Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Biotechnology and Food Science

Christoffer Gretarsson Alexandersen

Acknowledging the uncertainty of enzyme kinetic parameters in constraint-based metabolic modeling

Master's thesis in Biotechnology (5 year)

Supervisor: Eivind Almaas

May 2020

NTNU
Kunnskap for en bedre verden

Christoffer Gretarsson Alexandersen

Acknowledging the uncertainty of enzyme kinetic parameters in constraint-based metabolic modeling

Master's thesis in Biotechnology (5 year)
Supervisor: Eivind Almaas
May 2020

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Biotechnology and Food Science

NTNU
Kunnskap for en bedre verden

I would like to thank my supervisor Prof. Eivind Almaas for introducing me to the exciting world of systems biology and for enabling me to pursue my research interests with an exceptionally engaging thesis. I would also like to give a warm thanks to Prof. Allen Holder for his kind hospitality and invaluable mentoring during my stay in Terre Haute. Finally, I would also like to thank my co-supervisor Pål Røynestad for his readily attentive guidance.

Sammendrag

Begrensningsbaserte analysemetoder, slik som ”flux balance analysis” (FBA), har vist seg å være nyttige og allsidige beregningsverktøy for å studere og tilvirke cellemetabolisme. Med stadig nye anvendelser innenfor medisin og industri har FBA blitt en fanebærer for systembiologisk forskning. Til tross for stor suksess har studier vist at FBA-metoden kan feile grovt under visse omstendigheter. Dessuten viser beregningseksperimenter at metoden er ustabil i møte med små endringer i biomassekomposisjonen, hvilket motsier cellemetabolismens standheftige natur. En annen begrensningsbasert modelleringsmetode, ”robust analysis of metabolic pathways” (RAMP), forsøker å rette opp FBA-metodens ustabilitet ved å anse biomassekomposisjonen som usikker. Som beregningsverktøy presterer RAMP på linje med FBA og innehaver dessuten ønskede matematiske egenskaper. De siste årene har det blitt utviklet en rekke utvidelser av FBA-paradigmet. Noen av disse utvidelsene innebygger massebegrensninger på mengden enzym tilgjengelig for cellen og tar dermed i bruk enzymkinetisk data ladet med både biologisk og eksperimentell usikkerhet. I denne tesen utarbeides ”**R**obust **A**nalysis of **M**etabolic **P**athways under **E**nzymatic **R**egulation” (RAMPER); en utvidelse og viderutvikling av RAMP som inkorporerer enzymkinetiske parametre. Vi diskuterer stokastiske tolkninger av metoden, introduserer varianter av metodeformuleringen og viser at RAMPER arver RAMP’s viktigste matematiske egenskaper. Vi implementerer så metoden slik at stokastiske og ustokastiske simuleringer kan utføres både med og uten enzymkinetiske parametre. Deretter sammenligner vi implementasjonens beregningshastighet under forskjellige modelleringsantagelser. Videre formulerer vi en provisorisk modell for usikkerhet i kinetiske parametre ved å granske en omfattende enzymdatabase. Følgende analyserer vi betydningen av enzymkinetisk usikkerhet for metodens beregningsresultater og demonstrerer slik at RAMPER gjør rede for enzymkarakteristikker forbigåtte av tidligere modelleringsmetoder. Med dette argumenterer vi for at RAMP formalismen, i likhet med FBA, er et utbyggbart og ressurssterkt rammeverk for begrensningsbasert analyse av metabolske nettverk.

Summary

Constraint-based approaches, such as flux balance analysis (FBA), have proven proficient and versatile computational tools for the study and engineering of cellular metabolism. Growing ever more prevalent in medicine and industry, FBA has become a flag-bearer of the systems biology research field. Yet despite the successes, FBA has been shown to suffer substantial shortcomings in its predictive power. Additionally, computational experiments demonstrate that FBA is sensitive to small parametric perturbations in the biomass composition, contradicting the robust nature of cell metabolism. Another constraint-based approach, robust analysis of metabolic pathways (RAMP), intends to mend the parametric instability in FBA by inherently modeling uncertainty in the biomass composition. RAMP rivals FBA in its computational predictions and possesses desirable mathematical properties. Over the last years, additional extensions to the FBA paradigm have introduced mass constraints on the amount of metabolic enzyme available to the cell. These extensions rely on enzyme kinetic parameters that are fraught with experimental and biological uncertainty. Herein we present **R**obust **A**nalysis of **M**etabolic **P**athways under **E**nzymatic **R**egulation (RAMPER); an extension and further development of RAMP that incorporates enzyme kinetic constraints. We discuss novel stochastic interpretations of the method, introduce separate yet equivalent method formulations, and prove that RAMPER inherits RAMP's most essential mathematical properties. We then implement the method computationally, allowing varying degrees of parametric uncertainty and the noncompulsory inclusion of enzyme kinetic constraints. We furthermore evaluate the computational speed of the implementation under distinct modeling assumptions. Moreover, by investigating kinetic data extracted from a comprehensive enzyme repository, we formulate a provisional model for kinetic parameter uncertainty. Thereafter, we analyze the impact of enzyme kinetic uncertainty on modeling predictions and subsequently demonstrate that RAMPER recognizes enzyme characteristics beyond the reach of earlier methods. We thereby argue that the RAMP formalism, in similar fashion to FBA, is an extensible and resourceful framework for constraint-based metabolic network analysis.

Table of Contents

List of Tables

List of Figures

Abbreviations

E. coli	=	*Escherichia coli*
EC	=	Enzyme commission
ecFBA	=	Enzymatically constrained FBA
ET	=	Effective turnover
FBA	=	Flux balance analysis
GAM	=	Growth-associated maintenance
GEM	=	Genome-scale metabolic model
GPR	=	Gene-protein-reaction
LP	=	Linear program
MM	=	Michaelis-Menten
NGAM	=	Nongrowth-associated maintenance
QCP	=	Quadratically constrained program
RAMP	=	Robust analysis of metabolic pathways
RAMPER	=	RAMP under enzymatic regulation
rET	=	Random effective turnover
S. cerevisiae	=	*Saccharomyces cerevisiae*
SOCP	=	Second-order cone program
sRAMPER	=	Substituted RAMP
SS	=	Steady state
TN	=	Turnover number

Chapter 1

Introduction

During the last half of the past century, there has been a monumental expansion in our understanding of the molecular basis of life. An immensely complex and intertwined biomolecular world has been revealed to be the underpinning of all cellular processes. We have gotten so far by examining, to greater and greater detail, the constituents of life. Animals—or at least most of them—are composed of organs, which are composed of tissues, which are composed of cells, which are composed of organelles, which are composed of molecules. The idea is that if one understands the components of a structure, then one will gain an understanding of the structure itself. This approach to understanding complex phenomena, by investigating its smaller constituents, has been dubbed *reductionism* [1, 2]. As early as the mid-19th century, notable biologist Ludwig Van Bertalanffy argued that reductionism, though utterly essential, is not enough by itself to understand most biological phenomena [3].

