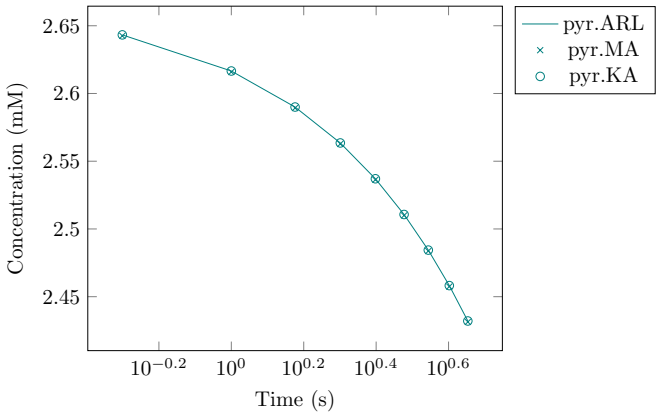


(b) Simulation from system initial conditions.

Figure A.20: Calibration of TKB Mass action enzymatic system to Aggregated Rate Law Mechanism.



(a) Rate vs Compound graph.



(b) Simulation from system initial conditions.

Figure A.21: Calibration of Synth2 Mass action enzymatic system to Aggregated Rate Law Mechanism.

A.0.1 Final Marl Model BioKroneckerScala Symbolic Representation

```
[Functions]

def muFunFixed():
    2.78E-5
end

def adpFun():
```

```
        0.582+1.73*(2.731^(-0.15*time))*(0.12*time+0.000214*time^3)
end

def ampFun():
        0.123+7.25*(time/(7.25+1.47*time+0.17*time^2))+1.073/(1.29+8.05*time)
end

def atpFun():
        4.27-4.163*(time/(0.657+1.43*time+0.0364*time^2))
end

def nadFun():
        1.314+1.314*2.73^(-0.0435*time-0.342)-(time+7.871)*(2.73^(-0.0218*time-0.171)
            /(8.481+time))
end

def nadhFun():
        0.0934+0.00111*2.371^(-0.123*time)*(0.844*time+0.104*time^3)
end

def nadpFun():
        0.159-0.00554*(time/(2.8-0.271*time+0.01*time^2))+0.182/(4.82+0.526*time)
end

def nadphFun():
    0.062+0.332*2.718^(-0.464*time)*(0.0166*time^1.58+0.000166*time^4.73+0.1312*10^-9*time
        ^7.89+0.1362*10^-12*time^11+0.1233*10^-15*time^14.2)
end


def pkR0(pkk0,pkKeq):
        Epk*pep*pkk0*((1/pkKeq)*(fdp/atp)^(4))/ ((1+(1/pkKeq))*(fdp/atp)^(4))
end

def pfkR0(pfkk0,pfkKeq,pfkatpn):
        Epfk*(atp^pfkatpn)*((1/pfkKeq)*(adp/pep)^(2))/ ((1+(1/pfkKeq))*(adp/pep)^(2))
end

def pfkR2(pfkk2,pfkf6pn):
        Epfk_atp*pfkk2*(f6p^pfkf6pn)
end

def rateLawExter(Dil,cfeed):
        ((Dil)*(cfeed-glc))
end

def rateK0K1(k0,k1):
        k0*(pyr^(4))*Epdh-k1*Epdh_pyr
end

[GlobalParameters]

rPfk2 = pfkR2(pfkk2=663220.6535256172,pfkf6pn=24.961890932718266) (pfkk2: [0.0,10.0],
      pfkf6pn: [0.0,1.0])

rPfk0 = pfkR0(pfkk0=1.0852286927072514,pfkKeq=1.003565508804163,pfkatpn
      =6.190400139404164) (pfkk0: [0.0,10.0],pfkKeq: [-3.0,10.0],pfkatpn: [0.0,1.0])
```

```
rPk0 = pkR0(pkk0=7.8685694024581 09E8,pkKeq=0.007849111186954508) // (pkk0: [0.0,10.0],pkKeq:
      [−3.0,10.0])

rRateK0K1 = rateK0K1(k0=42.1,k1=402420.87449511676)

rExter = rateLawExter(Dil=2.78E−5,cfeed=110.96)

[Compartments]

compartment outside {
        dimension = volume
        value = 65

        species {
            glc 0.055 (∗0.055)
        }

        inputs {glcFeed = 110.96;}

        compartment cell {
                dimension = volume
                value = 1
                species {
                 pep 2.67 (∗2.67)
                 g6p 3.48 (∗3.48)
                 pyr 2.67 (∗2.67)
                 f6p 0.6 (∗0.6)
                 g1p 0.653 (∗0.653)
                 pg 0.808 (∗0.808)
                 fdp 0.272 (∗0.272)
                 sed7p 0.276 (∗0.276)
                 gap 0.218 (∗0.218)
                 e4p 0.098 (∗0.098)
                 xyl5p 0.138 (∗0.138)
                 rib5p 0.398 (∗0.398)
                 dhap 0.167 (∗0.167)
                 pgp 0.00813025 (∗0.00813025)
                 pg3 2.1185 (∗2.1185)
                 pg2 0.399 (∗0.399)
                 ribu5p 0.111 (∗0.111)
         ser 0.0 (∗0.0)
                 polysac 0.0 (∗0.0)
                 murine 0.0 (∗0.0)
                 cho_mur 0.0 (∗0.0)
                 nucleotide 0.0 (∗0.0)
                 aromaticaminoacids 0.0 (∗0.0)
             glycerol 0.0 (∗0.0)
                 oaa 0.0 (∗0.0)
                 ile 0.0 (∗0.0)
                 aca 1.0 (∗0.0) [−3,0.7]
                 co2 0.0 (∗0.0)
                }

                inputs {coa = 1E−4;o2 = 16.25;
                amp = ampFun();
            atp = atpFun();
       adp = adpFun();

        nad = nadFun();
```

```
                nadp = nadpFun();
            nadh = nadhFun();
                nadph = nadphFun();
               }
        }
}

[Reactions]

vGLCExt: --- glc {

        modifiers{}

        enzymes{
        cell{
         }
        }

 -> glc @rExter (*rExter) [0.0,10.0]

}


vSerineSynth: pg3 --- ser {

        modifiers{}

        enzymes{
        cell{
           ESerSynth 0.001 (*0.001)
           ESerSynth_pg3 0.0 (*0.0)
         }
        }

ESerSynth + pg3 <-> ESerSynth_pg3 >@470019.00888823345 <@469993.29659735516
     (*470019.00888823345,*469993.29659735516) [0.0,8.0]---[0.0,10.0]
ESerSynth_pg3 -> ESerSynth + ser @25.712106994173883 (*25.712106994173883) [0.0,10.0] :slow

}


vPyrIn: --- pyr {

        modifiers{}

        enzymes{
        cell{
         }
        }

 -> pyr @0.0022627 (*0.0022627) [0.0,10.0]
 -> pyr @0.001037 (*0.001037) [0.0,10.0]

}


vGapIn: --- gap {

        modifiers{}
```

```
        enzymes{
        cell{
         }
        }

 -> gap @0.001037 (*0.001037) [0.0,10.0]

}


vPTS: glc + pep --- g6p + pyr {

        modifiers{}

        enzymes{
        cell{
           Epts 3.783118954706172E-6 (*3.783118954706172E-6)
           Epts_pep 6.029831727659324E-8 (*6.029831727659324E-8)
           Epts_glc_pep 3.827709869433558E-9 (*3.827709869433558E-9)
           Epts_g6p 1.4776985557367516E-11 (*1.4776985557367516E-11)
           Epts_pyr 2.602207010428719E-10 (*2.602207010428719E-10)
           Epts_pyr_I 1.0722546145720718E-9 (*1.0722546145720718E-9)
           Epts_pyr_II 1.8253251218852827E-11 (*1.8253251218852827E-11)
           Epts_pyr_III 9.310341038299109E-7 (*9.310341038299109E-7)
           Epts_pyr_IIII 2.0040433752649492E-16 (*2.0040433752649492E-16)
           Epts_I 1.0607203851607552E-6 (*1.0607203851607552E-6)
           Epts_II 3.1097255799095146E-7 (*3.1097255799095146E-7)
           Epts_III 2.458198146544232E-7 (*2.458198146544232E-7)
           Epts_IIII 8.135500177504968E-7 (*8.135500177504968E-7)
           Epts_pep_I 8.78872039240842E-4 (*8.78872039240842E-4)
           Epts_pep_II 6.003828626432092E-11 (*6.003828626432092E-11)
           Epts_pep_III 1.3037302028355134E-9 (*1.3037302028355134E-9)
           Epts_pep_IIII 7.675016225886025E-13 (*7.675016225886025E-13)
           Epts_glc_pep_I 1.1391563508421109E-4 (*1.1391563508421109E-4)
           Epts_glc_pep_II 1.3326279380138828E-9 (*1.3326279380138828E-9)
           Epts_glc_pep_III 1.0341400738926757E-12 (*1.0341400738926757E-12)
           Epts_glc_pep_IIII 9.219204058555948E-18 (*9.219204058555948E-18)
         }
        }

pep + Epts <-> Epts_pep >@19670.059863956474 <@1.0000005916391101
      (*19670.059863956474,*1.0000005916391101) [0.0,8.0]---[0.0,10.0]
Epts_pep + glc <-> Epts_glc_pep >@4.8230260320765525E7 <@200.42249808192256
      (*4.8230260320765525E7,*200.42249808192256) [0.0,8.0]---[0.0,10.0]
Epts_glc_pep -> Epts_g6p + pyr @5.234989615516764E7 (*5.234989615516764E7) [0.0,10.0] :
      slow
Epts_g6p -> g6p + Epts @1.356029031760482E10 (*1.356029031760482E10) [0.0,10.0]
pyr + Epts <-> Epts_pyr >@3.502970457313827 <@141028.3315536575
      (*3.502970457313827,*141028.3315536575) [0.0,8.0]---[0.0,10.0]
g6p + Epts <-> Epts_I >@7.427434187563649E7 <@9.238013271354692E8 (*7.427434187563649E7
      ,*9.238013271354692E8) [0.0,8.0]---[0.0,10.0]
Epts_I + g6p <-> Epts_II >@4.393716621494898 <@52.26387182038075
      (*4.393716621494898,*52.26387182038075) [0.0,8.0]---[0.0,10.0]
Epts_II + g6p <-> Epts_III >@1.0703319659549066E8 <@4.721873659570795E8
      (*1.0703319659549066E8,*4.721873659570795E8) [0.0,8.0]---[0.0,10.0]
Epts_III + g6p <-> Epts_IIII >@6.452386591324015E7 <@6.798978933670603E7
      (*6.452386591324015E7,*6.798978933670603E7) [0.0,8.0]---[0.0,10.0]
Epts_pep + g6p <-> Epts_pep_I >@6.4862777480111115E7 <@15519.072922666499
```

```
    (*6.48627774801111115E7,*15519.072922666499) [0.0,8.0]---[0.0,10.0]
Epts_pep_I + g6p <-> Epts_pep_II >@78.64610877676289 <@4.014812686432811E9
    (*78.64610877676289,*4.014812686432811E9) [0.0,8.0]---[0.0,10.0]
Epts_pep_II + g6p <-> Epts_pep_III >@2072.9173730314906 <@332.8995739884384
    (*2072.9173730314906,*332.8995739884384) [0.0,8.0]---[0.0,10.0]
Epts_pep_III + g6p <-> Epts_pep_IIII >@1.0000002396607397 <@5923.784022364391
    (*1.0000002396607397,*5923.784022364391) [0.0,8.0]---[0.0,10.0]
Epts_pyr + g6p <-> Epts_pyr_I >@6510.195209523406 <@5509.708434868172
    (*6510.195209523406,*5509.708434868172) [0.0,8.0]---[0.0,10.0]
Epts_pyr_I + g6p <-> Epts_pyr_II >@6147.343250147012 <@1259319.2557000252
    (*6147.343250147012,*1259319.2557000252) [0.0,8.0]---[0.0,10.0]
Epts_pyr_II + g6p <-> Epts_pyr_III >@1.142761760482617E8 <@7813.057643335574
    (*1.142761760482617E8,*7813.057643335574) [0.0,8.0]---[0.0,10.0]
Epts_pyr_III + g6p <-> Epts_pyr_IIII >@1.0000003320202309 <@1.620127911560665E10
    (*1.0000003320202309,*1.620127911560665E10) [0.0,8.0]---[0.0,10.0]
Epts_glc_pep + g6p <-> Epts_glc_pep_I >@8541.362786376698 <@1.0008604598598663
    (*8541.362786376698,*1.0008604598598663) [0.0,8.0]---[0.0,10.0]
Epts_glc_pep_I + g6p <-> Epts_glc_pep_II >@8092.292030933562 <@2.4123296589002085E9
    (*8092.292030933562,*2.4123296589002085E9) [0.0,8.0]---[0.0,10.0]
Epts_glc_pep_II + g6p <-> Epts_glc_pep_III >@3.149905447627534 <@14155.256752012448
    (*3.149905447627534,*14155.256752012448) [0.0,8.0]---[0.0,10.0]
Epts_glc_pep_III + g6p <-> Epts_glc_pep_IIII >@46696.2416120958 <@1.8266635169123493
    E10 (*46696.2416120958,*1.8266635169123493E10) [0.0,8.0]---[0.0,10.0]

}

//KA
vPGM: g6p --- g1p {

        modifiers{}

        enzymes{
        cell{
           Epgm 0.001 (*0.001)
           Epgm_g6p 0.0 (*0.0)
           Epgm_g1p 0.0 (*0.0)
         }
        }

Epgm + g6p <-> Epgm_g6p >@1160.6532898845637 <@4.3935842398368865E8
    (*1160.6532898845637,*4.3935842398368865E8) [0.0,8.0]---[0.0,10.0]
Epgm_g6p <-> Epgm_g1p >@1.2747978347679474E9 <@219.03091313489912 (*1.2747978347679474
    E9,*219.03091313489912) [0.0,10.0]---[0.0,10.0]
Epgm_g1p <-> Epgm + g1p >@839.8249750171041 <@65879.78391806714
    (*839.8249750171041,*65879.78391806714) [0.0,10.0]---[0.0,8.0]

}

//GlobalSim
vG1PAT: g1p + atp --- polysac + adp {

        modifiers{ fdp}

        enzymes{
        cell{
           Eg1pat_fdp0_adp 2.783077187068133E-17 (*2.783077187068133E-17)
           Eg1pat_fdp0 2.759242009499265E-12 (*2.759242009499265E-12)
           Eg1pat_fdp1_adp 2.6047631700683027E-13 (*2.6047631700683027E-13)
           Eg1pat_fdp1 1.1185927123148551E-7 (*1.1185927123148551E-7)
```