Many biological systems consist of thousands of constituents cross-interacting in remarkably convoluted yet purposeful patterns. And although an understanding of system constituents has been successful in predicting system properties of larger phenomena in other sciences, such as phase transition in statistical mechanics, this has been a challenge in biology, as constituents are rarely similar and are mixed highly heterogeneously. It has been argued that it does not suffice to understand each constituent separately; one must understand how the constituents interact and what emergent properties arise from these interactions [1, 3–7]. *Holism*, in contrast to reductionism, is a term used to describe a research approach in which phenomena are investigated at a system level as opposed to investigating parts of the phenomena in isolation [2, 4]. A shift from a reductionistic towards a more holistic investigatory approach has historically been limited by our ability to experimentally measure the characteristics of biological systems [6]. Yet in recent times, this has changed.

Over the last few decades, the wealth of available biological information has grown immensely upon the emergence of high-throughput measurement technologies. The arrival of genomics, transcriptomics, translatomics, proteomics, and metabolomics has facilitated the accumulation of data describing the constituents of diverse biological systems [8].

Concurrently, the world of research has witnessed the advent of computer science along with progressively sophisticated numerical methods for mathematical modeling. The untapped potential in addressing and probing such high-throughput biological data with state-of-the-art computational methods was a major driving force in the arrival of *systems biology*: the field of study of biological systems [5, 6, 8, 9].

The shift of focus from a reductionistic towards a holistic approach is often attributed as a defining characteristic of systems biology research; biological systems are to be studied in their own right and not presumed to be mere sums of their parts [2, 6]. More specifically, the scientific approaches covered in systems biology research are characterized by computational methodologies applied to high-throughput experimental data [5, 9] and are thus interdisciplinary in nature [10]. The biological systems examined may operate on disparate time scales [5] and can be as diverse as the Krebs cycle [11] and the global vegetative ecosystem [12]. Systems biology methodologies include both the formulation of bottom-up mechanistic mathematical models and bottom-down statistical data analysis [13]. Advancements in the modeling and analysis of complex biological systems have inspired the use of systems biology approaches in a variety of fields, including synthetic biology [14–16], cancer biology [17–19], precision medicine [20–23], and ecology [24, 25].

Among the most well-known and widely used models in computational systems biology belong to the class of *genome-scale metabolic models* (GEMs) [26]. Such models aim to describe cellular metabolism by digitally storing metabolic reactions and their gene-protein-reaction (GPR) associations. In other words, GEMs provide organism-specific computational reconstructions of cell metabolism that couple metabolic reactions with their respective genes [27]. The idea behind their construction is simple. First, genes are identified by probing the genome of the organism in question. Second, the identified genes are annotated with their corresponding enzymes alongside the encoding enzymes' biochemical reactions. Third, the metabolically relevant reactions are compiled together with their GPR associations effectively establishing the metabolic model [27]. These metabolic reconstructions do not only provide repositories of organism-specific cell metabolism but also platforms on which to study metabolism using computational methodologies.

Several methods for analyzing genome-scale metabolic reconstructions have been developed, such as extreme pathway analysis [28], elementary-mode analysis [29], and the vastly popular *flux balance analysis* (FBA) [30]. By employing tools of mathematical optimization, FBA has proved itself to be a particularly fruitful framework for the modeling of cellular metabolism. Accordingly, FBA has been shown to have numerous applications in bioengineering and medicine, as evidenced by its usage in lipid production by genetically engineered photosynthetic bacteria [31], hydrogen production by engineered *Escherichia coli* [32], antimicrobial drug target identification [33], and therapeutic target identification in cancer [34].

Moreover, numerous extensions to the FBA formulation have been published allowing for additional analytic features in addition to greater predictive power. Examples include dFBA, which was developed to study change in flux distributions as well as performing objective function sensitivity analysis [35]; cFBA [36], which extends FBA to microbial communities; ccFBA [37], which improves model accuracy by imposing mass constraints; and GECKO [38], MOMENT [39], sMOMENT [40], and FBAwMC [41], which all play

on the same theme of incorporating enzyme kinetic data to enhance predictive power. Additionally, FBA has been integrated into larger mathematical models that describe other facets of cell biology besides metabolism [42, 43].

Despite its success and promise, the FBA paradigm is based on questionable assumptions and is—in certain aspects—severely lacking in predictive power [44]. The FBA approach imposes the strict assumption of nonchanging metabolite concentrations and readily uses experimentally inferred parameters to which the method is highly sensitive. Attempts at ameliorating the strict assumptions made by FBA include the *robust analysis of metabolic pathways* (RAMP) formulation, which relaxes the FBA modeling assumptions and inherently acknowledges parameter uncertainty [45]. RAMP was shown to exhibit similar, and in some cases, better predictive power than FBA. Additionally, RAMP was shown to possess desirable mathematical properties absent in FBA. On the other hand, RAMP was found to suffer from computational instability not seen in FBA.

With a plethora of extensions and applications to the FBA paradigm emerging with rapid pace, it is imperative that its underlying framework is theoretically sound. **The aim of this thesis is to develop an extension to the RAMP approach that incorporates enzymatic constraints acknowledging the considerable parametric uncertainty embedded in enzyme kinetic data.** We thereby argue that it is quite possible to extend the RAMP method analogously to the development of FBA-derived methods and that it is achievable to improve upon the problematic backbone of FBA without losing its versatility and extensibility.

Chapter 2

Background and Theory

In this chapter, we will introduce the field of mathematical optimization, the reconstruction procedure of metabolic models, and the application of optimization to the analysis of metabolic reconstructions. First, we show that simple optimization problems called *linear programs* (LPs) can be generalized into problems handling uncertainty in modeling parameters called *second-order cone programs* (SOCPs). Second, we delineate the process of GEM reconstruction, noting especially the dependence on experimentally inferred parameters. Third, we introduce FBA—an LP formulation of great use in modeling cellular metabolism—and show how we may generalize FBA using an SOCP formulation that acknowledges parameter uncertainty. Finally, we describe how extensions made to the FBA framework increase predictive power but introduce many more uncertain parameters.