Eg1pat_fdp2_adp 2.6990181913733318E−5 (∗2.6990181913733318E−5)
Eg1pat_fdp2 1.0708204818804828E−6 (∗1.0708204818804828E−6)
Eg1pat_fdp0_polysac 6.011013919014954E−11 (∗6.011013919014954E−11)
Eg1pat_fdp1_polysac 1.8774106054457852E−14 (∗1.8774106054457852E−14)
Eg1pat_fdp2_polysac 6.63001005184783E−7 (∗6.63001005184783E−7)
Eg1pat_fdp0_polysac_adp 6.904843814799606E−20 (∗6.904843814799606E−20)
Eg1pat_fdp1_polysac_adp 1.1341278073581229E−8 (∗1.1341278073581229E−8)
Eg1pat_fdp2_polysac_adp 3.9152068168618993E−10 (∗3.9152068168618993E−10)
Eg1pat_fdp0_g1p_atp 1.924324934414624E−12 (∗1.924324934414624E−12)
Eg1pat_fdp1_g1p_atp 9.147337318394362E−4 (∗9.147337318394362E−4)
Eg1pat_fdp2_g1p_atp 8.374441028070024E−9 (∗8.374441028070024E−9)
Eg1pat_fdp0_atp 2.9205562546105214E−12 (∗2.9205562546105214E−12)
Eg1pat_fdp1_atp 5.760679565983262E−6 (∗5.760679565983262E−6)
Eg1pat_fdp2_atp 3.7506758370159944E−6 (∗3.7506758370159944E−6)
Eg1pat_fdp0_g1p 5.421820753316645E−12 (∗5.421820753316645E−12)
Eg1pat_fdp1_g1p 1.4153821504208194E−7 (∗1.4153821504208194E−7)
Eg1pat_fdp2_g1p 4.676028920484567E−5 (∗4.676028920484567E−5)
}
}

fdp + Eg1pat_fdp1 −> Eg1pat_fdp2 @4087664.386479138 (∗4087664.386479138) [0.0,8.0]
fdp + Eg1pat_fdp0 −> Eg1pat_fdp1 @2808066.9826038526 (∗2808066.9826038526) [0.0,8.0]
Eg1pat_fdp2 −> fdp + Eg1pat_fdp1 @124285.14439600696 (∗124285.14439600696) [0.0,10.0]
Eg1pat_fdp1 −> fdp + Eg1pat_fdp0 @20.161087056435548 (∗20.161087056435548) [0.0,10.0]
g1p + Eg1pat_fdp2 −> Eg1pat_fdp2_g1p @6712052.09587378 (∗6712052.09587378) [0.0,8.0]
g1p + Eg1pat_fdp1 −> Eg1pat_fdp1_g1p @17.55427613871025 (∗17.55427613871025) [0.0,8.0]
g1p + Eg1pat_fdp0 −> Eg1pat_fdp0_g1p @3792.827062052138 (∗3792.827062052138) [0.0,8.0]
Eg1pat_fdp2_g1p −> g1p + Eg1pat_fdp2 @97870.79989101064 (∗97870.79989101064) [0.0,10.0]
Eg1pat_fdp1_g1p −> g1p + Eg1pat_fdp1 @1.2325906011463107E8 (∗1.2325906011463107E8)
[0.0,10.0]
Eg1pat_fdp0_g1p −> g1p + Eg1pat_fdp0 @21202.541700048016 (∗21202.541700048016) [0.0,10.0]
Eg1pat_fdp2 + atp −> Eg1pat_fdp2_atp @80281.44287575649 (∗80281.44287575649) [0.0,8.0]
Eg1pat_fdp1 + atp −> Eg1pat_fdp1_atp @5.058351548149602E7 (∗5.058351548149602E7)
[0.0,8.0]
Eg1pat_fdp0 + atp −> Eg1pat_fdp0_atp @894190.9450849403 (∗894190.9450849403) [0.0,8.0]
Eg1pat_fdp2_atp −> Eg1pat_fdp2 + atp @112781.47906483682 (∗112781.47906483682) [0.0,10.0]
Eg1pat_fdp1_atp −> Eg1pat_fdp1 + atp @1165566.7270742091 (∗1165566.7270742091) [0.0,10.0]
Eg1pat_fdp0_atp −> Eg1pat_fdp0 + atp @3570141.141246303 (∗3570141.141246303) [0.0,10.0]
Eg1pat_fdp2_g1p + atp −> Eg1pat_fdp2_g1p_atp @359.0314131835494 (∗359.0314131835494)
[0.0,8.0]
Eg1pat_fdp1_g1p + atp −> Eg1pat_fdp1_g1p_atp @9481.415935824743 (∗9481.415935824743)
[0.0,8.0]
Eg1pat_fdp0_g1p + atp −> Eg1pat_fdp0_g1p_atp @2347566.8860264793 (∗2347566.8860264793)
[0.0,8.0]
Eg1pat_fdp2_g1p_atp −> Eg1pat_fdp2_g1p + atp @1572550.7870481152 (∗1572550.7870481152)
[0.0,10.0]
Eg1pat_fdp1_g1p_atp −> Eg1pat_fdp1_g1p + atp @19078.334764246825 (∗19078.334764246825)
[0.0,10.0]
Eg1pat_fdp0_g1p_atp −> Eg1pat_fdp0_g1p + atp @2.8299346679881293E7
(∗2.8299346679881293E7) [0.0,10.0]
g1p + Eg1pat_fdp2_atp −> Eg1pat_fdp2_g1p_atp @547216.3297743177 (∗547216.3297743177)
[0.0,8.0]
g1p + Eg1pat_fdp1_atp −> Eg1pat_fdp1_g1p_atp @4697803.196084147 (∗4697803.196084147)
[0.0,8.0]
g1p + Eg1pat_fdp0_atp −> Eg1pat_fdp0_g1p_atp @57674.666350686595 (∗57674.666350686595)
[0.0,8.0]
Eg1pat_fdp2_g1p_atp −> g1p + Eg1pat_fdp2_atp @1.6472637648863724E8
(∗1.6472637648863724E8) [0.0,10.0]
Eg1pat_fdp1_g1p_atp −> g1p + Eg1pat_fdp1_atp @6.314687151688067 (∗6.314687151688067)

[0.0,10.0]
Eg1pat_fdp0_g1p_atp -> g1p + Eg1pat_fdp0_atp @50.59231233151144 (*50.59231233151144) [0.0,10.0]
Eg1pat_fdp2_g1p_atp -> Eg1pat_fdp2_polysac_adp @2.847631566716145E7 (*2.847631566716145E7) [0.0,10.0]
Eg1pat_fdp1_g1p_atp -> Eg1pat_fdp1_polysac_adp @2.4221341014864057 (*2.4221341014864057) [0.0,10.0]
Eg1pat_fdp0_g1p_atp -> Eg1pat_fdp0_polysac_adp @168.35388256680358 (*168.35388256680358) [0.0,10.0]
Eg1pat_fdp2_polysac_adp -> Eg1pat_fdp2_g1p_atp @6.024804569170699E8 (*6.024804569170699E8) [0.0,10.0]
Eg1pat_fdp1_polysac_adp -> Eg1pat_fdp1_g1p_atp @165229.6041971684 (*165229.6041971684) [0.0,10.0]
Eg1pat_fdp0_polysac_adp -> Eg1pat_fdp0_g1p_atp @6.630575930902632E7 (*6.630575930902632E7) [0.0,10.0]
Eg1pat_fdp2_polysac_adp -> Eg1pat_fdp2_adp + polysac @6574276.215014287 (*6574276.215014287) [0.0,10.0]
Eg1pat_fdp1_polysac_adp -> Eg1pat_fdp1_adp + polysac @30081.83427310842 (*30081.83427310842) [0.0,10.0]
Eg1pat_fdp0_polysac_adp -> Eg1pat_fdp0_adp + polysac @177128.60377142864 (*177128.60377142864) [0.0,10.0]
Eg1pat_fdp2_polysac_adp -> Eg1pat_fdp2_polysac + adp @40105.03661853396 (*40105.03661853396) [0.0,10.0]
Eg1pat_fdp1_polysac_adp -> Eg1pat_fdp1_polysac + adp @46.417257582837316 (*46.417257582837316) [0.0,10.0]
Eg1pat_fdp0_polysac_adp -> Eg1pat_fdp0_polysac + adp @4.625405628586743E9 (*4.625405628586743E9) [0.0,10.0]
Eg1pat_fdp2_polysac -> Eg1pat_fdp2 + polysac @23.683148521916145 (*23.683148521916145) [0.0,10.0]
Eg1pat_fdp1_polysac -> Eg1pat_fdp1 + polysac @2.8040271218932714E7 (*2.8040271218932714E7) [0.0,10.0]
Eg1pat_fdp0_polysac -> Eg1pat_fdp0 + polysac @5.313197386559065 (*5.313197386559065) [0.0,10.0]
Eg1pat_fdp2_adp -> Eg1pat_fdp2 + adp @95.36671940648668 (*95.36671940648668) [0.0,10.0]
Eg1pat_fdp1_adp -> Eg1pat_fdp1 + adp @1.309779143743662E9 (*1.309779143743662E9) [0.0,10.0]
Eg1pat_fdp0_adp -> Eg1pat_fdp0 + adp @439.4579316226164 (*439.4579316226164) [0.0,10.0]

}

//KA
vPGI: g6p --- f6p {

        modifiers{ pg}

        enzymes{
        cell{
           Epgi 0.001 (*0.001)
           Epgi_g6p 0.0 (*0.0)
           Epgi_f6p 0.0 (*0.0)
           Epgi_pg 0.0 (*0.0)
           Epgi_pg_pg 0.0 (*0.0)
           Epgi_pg_f6p 0.0 (*0.0)
           Epgi_pg_g6p 0.0 (*0.0)
         }
        }