2.1 Mathematical optimization

Optimization occurs all around us. Manufacturers optimize operation procedures of production processes. Investors seek to maximize their profits carefully weighing in the risks of their investments. Physical systems aim to minimize potential energy. Light rays follow paths minimizing their travel time. And over eons, evolution maximizes species' reproduction capability. Often, such optimization processes occur within given restraints. A manufacturer may be tasked to find an optimal production procedure given a certain budget. Will three effective machines, or five less effective machines be more cost-effective? During the modeling process, we aim to identify a mathematical formulation that may aid us in finding the optimal solution to a given problem. But as there is no universal algorithm for solving optimization problems, we best thread carefully. A simple problem may not capture reality, whereas a highly descriptive model may prove unsolvable.

The field of mathematical optimization deals with the identification of maxima and minima—collectively known as *extrema*—of functions. And although optimization theory has been established for more general settings, we will restrict ourselves to functions mapping real vectors to real numbers. The search for extrema is formulated as so-called *optimization problems*, alternatively called *optimization programs*. In mathematical op-

timization we do not only consider algorithms for identifying extrema; we also seek to discover mathematical theorems concerning the optimization problems themselves. Optimization problems arise in a variety of fields, including finance, physics, chemistry, biology, artificial intelligence, and many more [46].

In general, an optimization problem may be described using the following notation

$$\min_{x} \; f(x) \tag{2.1a}$$

$$\text{subject to } \; c_i(x) = 0, \; \forall i \in \mathcal{E}, \tag{2.1b}$$

$$c_i(x) \leq 0, \; \forall i \in \mathcal{I}, \tag{2.1c}$$

where x is a real vector, f and c_i are functions of x, with $\mathcal{E}$ and $\mathcal{I}$ being index sets. We refer to the function over which we are optimizing—in this case f—as the *objective function*, whereas the input of the objective function—in this case x—is referred to as the *decision variable*, or simply *variable*. The functions that define the permitted values of the decision variables—in this case c_i for all i in $\mathcal{E} \cup \mathcal{I}$—are referred to as the *constraints functions*, or simply *constraints*. The constraints define the *solution space*—also called *feasible region*—which is the set of decision variables defined by the constraint functions. Hence, the preceding optimization problem describes the task of identifying a decision variable in the solution space corresponding to the minimum objective function value permissible in that same solution space. Note that the corresponding maximization problem is also captured in our preceding formulation by replacing f by $-f$ [46]. For simplicity's sake, we will only refer to minimization problems for the remainder of this section.

As an introductory example, consider the following optimization program,

$$\min_{x} \; 2x$$
$$\text{s. t. } \; 1 \leq x \leq 2,$$

in which $x \in \mathcal{R}$ and "s.t." is an abbreviation for "subject to". The solution to this problem is clearly found at $x = 1$ resulting in a minima of value 2. In this example we have not followed the notation as put forth in problem (2.1), which is in fact rarely followed in practical applications. This is by no means problematic, as we may reformulate our problem into the form found in (2.1) quite trivially. It is on the other hand very useful to agree on an explicit formulation while considering theoretical aspects of optimization programs.

Points that solve an optimization problem are called *globally optimal points*. We use the term global to distinguish such points from *locally optimal points*. Such points solve the problem (2.1) should we only consider points sufficiently close to the locally optimal point in question [47]. The corresponding objective function values of optimality points are referred to as *local* and *global minima* respectively. Optimization problems are by no means guaranteed to have any solution, and if they do, they may either have a finite or infinite number of solutions. As is shown in Figure 2.1, optimization problems need not be contrived to have infinite or no solutions. For some optimization problems it may be trivial to test whether or not a given decision variable solves the problem at hand, whereas for others it may even be theoretically impossible. Although certain algorithms exist that are guaranteed to succeed on specific subclasses of optimization problems, no algorithm

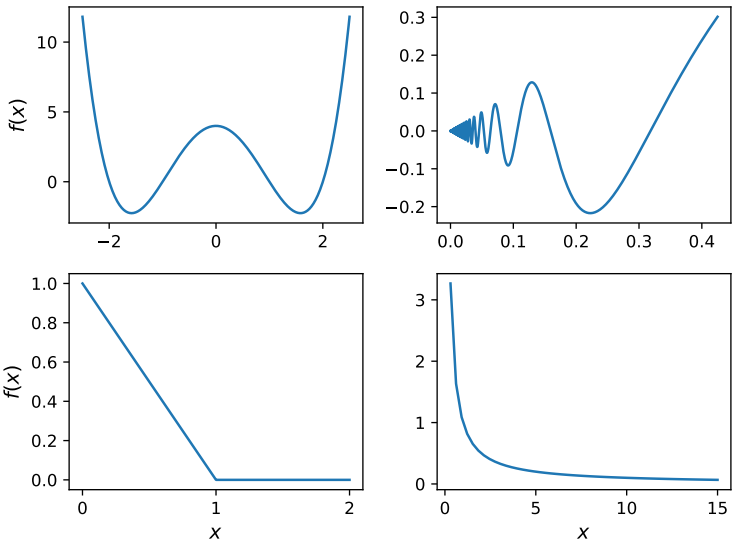


Figure 2.1: Four examples of minimization problems where all except the last one are restrained to the intervals of the x-axes. The graph in each plot corresponds to the objective function. *Top left:* This simple quartic objective function, $f(x) = x^4 - 5x^2 + 4$, has exactly two global minima. *Top right:* This objective function, $f(x) = x \sin(1/x)$, has an infinite number of local minima yet a unique global minimum. *Bottom left:* This simple piece-wise function has an infinite number of global minima in the interval $[1, 2]$. *Bottom right:* The objective function, $f(x) = 1/x$, defined for positive x admits neither local nor global minima. Although the objective function is bounded below from zero, no exact decision variable solves the minimization problem.

is known to reliably handle general optimization problems [46, 47].

Many state-of-the-art optimization algorithms rely on what is called the *dual problem*. The formal dual problem definition may appear contrived and esoteric and is therefore not presented here. It will suffice to describe the dual of an optimization problem as a distinct optimization program derived from the original problem that is—under favorable circumstances—intimately linked to the original optimization program. When speaking of duality, it is common to refer to the original optimization problem as the *primal problem*. If the primal problem is a minimization program, then the dual problem is a maximization program, and vice versa. Of special importance is the concept of the *duality gap*. The duality gap is the difference in value between the primal and dual global solutions. As a matter of fact, the global minimum of a primal minimization program is always larger than the global maximum of the dual, and vice versa for primal maximization problems. This is known as the *weak duality theorem* and holds for any optimization program. This is particularly useful in designing algorithms, as any value of the dual objective function specifies a lower bound on the primal minimization solution.