Epgi + g6p <-> Epgi_g6p >@330995.58572141704 <@308898.2586386819 (*330995.58572141704,*308898.2586386819) [0.0,8.0]---[0.0,10.0]

```
Epgi_g6p -> Epgi + f6p @650987.8178089942 (*650987.8178089942) [0.0,10.0]
Epgi + f6p -> Epgi_f6p @1301350.6923370787 (*1301350.6923370787) [0.0,8.0]
Epgi_f6p -> Epgi + f6p @6.589336748844003 (*6.589336748844003) [0.0,10.0]
Epgi_f6p -> Epgi + g6p @346150.6596492528 (*346150.6596492528) [0.0,10.0]
pg + Epgi <-> Epgi_pg >@1.6235374683748644E7 <@1623413.8684846556 (*1.6235374683748644
    E7,*1623413.8684846556) [0.0,8.0]---[0.0,10.0]
pg + Epgi_pg <-> Epgi_pg_pg >@2.290225850361946E7 <@9160473.679991715
    (*2.290225850361946E7,*9160473.679991715) [0.0,8.0]---[0.0,10.0]
f6p + Epgi_pg <-> Epgi_pg_f6p >@650700.4852738179 <@18.79899026910738
    (*650700.4852738179,*18.79899026910738) [0.0,8.0]---[0.0,10.0]
g6p + Epgi_pg -> Epgi_pg_g6p @112896.27870790965 (*112896.27870790965) [0.0,8.0]
Epgi_pg_f6p -> g6p + Epgi_pg @9.607012287399351E9 (*9.607012287399351E9) [0.0,10.0]
Epgi_pg_g6p -> g6p + Epgi_pg @3774.880945544257 (*3774.880945544257) [0.0,10.0]
Epgi_pg_g6p -> f6p + Epgi_pg @650988.1999683973 (*650988.1999683973) [0.0,10.0]

}

//KA
vMurSynth: f6p --- murine {

        modifiers{}

        enzymes{
        cell{
           Emur 0.001 (*0.001)
           Emur_f6p 0.0 (*0.0)
         }
        }

f6p + Emur <-> Emur_f6p >@9.793623090350708E7 <@1.702214465239815E8 (*9.793623090350708
    E7,*1.702214465239815E8) [0.0,8.0]---[0.0,10.0]
Emur_f6p -> murine + Emur @1.0 (*1.0) [0.0,10.0]

}

//KA
vALDO: fdp --- dhap + gap {

        modifiers{}

        enzymes{
        cell{
           Ealdo 0.001 (*0.001) [-3.0,-2.0]
           Ealdo_fdp 0.0 (*0.0) [-3.0,-5.0]
           Ealdo_dhap_gap 0.0 (*0.0) [-3.0,-5.0]
           Ealdo_dhap 0.0 (*0.0) [-3.0,-5.0]
         }
        }

Ealdo + fdp <-> Ealdo_fdp >@6610381.514684971 <@9179635.500834823
    (*6610381.514684971,*9179635.500834823) [0.0,12.0]---[0.0,10.0]
Ealdo_fdp <-> Ealdo_dhap_gap >@19779.867595307092 <@35731.23585387635
    (*19779.867595307092,*35731.23585387635) [0.0,10.0]---[0.0,10.0]
Ealdo_dhap_gap <-> gap + Ealdo_dhap >@163271.39045632462 <@1984270.402729924
    (*163271.39045632462,*1984270.402729924) [0.0,10.0]---[0.0,12.0]
Ealdo_dhap <-> dhap + Ealdo >@573237.7623436135 <@134228.8815777503
    (*573237.7623436135,*134228.8815777503) [0.0,10.0]---[0.0,12.0]

}
```

```
//KA
vTIS: dhap --- gap {

        modifiers{}

        enzymes{
        cell{
           Etis 0.001 (*0.001)
           Etis_dhap 0.0 (*0.0)
           Etis_gap 0.0 (*0.0)
         }
        }

Etis + dhap <-> Etis_dhap >@39462.29077851251 <@4.692960139235987E9
     (*39462.29077851251,*4.692960139235987E9) [0.0,8.0]---[0.0,10.0]
Etis_dhap <-> Etis_gap >@9.503593582872984E9 <@16013.34374260495 (*9.503593582872984E9
     ,*16013.34374260495) [0.0,10.0]---[0.0,10.0]
Etis_gap <-> Etis + gap >@68675.35585755992 <@246562.69851666034
     (*68675.35585755992,*246562.69851666034) [0.0,10.0]---[0.0,8.0]

}

//KA
vG3PDH: dhap --- glycerol {

        modifiers{}

        enzymes{
        cell{
           Eg3pdh 0.001 (*0.001)
           Eg3pdh_dhap 0.0 (*0.0)
         }
        }

Eg3pdh + dhap <-> Eg3pdh_dhap >@1000000.0 <@999988.3788653208
     (*1000000.0,*999988.3788653208) [0.0,8.0]---[0.0,10.0]
Eg3pdh_dhap -> Eg3pdh + glycerol @11.62042698826148 (*11.62042698826148) [0.0,10.0]

}

//KA
vGAPDH: gap + nad --- pgp + nadh {

        modifiers{}

        enzymes{
        cell{
           Egapdh 0.001 (*0.001)
           Egapdh_gap 0.0 (*0.0)
           Egapdh_pgp 0.0 (*0.0)
           Egapdh_nad 0.0 (*0.0)
           Egapdh_nadh 0.0 (*0.0)
           Egapdh_gap_nad 0.0 (*0.0)
           Egapdh_pgp_nadh 0.0 (*0.0)
         }
        }

Egapdh + gap <-> Egapdh_gap >@1120117.2411207086 <@903070.6133008203
```

```
      (*1120117.2411207086,*903070.6133008203) [0.0,8.0]---[0.0,10.0]
Egapdh + nad <-> Egapdh_nad >@7807.221596387721 <@744246.1068546197
      (*7807.221596387721,*744246.1068546197) [0.0,8.0]---[0.0,10.0]
Egapdh_gap + nad <-> Egapdh_gap_nad >@2825818.0322683873 <@244796.59965101167
      (*2825818.0322683873,*244796.59965101167) [0.0,8.0]---[0.0,10.0]
Egapdh_nad + gap <-> Egapdh_gap_nad >@241588.638482031 <@396.8523135400985
      (*241588.638482031,*396.8523135400985) [0.0,8.0]---[0.0,10.0]
Egapdh_gap_nad <-> Egapdh_pgp_nadh >@4.835694944353578E7 <@9792.280444209222
      (*4.835694944353578E7,*9792.280444209222) [0.0,10.0]---[0.0,10.0]
Egapdh_pgp_nadh <-> pgp + Egapdh_nadh >@5203035.907279214 <@403546.2565682074
      (*5203035.907279214,*403546.2565682074) [0.0,10.0]---[0.0,8.0]
Egapdh_pgp_nadh <-> Egapdh_pgp + nadh >@4955.708975515733 <@1.1123294424598194
      (*4955.708975515733,*1.1123294424598194) [0.0,10.0]---[0.0,8.0]
Egapdh_pgp <-> pgp + Egapdh >@1252.028022562376 <@9.910017542829737E7
      (*1252.028022562376,*9.910017542829737E7) [0.0,10.0]---[0.0,10.0]
Egapdh_nadh <-> Egapdh + nadh >@1.4049752168701185E7 <@22521.51557298744
      (*1.4049752168701185E7,*22521.51557298744) [0.0,10.0]---[0.0,10.0]

}

//KA
vPGK: pgp + adp --- pg3 + atp {

        modifiers{}

        enzymes{
        cell{
           Epgk 0.001 (*0.001)
           Epgk_pgp 0.0 (*0.0)
           Epgk_pg3 0.0 (*0.0)
           Epgk_adp 0.0 (*0.0)
           Epgk_atp 0.0 (*0.0)
           Epgk_pgp_adp 0.0 (*0.0)
           Epgk_pg3_atp 0.0 (*0.0)
         }
        }

pgp + Epgk <-> Epgk_pgp >@446.3745240747576 <@630124.2093590024
      (*446.3745240747576,*630124.2093590024) [-5.0,15.0]---[-5.0,15.0]
Epgk + adp <-> Epgk_adp >@1.6400932497744534E7 <@3025228.933135766 (*1.6400932497744534
      E7,*3025228.933135766) [-5.0,8.0]---[-5.0,15.0]
Epgk_pgp + adp <-> Epgk_pgp_adp >@263938.8174733853 <@1.000000046350354
      (*263938.8174733853,*1.000000046350354) [-5.0,8.0]---[-5.0,15.0]
pgp + Epgk_adp <-> Epgk_pgp_adp >@6.623092221632249E7 <@124854.27991871494
      (*6.623092221632249E7,*124854.27991871494) [-5.0,8.0]---[-5.0,15.0]
Epgk_pgp_adp <-> Epgk_pg3_atp >@4184301.836960252 <@1.0632990466729525E8
      (*4184301.836960252,*1.0632990466729525E8) [-5.0,15.0]---[-5.0,15.0]
Epgk_pg3_atp <-> Epgk_atp + pg3 >@1.7833214207442951E9 <@6.812725408320574E7
      (*1.7833214207442951E9,*6.812725408320574E7) [-5.0,15.0]---[-5.0,8.0]
Epgk_pg3_atp <-> Epgk_pg3 + atp >@58.53341319624397 <@577.0415993674669
      (*58.53341319624397,*577.0415993674669) [-5.0,15.0]---[-5.0,15.0]
Epgk_pg3 <-> Epgk + pg3 >@1217713.2083092756 <@1179765.6876562668
      (*1217713.2083092756,*1179765.6876562668) [-5.0,15.0]---[-5.0,8.0]
Epgk_atp <-> Epgk + atp >@1.2990066517530879E7 <@1.9908872271729697E7
      (*1.2990066517530879E7,*1.9908872271729697E7) [-5.0,15.0]---[-5.0,8.0]

}

//KA
```

```
vPGluMu: pg3 --- pg2 {

        modifiers{}

        enzymes{
        cell{
           Epglumu 0.001 (*0.001)
           Epglumu_pg2 0.0 (*0.0)
           Epglumu_pg3 0.0 (*0.0)
         }
        }

pg3 + Epglumu <-> Epglumu_pg3 >@5249464.910372856 <@3618684.170141491
     (*5249464.910372856,*3618684.170141491) [0.0,8.0]---[0.0,10.0]
Epglumu_pg3 <-> Epglumu_pg2 >@2.9532936322442837E7 <@1.0555290518934883E7
     (*2.9532936322442837E7,*1.0555290518934883E7) [0.0,10.0]---[0.0,10.0]
Epglumu_pg2 <-> pg2 + Epglumu >@121242.2385785785 <@2617561.4828406023
     (*121242.2385785785,*2617561.4828406023) [0.0,10.0]---[0.0,8.0]

}

//FIMGE
vENO: pg2 --- pep {

        modifiers{}

        enzymes{
        cell{
           Eeno 0.001 (*0.001)
           Eeno_pg2 0.0 (*0.0)
           Eeno_pep 0.0 (*0.0)
         }
        }

pg2 + Eeno <-> Eeno_pg2 >@3778819.7663712683 <@66975.9695620004
     (3778819.7663712683,66975.9695620004) [0.0,8.0]---[0.0,10.0]
Eeno_pg2 <-> Eeno_pep >@352640.20134127594 <@1.7956930428046398E9
     (352640.20134127594,1.7956930428046398E9) [0.0,10.0]---[0.0,10.0]
Eeno_pep <-> Eeno + pep >@2.481066748781883E10 <@4.081731715136534E7 (2.481066748781883
     E10,4.081731715136534E7) [0.0,10.0]---[0.0,8.0]

}

//KA
vSynth1: pep --- cho_mur {

        modifiers{}

        enzymes{
        cell{
           Esynth1 0.001 (*0.001)
           Esynth1_pep 0.0 (*0.0)
         }
        }

Esynth1 + pep <-> Esynth1_pep >@7441792.888847801 <@7441773.349495178
     (*7441792.888847801,*7441773.349495178) [0.0,8.0]---[0.0,10.0]
Esynth1_pep -> cho_mur + Esynth1 @19.538970029516282 (*19.538970029516282) [0.0,10.0]
```

```
}

//KA
vPEPCxylase: pep ---- oaa {

        modifiers{ fdp}

        enzymes{
        cell{
           Epepcxylase 5.927265718579291E-4 (*0.001)
           Epepcxylase_pep 4.072357221637869E-4 (*0.0)
           Epepcxylase_fdp 5.194095009151411E-13 (*0.0)
           Epepcxylase_fdp_fdp 1.499063483645509E-15 (*0.0)
           Epepcxylase_fdp_fdp_fdp 3.2221485567683185E-17 (*0.0)
           Epepcxylase_fdp_fdp_fdp_fdp 5.770941932453426E-14 (*0.0)
           Epepcxylase_fdp_pep 2.415668789751241E-22 (*0.0)
           Epepcxylase_fdp_fdp_pep 1.7992200207370836E-22 (*0.0)
           Epepcxylase_fdp_fdp_fdp_pep 4.1788456031994374E-21 (*0.0)
           Epepcxylase_fdp_fdp_fdp_fdp_pep 2.8769028127175685E-14 (*0.0)
         }
        }

Epepcxylase + fdp <-> Epepcxylase_fdp >@1.0001200873229572 <@3.385461445034598E8
      (1000000.0,8.03895069957E9) [0.0,10.0]---[0.0,10.0]
Epepcxylase + pep <-> Epepcxylase_pep >@7.034513127507317E8 <@2.766577164348992E9
      (1000000.0,4226984.62435) [0.0,10.0]---[0.0,10.0]
Epepcxylase_pep -> Epepcxylase + oaa @102.48310910948966 (104.09077487) [0.0,10.0]
Epepcxylase_fdp + fdp -> Epepcxylase_fdp_fdp @57.70358017293963 (1000000.0) [0.0,10.0]
Epepcxylase_fdp_fdp -> Epepcxylase_fdp + fdp @5930.807613027551 (1847545.40574)
      [0.0,10.0]
Epepcxylase_fdp + pep -> Epepcxylase_fdp_pep @1.5646115041832669 (1000000.0) [0.0,10.0]
Epepcxylase_fdp_pep -> Epepcxylase_fdp + pep @1065564.372591534 (4226984.62435)
      [0.0,10.0]
Epepcxylase_fdp_pep -> Epepcxylase_fdp + oaa @9.089256065011168E9 (15672.0733318)
      [0.0,10.0]
Epepcxylase_fdp_fdp + fdp <-> Epepcxylase_fdp_fdp_fdp >@92458.5023005776 <@1275976
      .0821785175 (1000000.0,37438.6832035) [0.0,10.0]---[0.0,10.0]
Epepcxylase_fdp_fdp + pep <-> Epepcxylase_fdp_fdp_pep >@8.948527937609011 <@2
      .014594182254282E8 (1000000.0,4226984.62435) [0.0,10.0]---[0.0,10.0]
Epepcxylase_fdp_fdp_pep -> Epepcxylase_fdp_fdp + oaa @0.9999166772881231
      (15672.0733318) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp + fdp -> Epepcxylase_fdp_fdp_fdp_fdp @5.931677056744425E9
      (1000000.0) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp_fdp -> Epepcxylase_fdp_fdp_fdp + fdp @982420.8432799311
      (61748.8549115) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp + pep -> Epepcxylase_fdp_fdp_fdp_pep @81192.44355732527
      (1000000.0) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp_pep -> Epepcxylase_fdp_fdp_fdp + pep @1.6916256088012602E9
      (4226984.62435) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp_pep -> Epepcxylase_fdp_fdp_fdp + oaa @3210.963950462683
      (15672.0733318) [0.0,10.0]
Epepcxylase_fdp_fdp_fdp_fdp + pep <-> Epepcxylase_fdp_fdp_fdp_fdp_pep >@1
      .1840591195063017E10 <@36.84597541032951 (1000000.0,4226984.62435)
      [0.0,10.0]---[0.0,10.0]
Epepcxylase_fdp_fdp_fdp_fdp_pep -> Epepcxylase_fdp_fdp_fdp_fdp + oaa @6
      .417932270710723E10 (15672.0733318) [0.0,10.0]

}
```

```
//KA
vSynth2: pyr ---- ile {

        modifiers{}

        enzymes{
        cell{
           Esynth2 0.001 (*0.001)
           Esynth2_pyr 0.0 (*0.0)
         }
        }

Esynth2 + pyr <-> Esynth2_pyr >@74.61855067344412 <@1.0000001255706157
     (*74.61855067344412,*1.0000001255706157) [0.0,8.0]---[0.0,10.0]
Esynth2_pyr -> Esynth2 + ile @73.61855054981174 (*73.61855054981174) [0.0,10.0]

}

//GlobalSim
vPDH: pyr ---- {

        modifiers{}

        enzymes{
        cell{
           Epdh 0.001 (*0.001)
           Epdh_pyr 0.0 (*0.0)
         }
        }

Epdh_pyr -> Epdh @31481.27786878084 (*31481.27786878084) [0.0,10.0]
pyr -> @rRateK0K1 [0.0,10.0]
 -> Epdh_pyr @rRateK0K1 [0.0,10.0]
Epdh -> @rRateK0K1 [0.0,10.0]

}



//Global
vG6PDH: g6p + nadp ---- pg + nadph {

        modifiers{}

        enzymes{
        cell{
           Eg6pdh 0.001 (*0.001)
           Eg6pdh_g6p 0.0 (*0.0)
           Eg6pdh_pg 0.0 (*0.0)
           Eg6pdh_nadp 0.0 (*0.0)
           Eg6pdh_nadph 0.0 (*0.0)
           Eg6pdh_pg_nadph 0.0 (*0.0)
           Eg6pdh_g6p_nadp 0.0 (*0.0)
           Eg6pdh_g6p_nadph 0.0 (*0.0)
           Eg6pdh_nadp_nadph 0.0 (*0.0)
           Eg6pdh_g6p_nadp_nadph 0.0 (*0.0)
           Eg6pdh_pg_nadph_nadph 0.0 (*0.0)
         }
```

```
        }

g6p + Eg6pdh <-> Eg6pdh_g6p >@34.84194085548704 <@4165.583133368833
      (*34.84194085548704,*4165.583133368833) [0.0,6.0]---[0.0,10.0]
Eg6pdh + nadp <-> Eg6pdh_nadp >@43368.36780670752 <@997.1022060995822
      (*43368.36780670752,*997.1022060995822) [0.0,6.0]---[0.0,10.0]
Eg6pdh_g6p + nadp <-> Eg6pdh_g6p_nadp >@178111.08743165756 <@1.583664949415936E8
      (*178111.08743165756,*1.583664949415936E8) [0.0,6.0]---[0.0,10.0]
g6p + Eg6pdh_nadp <-> Eg6pdh_g6p_nadp >@96.393133682726 <@5.03509187052355
      (*96.393133682726,*5.03509187052355) [0.0,6.0]---[0.0,10.0]
Eg6pdh_g6p_nadp -> Eg6pdh_pg_nadph @3.440492715874786E8 (*3.440492715874786E8) [0.0,10.0]
Eg6pdh_pg_nadph -> pg + Eg6pdh_nadph @1.5941588041036442E7 (*1.5941588041036442E7)
      [0.0,10.0]
Eg6pdh_pg_nadph -> Eg6pdh_pg + nadph @1.7484481498608302E7 (*1.7484481498608302E7)
      [0.0,10.0]
Eg6pdh_pg -> Eg6pdh + pg @67149.3983436132 (*67149.3983436132) [0.0,10.0]
Eg6pdh_nadph <-> Eg6pdh + nadph >@666.6139445700364 <@67022.53579131703
      (*666.6139445700364,*67022.53579131703) [0.0,10.0]---[0.0,6.0]
Eg6pdh_g6p + nadph <-> Eg6pdh_g6p_nadph >@85.23224971307718 <@97360.03886884112
      (*85.23224971307718,*97360.03886884112) [0.0,6.0]---[0.0,10.0]
Eg6pdh_nadp + nadph <-> Eg6pdh_nadp_nadph >@1642.4157646890612 <@4142.251600664168
      (*1642.4157646890612,*4142.251600664168) [0.0,6.0]---[0.0,10.0]
Eg6pdh_g6p_nadp + nadph <-> Eg6pdh_g6p_nadp_nadph >@55.602138398162 <@1
      .5245599969655123E8 (*55.602138398162,*1.5245599969655123E8) [0.0,6.0]---[0.0,10.0]
Eg6pdh_pg_nadph + nadph <-> Eg6pdh_pg_nadph_nadph >@93.95001537782768 <@4034
      .014017606999 (*93.95001537782768,*4034.014017606999) [0.0,6.0]---[0.0,10.0]

}

//Global
vPGDH: pg + nadp --- ribu5p + nadph {

        modifiers{ atp}

        enzymes{
        cell{
           Epgdh 3.628697150773524E-5 (*3.628697150773524E-5)
           Epgdh_pg 2.6475669932371512E-8 (*2.6475669932371512E-8)
           Epgdh_nadp 1.318939490675069E-4 (*1.318939490675069E-4)
           Epgdh_nadph 8.116341916435976E-4 (*8.116341916435976E-4)
           Epgdh_ribu5p 1.6069063510144794E-5 (*1.6069063510144794E-5)
           Epgdh_atp 9.702697046716019E-7 (*9.702697046716019E-7)
           Epgdh_pg_nadp 3.109478743095457E-6 (*3.109478743095457E-6)
           Epgdh_pg_atp 3.118740445600252E-11 (*3.118740445600252E-11)
           Epgdh_ribu5p_nadph 1.033467924688673E-8 (*1.033467924688673E-8)
         }
        }

pg + Epgdh <-> Epgdh_pg >@1063681.9756333109 <@1.1251267594407966E9
      (*1063681.9756333109,*1.1251267594407966E9) [0.0,8.0]---[0.0,10.0]
Epgdh + nadp <-> Epgdh_nadp >@1039254.0371311974 <@49544.14744241454
      (*1039254.0371311974,*49544.14744241454) [0.0,8.0]---[0.0,10.0]
Epgdh_pg + nadp <-> Epgdh_pg_nadp >@1064917.9650837632 <@220316.03471341837
      (*1064917.9650837632,*220316.03471341837) [0.0,8.0]---[0.0,10.0]
pg + Epgdh_nadp <-> Epgdh_pg_nadp >@1037408.1894653962 <@3.2922223299665887E7
      (*1037408.1894653962,*3.2922223299665887E7) [0.0,8.0]---[0.0,10.0]
Epgdh_pg_nadp -> Epgdh_ribu5p_nadph @44887.28029799544 (*44887.28029799544) [0.0,10.0] :
      slow
Epgdh_ribu5p_nadph -> Epgdh_nadph + ribu5p @8.8246510777885E9 (*8.8246510777885E9)
```