Different kinds of optimization problems exhibit different properties. Due to the large variety of optimization problems, they are split into different classes. We will only discuss the class of *convex optimization problems* along with two of its subclasses.

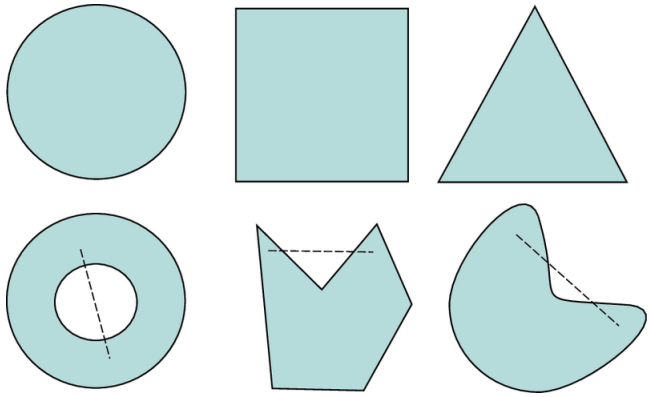


Figure 2.2: Examples of convex and nonconvex sets. *Top:* Three examples of convex sets. Any line connecting two points in the sets is part of the sets themselves. *Bottom:* Three examples of nonconvex sets. Lines violating the set convexity property are illustrated as dashed lines. Figure reprinted from Schweinzer [48].

2.1.1 Convex optimization

Convex optimization problems share desirable mathematical properties and are so well-studied that *convex optimization* has grown to become a research field in its own right. Before defining what is meant by convexity in optimization programs, we need to first define convexity in reference to sets and functions.

The defining feature of a *convex set* is that any line connecting two points in that set is itself in the set. Mathematically we say that set S is convex if, for any pair $x, y \in S$, we have that $\alpha x + (1 - \alpha)y \in S$ for all $\alpha \in [0, 1]$. Examples of convex and nonconvex sets are shown in Figure 2.2

A *convex function* is a function in which any line drawn between two points on its graph lies at or above the graph itself. Mathematically speaking a function, $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, is said to be convex if

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(x)$$

for all $x, y \in \Omega$ and all $\alpha \in [0, 1]$. The two bottom functions in Figure 2.1 are convex, whereas the two upper functions are not. We may now define what it means for an optimization program to be convex.

A convex optimization problem is an optimization problem with a convex function and a convex solution space. There is no known mathematical property that fully describes all convex optimization problems, but we do know some sufficient conditions to ensure convexity in optimization programs. In order to do present these conditions, we must first briefly define what it means for a function to be *affine*. A function, $f : \mathbb{R}^n \to \mathbb{R}^m$, is said to be affine if they are of the form $f(x) = Ax+b$ in which $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^n$. Affine functions happen to be the only kind of function, f, for which both f and $-f$ are convex.

An optimization problem of the form (2.1) with a convex objective, f, convex negative inequality constraints, $-c_i(x)$ for $i \in \mathcal{I}$, and affine equality constraints, $c_i(x)$ for $i \in \mathcal{E}$,

is itself a convex optimization problem. Although this is not a complete description of convex problems, it will suffice for our intentions.

Convex optimization problems share desirable mathematical properties of which we will briefly describe three. First, any locally optimal point in a convex optimization problem is a globally optimal point. This fundamental property is hugely advantageous in designing solvers for convex problems, as it is in general not possible to distinguish local and global minima. Second, the set of optimal points in convex problems is itself a convex set. Third, the duality gap of most convex problems is zero. This extraordinary fact is stated by the *strong duality theorem* and holds whenever *Slater's interiority condition* is satisfied, or in other words, whenever the solution space contains an interior point [46]. Slater's interiority condition may be replaced by other conditions for certain types of convex problems. When the duality gap is zero, solving the dual problem corresponds to solving the primal. The strong duality theorem is at the heart of modern convex optimization solvers [47]. We will now describe two types of convex optimization problems: the **linear program** and the **second-order cone program**.

Linear programs

The LP is one of the most popular optimization models and is especially well-established in management, finance, and engineering. This is mostly due to its mathematical simplicity, wide applicability, and in particular the effectiveness of a solving algorithm called the *simplex method* [46].

The LP is a convex optimization problem (2.1) with a linear objective function as well as linear equality constraint functions and linear inequality constraint functions. An LP will therefore be of the form

$$\begin{aligned} \min_x \ & f^T x \\ \text{s.t. } & a_i^T x = b_i, \ \forall i \in \mathcal{E}, \\ & a_i^T x \leq b_i, \ \forall i \in \mathcal{I}, \end{aligned}$$

where $a_i, x, f \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ with index sets as in problem (2.1). It can easily be verified that LPs are convex. By assuming x, y in the solution space and inserting $\alpha x + (1-\alpha)y$ for x in the constraints, we can easily show that the solution space is convex, which together with the linearity of the objective shows that LPs are convex programs. We could also argue that since all the constraints as well as the objective are affine, we have both a convex objective and solution space, and hence a convex program.

LPs are special, even in the context of convex programming, as strong duality is guaranteed by the structure of the program itself. In other words, LPs are exempt from having to satisfy any type of condition for strong duality to hold. Additionally, the dual programs of LPs are themselves LPs.

The LP solution space is not only convex but also polyhedral. Bounded polyhedrons—extending finitely in all directions—are referred to as *polytopes*. The upper three shapes in Figure 2.2 are all examples of convex polytopes. LPs which possess a nonempty polytope solution space are guaranteed the existence of an optimal solution. Furthermore, the *fundamental theorem of linear programming* states that any LP with a nonempty polytope

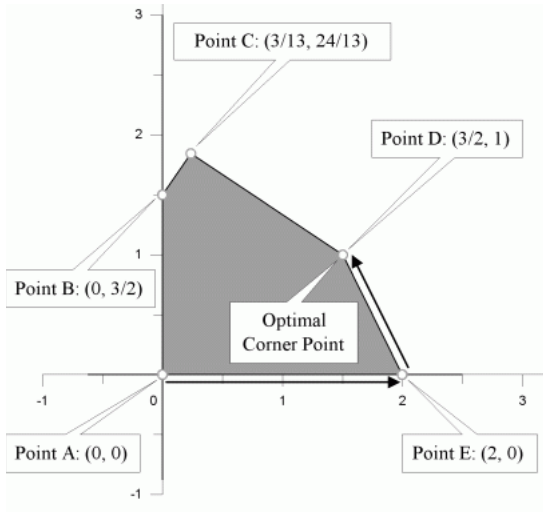


Figure 2.3: The simplex method iterates through the vertices of the LP solution space. LP optimality conditions enable us to confirm or disprove optimality at the vertex of each iterate. Figure reprinted from Niu [50].

solution space has at least one optimal point in a vertex of the polytope. It is this theorem that is the core of the simplex method [49].