[0.0,10.0]
Epgdh_nadph + ribu5p -> Epgdh_ribu5p_nadph @1022763.4941811053 (*1022763.4941811053) [0.0,8.0]
Epgdh_ribu5p_nadph -> Epgdh_ribu5p + nadph @2.847789832952994E8 (*2.847789832952994E8) [0.0,10.0]
Epgdh_ribu5p + nadph -> Epgdh_ribu5p_nadph @987244.6503765623 (*987244.6503765623) [0.0,8.0]
Epgdh_nadph -> Epgdh + nadph @476.6652484421763 (*476.6652484421763) [0.0,10.0]
Epgdh + nadph -> Epgdh_nadph @980902.6683718587 (*980902.6683718587) [0.0,8.0]
Epgdh_ribu5p -> Epgdh + ribu5p @372993.4928825547 (*372993.4928825547) [0.0,10.0]
Epgdh + ribu5p -> Epgdh_ribu5p @992108.0676651685 (*992108.0676651685) [0.0,8.0]
Epgdh + atp -> Epgdh_atp @1013166.5928799342 (*1013166.5928799342) [0.0,8.0]
Epgdh_atp -> Epgdh + atp @1.617957050145157E8 (*1.617957050145157E8) [0.0,10.0]
Epgdh_pg + atp -> Epgdh_pg_atp @1049547.791053956 (*1049547.791053956) [0.0,8.0]
Epgdh_pg_atp -> Epgdh_pg + atp @3.804502025348008E9 (*3.804502025348008E9) [0.0,10.0]

}

//KA
vR5PI: ribu5p --- rib5p {

modifiers{}

enzymes{
cell{
Er5p1 0.001 (*0.001)
Er5p1_ribu5p 0.0 (*0.0)
Er5p1_rib5p 0.0 (*0.0)
}
}

Er5p1 + ribu5p <-> Er5p1_ribu5p >@23477.483275094033 <@6.785446906927375E9 (*23477.483275094033,*6.785446906927375E9) [0.0,8.0]---[0.0,10.0]
Er5p1_ribu5p <-> Er5p1_rib5p >@2.1452692217704463E9 <@1.0138583480692043E9 (*2.1452692217704463E9,*1.0138583480692043E9) [0.0,10.0]---[0.0,10.0]
Er5p1_rib5p <-> Er5p1 + rib5p >@4.652498351132615E9 <@8515.360250531308 (*4.652498351132615E9,*8515.360250531308) [0.0,10.0]---[0.0,8.0]

}

//KA
vRU5P: ribu5p --- xyl5p {

modifiers{}

enzymes{
cell{
Eru5p 0.001 (*0.001)
Eru5p_ribu5p 0.0 (*0.0)
Eru5p_xyl5p 0.0 (*0.0)
}
}

Eru5p + ribu5p <-> Eru5p_ribu5p >@40048.7502383563 <@1.0E10 (*40048.7502383563,*1.0E10) [0.0,10.0]---[0.0,10.0]
Eru5p_ribu5p <-> Eru5p_xyl5p >@2.3704803565611067E9 <@9.266350317530496E8 (*2.3704803565611067E9,*9.266350317530496E8) [0.0,10.0]---[0.0,10.0]

```
Eru5p_xyl5p <-> Eru5p + xyl5p >@5.3982791457319 06E9 <@39504.324854236096
    (*5.398279145731906E9,*39504.324854236096) [0.0,10.0]---[0.0,10.0]

}

//KA
vPPK: rib5p --- nucleotide {

        modifiers{}

        enzymes{
        cell{
           Erppk 0.001 (*0.001)
           Erppk_rib5p 0.0 (*0.0)
         }
        }

Erppk + rib5p <-> Erppk_rib5p >@1708.7585245258308 <@157.97540064790883
    (*1708.7585245258308,*157.97540064790883) [0.0,8.0]---[0.0,10.0]
Erppk_rib5p -> nucleotide + Erppk @12.900452250930224 (*12.900452250930224) [0.0,10.0]

}

//KA
vTKA: xyl5p + rib5p --- sed7p + gap {

        modifiers{}

        enzymes{
        cell{
           Etka 0.001 (*0.001)
           Etka_rib5p 0.0 (*0.0)
           Etka_rib5p_xyl5p 0.0 (*0.0)
           Etka_sed7p_gap 0.0 (*0.0)
           Etka_gap 0.0 (*0.0)
         }
        }

Etka + rib5p <-> Etka_rib5p >@9793907.7911523 <@1.0E10 (*9793907.7911523,*1.0E10)
    [0.0,8.0]---[0.0,10.0]
xyl5p + Etka_rib5p <-> Etka_rib5p_xyl5p >@2.937990797469441E7 <@1.0E10
    (*2.937990797469441E7,*1.0E10) [0.0,8.0]---[0.0,10.0]
Etka_rib5p_xyl5p <-> Etka_sed7p_gap >@1.0E10 <@1.0E10 (*1.0E10,*1.0E10)
    [0.0,10.0]---[0.0,10.0]
sed7p + Etka_gap <-> Etka_sed7p_gap >@2.680863603910198E7 <@1.0E10 (*2.680863603910198
    E7,*1.0E10) [0.0,8.0]---[0.0,10.0]
Etka + gap <-> Etka_gap >@8937047.406742867 <@1.0E10 (*8937047.406742867,*1.0E10)
    [0.0,8.0]---[0.0,10.0]

}

//KA
vTA: gap + sed7p --- f6p + e4p {

        modifiers{}

        enzymes{
        cell{
           Eta 4.3242434232992595E-4 (*0.001)
```

```
            Eta_gap 3.018021191721868E-4 (*0.0)
            Eta_gap_sed7p 6.189942785504624E-6 (*0.0)
            Eta_e4p_f6p 8.045588919060367E-9 (*0.0)
            Eta_f6p 2.5957875753861674E-4 (*0.0)
          }
        }

gap + Eta <-> Eta_gap >@7793069.098309249 <@2473367.893118087 (1000000.0,3.47783815678E8
    ) [0.0,8.0]---[0.0,10.0]
Eta_gap + sed7p <-> Eta_gap_sed7p >@28539.602545734106 <@366450.3941456761
    (8988365.11673,4.76357491187E8) [0.0,8.0]---[0.0,10.0]
Eta_gap_sed7p <-> Eta_e4p_f6p >@2395930.591911783 <@1.8376878360119967E9
    (3.47038005266E9,3.76094610236E9) [0.0,10.0]---[0.0,10.0]
Eta_f6p + e4p <-> Eta_e4p_f6p >@293745.15677329473 <@9.52086887221295E8
    (8985104.78013,4.49900668219E8) [0.0,8.0]---[0.0,10.0]
f6p + Eta <-> Eta_f6p >@27565.59206641448 <@27690.87907295803 (1000000.0,4.1921580168E8)
    [0.0,8.0]---[0.0,10.0]

}

//KA
vTKB: xyl5p + e4p --- f6p + gap {

        modifiers{}

        enzymes{
        cell{
            Etkb 0.001 (*0.001)
            Etkb_xyl5p 0.0 (*0.0)
            Etkb_xyl5p_e4p 0.0 (*0.0)
            Etkb_f6p_gap 0.0 (*0.0)
            Etkb_gap 0.0 (*0.0)
          }
        }

Etkb + xyl5p <-> Etkb_xyl5p >@2.9767315389565293E7 <@1.0E10 (*2.9767315389565293E7,*1.0
    E10) [0.0,8.0]---[0.0,10.0]
e4p + Etkb_xyl5p <-> Etkb_xyl5p_e4p >@6.5616411795814976E7 <@1.0E10
    (*6.5616411795814976E7,*1.0E10) [0.0,8.0]---[0.0,10.0]
Etkb_xyl5p_e4p <-> Etkb_f6p_gap >@1.0E10 <@2.0457215340563703E9 (*1.0E10
    ,*2.0457215340563703E9) [0.0,10.0]---[0.0,10.0]
f6p + Etkb_gap <-> Etkb_f6p_gap >@1.0E8 <@1.0E10 (*1.0E8,*1.0E10) [0.0,8.0]---[0.0,10.0]
Etkb + gap <-> Etkb_gap >@9547837.252848232 <@1.0E10 (*9547837.252848232,*1.0E10)
    [0.0,8.0]---[0.0,10.0]

}

//KA
vDHAPS: pep + e4p --- {

        modifiers{}

        enzymes{
        cell{
            Edhaps 5.706982565293556E-4 (*5.706982565293556E-4)
            Edhaps_e4p 6.034147930686969E-8 (*6.034147930686969E-8)
            Edhaps_e4p_e4p 1.798490644529588E-10 (*1.798490644529588E-10)
            Edhaps_e4p_pep 7.801714016496681E-7 (*7.801714016496681E-7)
            Edhaps_e4p_pep_pep 5.699029024039884E-6 (*5.699029024039884E-6)
```

```
            Edhaps_e4p_e4p_pep 2.1387974019449E-5 (*2.1387974019449E-5)
            Edhaps_e4p_e4p_pep_pep 1.9507211839928607E-9 (*1.9507211839928607E-9)
            Edhaps_pep 4.0071811539120337E-4 (*4.0071811539120337E-4)
            Edhaps_pep_pep 6.580972180827027E-7 (*6.580972180827027E-7)
          }
        }

e4p + Edhaps <-> Edhaps_e4p >@1245539.8540814104 <@1.1793782500178425E9
      (*1245539.8540814104,*1.1793782500178425E9) [0.0,8.0]---[0.0,10.0]
pep + Edhaps <-> Edhaps_pep >@70510.58066931683 <@270477.47414940386
      (*70510.58066931683,*270477.47414940386) [0.0,8.0]---[0.0,10.0]
e4p + Edhaps_pep <-> Edhaps_e4p_pep >@2.9742211875658955E7 <@1.5298302585839407E9
      (*2.9742211875658955E7,*1.5298302585839407E9) [0.0,8.0]---[0.0,10.0]
pep + Edhaps_pep <-> Edhaps_pep_pep >@246.9367276949254 <@64091.75349973866
      (*246.9367276949254,*64091.75349973866) [0.0,8.0]---[0.0,10.0]
e4p + Edhaps_pep_pep <-> Edhaps_e4p_pep_pep >@3410505.1379636535 <@68.7018891811367
      (*3410505.1379636535,*68.7018891811367) [0.0,8.0]---[0.0,10.0]
e4p + Edhaps_e4p <-> Edhaps_e4p_e4p >@1783913.2692835033 <@3.0253854056846093E7
      (*1783913.2692835033,*3.0253854056846093E7) [0.0,8.0]---[0.0,10.0]
pep + Edhaps_e4p <-> Edhaps_e4p_pep >@3.2714152634360576 <@1.2994929103630992
      (*3.2714152634360576,*1.2994929103630992) [0.0,8.0]---[0.0,10.0]
e4p + Edhaps_e4p_pep <-> Edhaps_e4p_e4p_pep >@42273.24450904922 <@62.962921371613675
      (*42273.24450904922,*62.962921371613675) [0.0,8.0]---[0.0,10.0]
pep + Edhaps_e4p_pep <-> Edhaps_e4p_pep_pep >@4.669797239747408E7 <@1
      .7257063452013325E7 (*4.669797239747408E7,*1.7257063452013325E7) [0.0,8.0]---[0.0,10.0]
e4p + Edhaps_e4p_pep_pep <-> Edhaps_e4p_e4p_pep_pep >@75.20751388577412 <@34407
      .23795785133 (*75.20751388577412,*34407.23795785133) [0.0,8.0]---[0.0,10.0]
pep + Edhaps_e4p_e4p <-> Edhaps_e4p_e4p_pep >@1.122550000844095E7 <@4.71910392297437
      (*1.122550000844095E7,*4.71910392297437) [0.0,8.0]---[0.0,10.0]
pep + Edhaps_e4p_e4p_pep <-> Edhaps_e4p_e4p_pep_pep >@158.3754986143872 <@938675
      .0600673703 (*158.3754986143872,*938675.0600673703) [0.0,10.0]---[0.0,8.0]
Edhaps_e4p_e4p_pep_pep -> Edhaps_e4p_pep @3725732.96976031 (*3725732.96976031)
      [0.0,10.0] :slow

}

//GlobalSim
vPFK: f6p + atp --- fdp + adp {

        modifiers{ pep amp}

        enzymes{
        cell{
            Epfk 4.12061534668244E-5 [-5.0,-1.0]
            Epfk_atp 0.029901686306988254 (*0.0)
            Epfk_atp_f6p 0.0041121405578154946 (*0.0)
            Epfk_adp 1.2694279008974528E-5 (*0.0)
          }
        }

Epfk_atp -> Epfk + atp @0.8136892613257293 [0.0,10.0]
Epfk_atp_f6p -> Epfk_atp + f6p @3.1400096953009937 [0.0,10.0]
Epfk_atp_f6p -> fdp + Epfk_adp @34.049272194152344 [0.0,10.0] :slow
Epfk_adp -> Epfk + adp @11029.802720949023 [0.0,10.0] :slow
Epfk + atp -> Epfk_atp @rPfk0 (*rPfk0) [0.0,10.0]
Epfk_atp + f6p -> Epfk_atp_f6p @rPfk2 (*rPfk2) [0.0,10.0]

}
```

```
//GlobalSim
vPK: pep + adp ---> pyr + atp {

        modifiers{ fdp}

        enzymes{
        cell{
           Epk 2.9475043117849926E-5 (*2.9475043117849926E-5)
           Epk_pep 1.9890263116954396E-5 (*1.9890263116954396E-5)
           Epk_pep_adp 9.064711832110233E-4 (*9.064711832110233E-4)
           Epk_atp 1.396167514575113E-9 (*1.396167514575113E-9)
         }
        }

Epk_pep -> Epk + pep @2.955353679650043E9 (*2.955353679650043E9) [0.0,10.0]
Epk_pep + adp -> Epk_pep_adp @672761.498098979 (*672761.498098979) [0.0,7.0]
Epk_pep_adp -> Epk_pep + adp @8595.000879468533 (*8595.000879468533) [0.0,10.0]
Epk_pep_adp -> pyr + Epk_atp @40.81791333168355 (*40.81791333168355) [0.0,10.0] :slow
Epk_atp -> Epk + atp @2.6501305758597497E7 (*2.6501305758597497E7) [0.0,10.0] :slow
Epk + pep -> Epk_pep @rPk0 (*rpK0) [0.0,7.0]

}
```