As the fundamental theorem of linear programming heavily suggests, the simplex method searches the vertices of the solution space in order to find an optimum [49] as illustrated in Figure 2.3. The simplex method is only guaranteed to find an optimum under very specific circumstances, and the number of iterations needed to find such an optimum will—in the worst case—grow exponentially with the size of the LP [46]. Nonetheless, the number of iterations needed in practice seldom exceeds three times the number of equations in the LP [49], which is excellent when compared to other solving methods. The simplex method continues to be the fastest and most popular modern solving algorithm for LPs [46]. Variants of the simplex methods exist, as for example the dual simplex method which searches the dual solution space for a solution that is feasible in the primal. Should such a solution exist, it will solve the primal problem due to the strong duality theorem.

The theoretical limitations of the simplex method have prompted the use of more sophisticated methods primarily developed for nonlinear optimization problems. These methods typically require the constraints to be satisfied strictly and are hence named *interior-point methods*. One such method was the first to show that LPs are in fact solvable in polynomial time [46]. As the dual simplex method, many commercial interior-point methods rely on the dual in identifying a primal solution.

Many modern algorithms iteratively provide estimates of both the primal and dual solutions. Whenever the difference, or gap, between these iterates' objective value is sufficiently small, the algorithm identifies a global optimum. A wealth of commercial optimization solver suites exists supporting both simplex and interior-point methods for the solving of LPs, some examples being Gurobi [51], FICO Xpress [52], and CPLEX [53].

Second-order cone programs

SOCPs are convex optimization problems often used to recapture parametric uncertainty in LPs [47, 54]. They are solvable by interior-point methods [55, 56] in polynomial time [56], although some solvers require the user to reformulate SOCPs using quadratic constraints.

SOCPs are of the form

$$\min_x \ f^T x \tag{2.4a}$$

$$\text{s.t. } a_i^T x = d_i, \ \forall i \in \mathcal{E}, \tag{2.4b}$$

$$\|A_i x + b_i\| \leq c_i^T x + d_i, \ \forall i \in \mathcal{I}, \tag{2.4c}$$

where $A_i \in \mathbb{R}^{n_i \times n}$, $f, a_i, c_i, x \in \mathbb{R}^n$, $b_i \in \mathbb{R}^{n_i}$, $d_i \in \mathbb{R}$, $\|\cdot\|$ refers to the Euclidean norm, and index sets are defined as for problem (2.1).

The constraints of type (2.4c) are called *second-order cone constraints*. The name is derived from the fact that such a constraint restrains the vector $(A_i x + b_i, c_i^T x + d_i)$ to a second-order cone in $\mathbb{R}^{n_i+1}$ [47]. In other words, the set $\{(A_i x + b_i, c_i^T x + d_i) : \|A_i x + b_i\| \leq c_i^T x + d_i\}$ is a second-order cone. We note that if A_i equals the zero matrix for all $i \in \mathcal{I}$ then the SOCP is equivalent to an LP. Hence, the SOCP generalizes the LP. This will be of later importance to us in the computational implementation of the herein-introduced method (see Section 3.3.1).

We may show that SOCPs are convex by, as before, assuming x, y in the SOCP solution space and then substituting $\alpha x + (1-\alpha)y$ for x in the SOCP constraints. For the second-order cone constraints, we will have to make the observation that $\|A_i(\alpha x + (1-\alpha)y) + b_i\| = \|A_i(\alpha x + (1-\alpha)y) + \alpha b_i + (1-\alpha)b_i\|$, and subsequently apply the triangle inequality. As before, we note that the objective is affine and conclude that SOCPs are indeed convex programs.

SOCPs are convex optimization problems of a form known to be solvable by interior-point methods [55]. Several mathematical optimization suites provide SOCP solvers, such as Gurobi [51], MOSEK [57], and FICO Xpress [52].

Additionally, SOCPs have been shown to behave as robust counterparts to LP [47, 54]. Such an interpretation lends an intuitive understanding of SOCPs worthy of ample discussion.

SOCPs as robust LPs

Consider the following LP-variant, which we will name the *robust* LP,

$$\min_x \ c^T x \tag{2.5a}$$

$$\text{s.t. } a_i^T x \leq b_i, \ \forall a_i^T \in \mathcal{P}_i, \ \forall i \in \{1 \dots m\}, \tag{2.5b}$$

where $\mathcal{P}_i = \{\bar{a}_i^T + u^T P_i : \|u\| \leq 1\}$ with $P_i \in \mathbb{R}^{n \times n}$ and $u \in \mathbb{R}^n$. That is, $\mathcal{P}_i$ describes an ellipsoid centered at $\bar{a}_i$ described by the matrices P_i. For example, with P_i equal to the identity matrix, $\mathcal{P}_i$ is the unit ball centered around the point $\bar{a}_i$.

Intuitively, this means that the ellipsoids describe an uncertainty around $\bar{a}_i$ that our decision variable must account for. For example, an ellipsoid flat in one single dimen-

sion would agree with the interpretation that a single element in $\bar{a}_i$ is free of uncertainty, whereas an ellipsoid extending greatly in one dimension would infer great uncertainty in the element of $\bar{a}_i$ corresponding to that dimension. Sets such as $\mathcal{P}_i$ are therefore often referred to as *uncertainty sets*. Note that these uncertainty sets correspond to sets of constraints that must all be satisfied at once. That is, the robustness of the LP-variant (2.5) stems from the fact that the solutions to the canonical LP constraint, $\bar{a}_i^T x \leq b_i$, must be robust in the uncertain elements of $\bar{a}_i$, as x has to satisfy a whole range of similar LP constraints at once. In other words, small changes to the parameters, $\bar{a}_i$, will not cause dramatic changes in the solution space. This is certainly not true for LPs in general. Just consider an LP with a solution space defined by $x_1 = x_2$ in $\mathbb{R}^2$. Now consider the solution space defined by $(1+\epsilon)x_1 = x_2$ for any $\epsilon \neq 0$. These two solution spaces share only a single point, $x_1 = x_2 = 0$, even though the difference of the constraint functions evaluated at any point can be made arbitrarily small.

Note that a robust LP constraint of the form (2.5b) is equivalent to

$$\sup\{a_i^T x : a_i^T \in \mathcal{P}_i\} \leq b_i$$

and that

$$\begin{aligned}
\sup\{a_i^T x : a_i^T \in \mathcal{P}_i\} &= \sup\{(\bar{a}_i^T + u^T P_i)x : \|u\| \leq 1\} \\
&= \bar{a}_i^T x + \sup\{u^T P_i x : \|u\| \leq 1\} \\
&= \bar{a}_i^T x + \sup\{\|u\|\|P_i x\| \cos\theta : \|u\| \leq 1\} \\
&= \bar{a}_i^T x + \|P_i x\|,
\end{aligned}$$

where θ is the Euclidean angle between u and $P_i x$. We may therefore rewrite problem (2.5) as

$$\min_x \; c^T x \tag{2.6a}$$

$$\text{s.t. } \bar{a}_i^T x + \|P_i x\| \leq b_i, \; i \in \{1 \dots m\}, \tag{2.6b}$$

which we recognize as an SOCP. The added norm term to the constraints are referred to as *regularization terms* and act as barriers restraining x from growing indefinitely. We also see—again—that the P_i transformation determines the allowed deviation for the different elements of x. We conclude that SOCPs of the form (2.6) are aptly interpreted as robust counterparts to LPs.

To ease intuition of the SOCP solution space, we have illustrated an uncertainty set and its resulting constraints in Figure 2.4. Here we see how uncertainty sets describe sets of constraints that together form convex shapes. Robust LPs have smaller solution spaces compared to their respective LPs, as we are essentially adding an infinite number of linear constraints. This fact is also demonstrated in Figure 2.4, where the bisected $\mathbb{R}^2$-plane is transformed into a smaller, distinct convex shape.

Extraordinarily enough, SOCPs have yet another apt interpretation in terms of linear programming. Specifically, we may derive an SOCP by considering an LP with random variable parameters [47]. We will at times refer to such an LP as a *stochastic* LP.

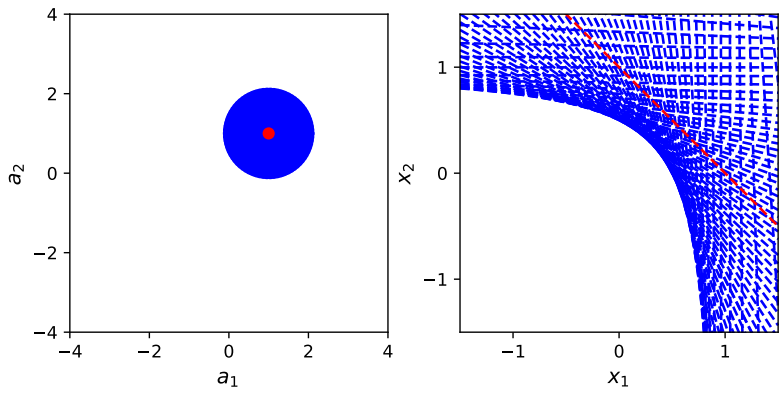


Figure 2.4: The uncertainty set and the resulting constraints of an SOCP of form (2.6b) in which $\bar{a} = [1, 1]^T$, $b = 1$ and P is the 2-by-2 identity matrix. *Left:* The uncertainty set described by $\bar{a}$ and P. The red dot denotes the centre $\bar{a}$. *Right:* Here we have added a representative amount of constraints each derived from points on the boundary of the uncertainty set ellipsoid. The red line corresponds to the constraint derived from the centre $\bar{a}$. The convex shape in bottom-left corner consists of solutions to the corresponding second-order cone constraint.

SOCPs as stochastic LPs

As LPs formulated using stochastic parameters may have varying structures resulting in distinct problem formulations, we will not state the stochastic LP in general form but rather consider two specific types of stochastic LP constraints resulting in SOCP formulations.

Consider the linear constraint, $a^T x \leq b$, where $b \in \mathbb{R}$, $x \in \mathbb{R}^n$, and a is a random n-vector, that is, a vector with at least one entry containing a random variable. Now consider $a^T x$ to be a normally distributed random variable with mean $\bar{a}^T x$ and variance $x^T Q x$ with Q being the covariance matrix of the random elements of a. Now say we enforce the following constraint $P(a^T x > b) \leq \epsilon$, where ϵ is an arbitrary positive scalar. Moving on, we will refer to constraints involving probabilities as *probability constraints* or *stochastic constraints*. With $\delta_{1-\epsilon}$ denoting the $1 - \epsilon$ percentile of the standard normal distribution, rewriting the preceding stochastic constraint in terms of standard normality gives

$$P\left(\frac{a^T x - \bar{a}^T x}{\sqrt{x^T Q x}} \leq \frac{b - \bar{a}^T x}{\sqrt{x^T Q x}}\right) < 1 - \epsilon,$$

which holds if and only if

$$\delta_{1-\epsilon} \leq \frac{b - \bar{a}^T x}{\sqrt{x^T Q x}},$$

and hence

$$\bar{a}^T x + \delta_{1-\epsilon}\sqrt{x^T Q x} \leq b.$$

Now define R—which we will dub the *uncertainty matrix*—so that $R^T R = Q$. We then have $x^T Q x = x^T R^T R x = \|Rx\|^2$. The preceding inequality may thus be expressed by $\bar{a}^T x + \|\delta_{1-\epsilon} R x\| \leq b$ which we recognize as an SOCP constraint.

The reader may call into question the existence of the uncertainty matrix. Are we always guaranteed that the covariance matrix can be written as $R^T R$? Yes, this is guaranteed by the fact that all symmetric positive semidefinite matrices, such as the covariance matrix, has a so-called Cholesky decomposition. More specifically, for any real symmetric positive semidefinite matrix, A, there exists a real lower triangular matrix, L, such that $A = LL^T$. By setting $R = L^T$, we conclude that it is perfectly fine under any circumstances to assume the existence of an uncertainty matrix. The reader should be advised that the uncertainty matrix used is rarely L^T but rather hand-sewn for each application.

Another intriguing question is whether or not the uncertainty matrix is unique. We can show that diagonal covariance matrices—that is, the covariance matrix for uncorrelated random parameters, but for which there may be nonrandom variables as well—have an infinite number of uncertainty matrix representations (see Appendix A.1). This is not worrying though. As long as $R^T R = Q$, the solution space defined by the corresponding second-order cone constraint is the same regardless of R.