APPENDIX B

DYNAMIC FLUX BALANCE ANALYSIS EXTENSION

The DFBA model file utilized in **Chapter 4**:

```
LpSolver: Gurobi
OdeSolver: CVODE

MethodType{
        PFBA
   MethodConfig {
        }
}

Simulation{
  initialTime: 0.0
  finalTime: 15.0
  stepSize: 0.125
}

SbmlFile:"EcoliCoreModel.xml"

FluxBalanceAnalysis{
        ObjectiveFunction{
                Max R_Biomass_Ecoli_core_w_GAM
        }
}

Fluxes{
R_EX_glc_e qsval
R_EX_ac_e qacval
R_EX_o2_e qosval
}

FreeVariables{
f
}

AlgebricRules {}

Functions {

        def glucoseKinetics(S,I,qsmaxglc,kiglc,ksglc):
           qsmaxglc*(1.0/(1.0+(I/kiglc)))*(S/(S+ksglc))
        end

        def glucoseKinetics2(S,qsmaxglc,ksglc):
           (qsmaxglc*S)/(S+ksglc)
        end


        def o2Kinetics(qs,qm,cx,cs,yxs,yos):
```

```
        (qs-(qs-qm)*yxs*(cx/cs))*yos
    end

    def o2Kinetics2(S,qsmaxo2,kso2):
        (qsmaxo2*O2)/(O2+kso2)
    end

    def o2Kinetics2Backup(S,I,qsmaxo2,kio2,kso2):
        qsmaxo2*(1.0/(1.0+(I/kio2)))*(S/(S+kso2))
    end

    def o2Kinetics3(qsmaxo2):
        qsmaxo2
    end

    def acKinetics(S,qsmaxac,ksac):
            qsmaxac*(S/(S+ksac))
    end
}


ODESystem{

    val kla=7.5
    val o2Eq=0.21
    val sin = 277.0

    val qsmax = 4.10541652735493
    val qski = 999.99450445301
    val qsks = 0.0010005069647388664
    val qsITerm = (1.0/(1.0+(Ac/qski)))
    val qsGlcOnly = (qsmax*Glc)*(1.0/(qsks+Glc))
    val currentqsval = -1.0*glucoseKinetics(Glc,Ac,qsmax,qski,qsks)

    val qsmaxo2 = 5.807948704120516
    val qosval = -1.0*o2Kinetics3(qsmaxo2)

    val qacmax = 4.215873755359025
    val qacks = 1.2614925738174751

    val fin = if V > 20.0 then 0.0 else f end
    val fluxTerm = (fin/V)*(sin)*(1.0/X)
    val fluxTermQs = fluxTerm+qsGlcOnly


    val qacval = -1.0*acKinetics(Ac,qacmax,qacks)

    val qsval = if (Glc = 0.0) then
                    if (qsmax >= fluxTerm) then
                        -1.0*fluxTerm*qsITerm
                    else
                        -1.0*qsmax*qsITerm
                    end
                else
                    if (qsmax >= fluxTermQs) then
                        -1.0*fluxTerm*qsITerm+currentqsval
                    else
                        -1.0*qsmax*qsITerm
                    end
```

```
                end

        val fluxDifference = fluxTerm+qsval
        val extraGlcFlux = if fluxDifference > 0.0 then fluxDifference else 0.0 end
        val addExtraGlc = if extraGlcFlux > 0.0 then extraGlcFlux*X else 0.0 end

        X:0.2:R_Biomass_Ecoli_core_w_GAM = R_Biomass_Ecoli_core_w_GAM*X-(fin/V)*X
        Glc:0.0:R_EX_glc_e = R_EX_glc_e*X-(fin/V)*Glc+addExtraGlc
        Ac:0.0:R_EX_ac_e = R_EX_ac_e*X-(fin/V)*Ac
        O2:0.21:R_EX_o2_e = 0.0
        xEth:0.0:R_EX_etoh_e = R_EX_etoh_e*X-(fin/V)*xEth
        Form:0.0:R_EX_for_e = R_EX_for_e*X-(fin/V)*Form
        Lac:0.0:R_EX_lac_D_e = R_EX_lac_D_e*X-(fin/V)*Lac
        V:1.0 = fin


        substrate{
           Glc
        }

        product{
           Ac
        }
}

DFBAVariables{
        biomassMetabolite: X
        volume: V
}
```