Returning to the stochastic LP-variant, consider again the linear inequality $a^T x \leq b$ only this time it is b that is a normal random variable for which we designate mean μ and variance σ^2. We now, as earlier, impose the stochastic constraint $P(a^T x > b) \leq \epsilon$ and arrive at a deterministic inequality. The stochastic constraint in terms of standard normality is

$$P\left(\frac{b-\mu}{\sigma} < \frac{a^T x - \mu}{\sigma}\right) \leq \epsilon,$$

which holds if and only if

$$\frac{a^T x - \mu}{\sigma} < \delta_\epsilon,$$

and hence

$$a_i^T x < \delta_\epsilon \sigma + \mu.$$

As it is intended that ϵ is to be interpreted as a small number, it is reasonable to assume δ_ϵ negative; we see that the inequality resulting from such a probability constraint is simply a stricter linear inequality. We see that we may interpret the parameters of an LP's constraint stochastically and arrive at SOCP constraints.

All in all, we can conclude that robust and stochastic LP constraints correspond to SOCP constraints. It is subsequently reasonable to interpret SOCPs both as stochastic and robust LP counterparts. Before concluding this section, we will present one final SOCP formulation.

SOCPs formulation using quadratic constraints

Some optimization solvers accept only certain formulations of SOCPs. For instance, the Gurobi solver [58] solely accepts SOCPs written as *quadratically constrained programs* (QCPs). As the computational implementations described in the methods chapter rely on the Gurobi solver, we will rewrite the SOCP problem (2.4) as a QCP. The reformulation is given as

$$\min_{x} \; f^T x \tag{2.7a}$$

$$\text{s.t. } a_i^T x = d_i, \; \forall i \in \mathcal{E}, \tag{2.7b}$$

$$A_i^{(j)} x + b_i^{(j)} = y_{i,j}, \; \forall i \in \mathcal{I}, \; \forall j \in \mathcal{J}_i, \tag{2.7c}$$

$$\sum_{j=1}^{n_i} y_{i,j}^2 \leq z_i^2, \; \forall i \in \mathcal{I}, \tag{2.7d}$$

$$z_i \leq c_i^T x + d_i, \; \forall i \in \mathcal{I}, \; , \tag{2.7e}$$

$$z_i \geq 0, \; \forall i \in \mathcal{I}, \tag{2.7f}$$

where the rows of A_i and b_i are denoted by $A_i^{(j)}$ and $b_i^{(j)}$ respectively, and we have introduced two new variables $y_{i,j}, z_i \in \mathbb{R}$ alongside index sets $\mathcal{J}_i$ containing the row indices of A_i and b_i.

By first substituting $A_i^{(j)} x + b_i^{(j)}$ for $y_{i,j}$ into constraint (2.7d), raising it to power $1/2$ using the nonnegativity of z_i from constraint (2.7f), and then finally extending the inequality of constraint (2.7e), we see that the constraints (2.7c) through (2.7f) are equivalent to a second-order cone constraint. The solution space of x in the above problem reformulation is then clearly equivalent to the original SOCP formulation (2.4). This reformulation also shows us that SOCPs belong to the larger class of QCPs. Note that convex QCPs are themselves a subclass of SOCPs whereas this is not true for nonconvex QCPs.

2.2 Genome-scale metabolic models

GEMs are digital, organism-specific representations of metabolic biochemical networks used as the foundation in several computational modeling approaches. Ideally, GEMs are exhaustive reconstructions of the entirety of reactions, genes, and metabolites participating in a given organism's cellular metabolism. For this reason, we often refer to GEMs as metabolic network reconstructions, or simply metabolic reconstructions. GEMs typically represent unicellular species, such as *E. coli* and *Saccharomyces cerevisiae*, but have in recent years expanded in scope and increased in granularity with the introduction of strain-specific prokaryotic GEMs [59, 60] and cell-line specific human GEMs [61].

The reconstruction process is dependent on genome annotation and experimentally obtained metabolic parameters derived from databases and scientific literature. To date there exists GEMs—either manually or automatically curated—for over 6239 organisms [26]. GEMs are used extensively, with a repertoire of different approaches, to computationally model cellular metabolism and are frequently employed in computational biology research. As will be evident in Section 2.3.1, mathematical optimization is the crux of

most computational efforts using metabolic reconstructions. A coarse outline of the reconstruction and applications of GEMs is shown in Figure 2.5. In this section on GEMs, we will first briefly describe some of its applications before outlining the GEM reconstruction procedure as described by Thiele & Palsson [27].

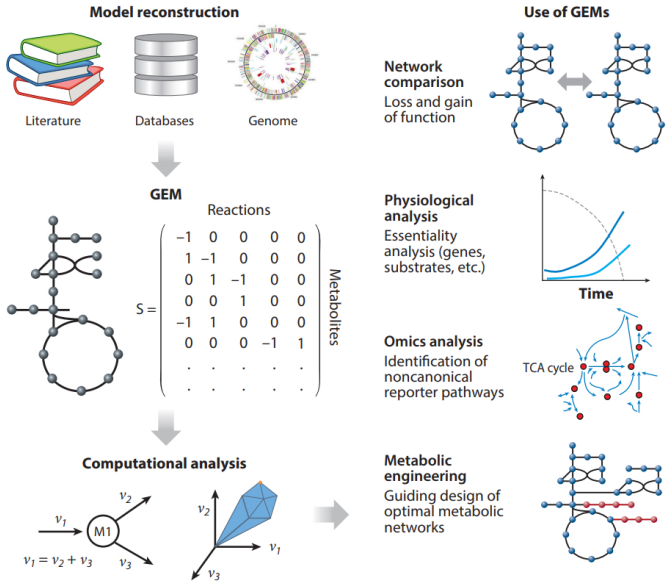


Figure 2.5: An overview of the process of reconstructing and analyzing GEMs. Metabolic reconstructions are created using knowledge drawn from online databases, literature, and annotated genomes. As can be seen here and will be discussed further in Section 2.2.2, GEMs are often represented mathematically by a matrix. Computational analyses include, but are not limited to, the comparison of metabolic networks across taxa, identification of essential genes, discovery of novel reporter pathways, and metabolic engineering. Figure reprinted from Nielsen [62].

2.2.1 Applications of genome-scale metabolic models

Several methods have been used for analyzing genome-scale metabolic reconstructions, for example, elementary-mode analysis [63], identification of extreme pathways [64], Monte Carlo sampling of metabolites states [65], topological analysis [66], FBA [30] and numerous FBA-derived methods [35, 38, 39, 41, 67, 68].

These approaches commonly impose restrictions on permissible metabolic states to provide further insights into cellular metabolism. These restrictions, or constraints, are often based on combinations of genetic, transcriptomic, proteomic, and thermodynamic data. Such constraints may also impose certain environmental conditions as to simulate growth under specific culture media. For these reasons, the vast majority of methodologies used to analyze GEMs fall under the umbrella term of *constraint-based analysis*. In fact, the reconstruction and *in silico* analysis of GEMs are often collectively referred to

as *constraint-based metabolic modeling*. An overview of constraint-based modeling approaches is shown in Figure 2.6. In applying these methods, researchers have drawn new insight into cellular metabolism. We outline a few examples of constraint-based metabolic modeling applications before continuing on to describe the GEM reconstruction process.

Combining topological analysis and FBA, Almaas *et al.* [69] found that a small subset of key biochemical reactions dictates the metabolic changes undergone by *E. coli* under different media conditions. Also working on *E. coli*, Yim *et al.* [70] engineered a mutant strain capable of producing the commodity polymer 1,4-butanediol using a genome-scale metabolic reconstruction. By reconstructing human hepatocyte metabolism and overlaying the subsequent GEM with transcriptomic data, Mardinoglu *et al.* [71] proposed new diagnosis techniques for non-alcoholic fatty liver diseases and predicted patients of said diseases to suffer from serine deficiency. Zelezniak *et al.* [72] performed a large-scale survey of bacterial metabolic reconstructions whose results suggested that interacting groups of bacterial cooperators persist in different conditions due to metabolic interdependencies. As is evident from the preceding examples, GEMs have proven viable computational tools for biological research.

2.2.2 Reconstruction process of genome-scale metabolic models

The reconstruction described here compromises four stages: drafting, refinement, conversion to computable format, and validation. Figure 2.7 presents an overview of the process and illustrates the iterative nature of the reconstruction procedure. GEMs may be refined several times before they are deemed adequate in quality. First, metabolic reactions are inferred from an organism-specific genome annotation. Second, the reactions are manually curated, and aspects of the model especially important to its predictive power are heavily scrutinized. Of paramount importance to the model's predictive capabilities is the *biomass reaction*: an artificial reaction consuming the nutrients required for cell growth. Its importance stems from its intimate link to the widely used FBA method, which we describe later in 2.3.1. Third, the model is converted into a standardized computational format. Fourth, the model is evaluated and verified. Upon inconsistencies and shortcomings, the process loops back to the refinement stage. As GEMs are formed in a bottom-up fashion and rely extensively on large-scale databases, they are themselves repositories of biochemical, genetic, and metabolic knowledge.

Draft reconstruction

Initially, a genome annotation of the target organism is retrieved. Metabolic genes are identified in the annotation along with their gene products—commonly specified by enzyme commission (EC) numbers—and associated biochemical reactions. It is not expected that the initial draft reconstruction will be exhaustive nor free of errors; manual curation of the draft is necessary to ensure a high-quality GEM. The draft reconstruction is a collection of candidate metabolic genes and their associated biochemical reactions, or in other words, a collection of metabolic GPR associations. The collection is corrected, pruned, and expanded upon during the reconstruction refinement stage.

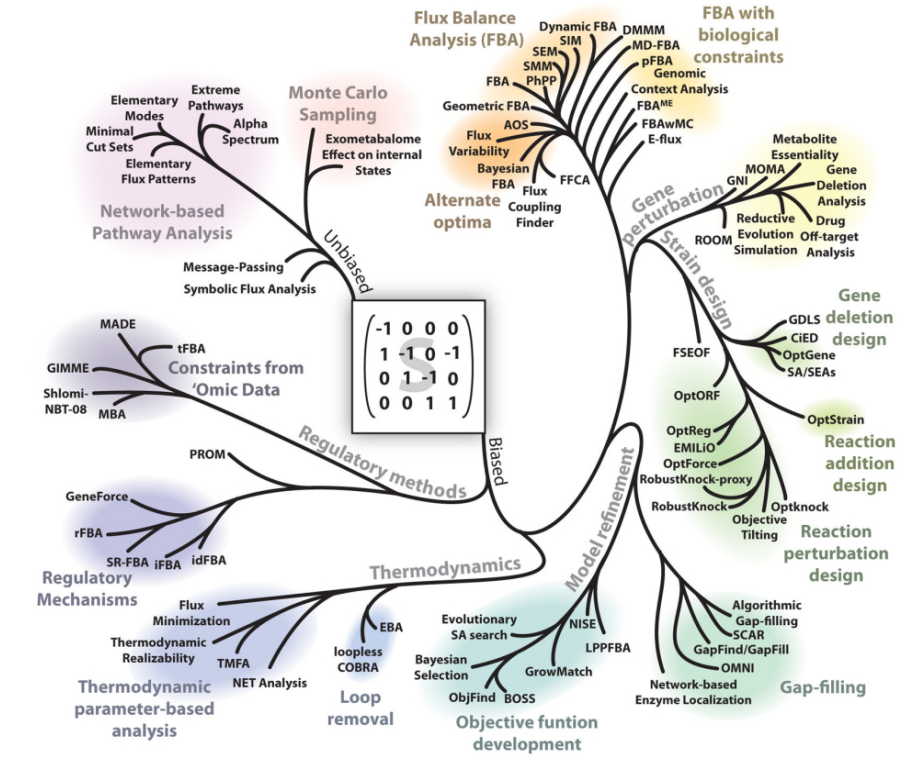


Figure 2.6: An overview of constraint-based analysis methods of GEMs. Again, we see the matrix representation of a metabolic network reconstruction in the center of the figure. Be aware that these methods are not necessarily mutually exclusive and are more interrelated than what this figure might suggest. As is shown, constraint-based analysis methods exist that are developed for the sole purpose of refining the GEMs themselves. Figure reprinted from Lewis *et al.* [73].

Reconstruction refinement

Refinement is by far the most comprehensive stage of the reconstruction process. The metabolic functions of annotated genes are verified—using either online databases or literature—and assigned confidence scores. Reactions related to metabolic functions that contain generic terms are removed. Organism-specific data are verified whenever possible, which is especially important for enzymes that catalyze multiple substrates, bind different cofactors, or catalyze disparate biochemical reactions.

The reactions of the reconstruction are probed for mass and charge balance. As metabolite charges are often pH-dependent—a fact at times ignored in biochemical databases—they may differ from the biochemical environment assumed by the modeler. This often necessitates the use of software to predict pK_a values via the chemical groups of metabolite species. Unbalanced reactions may lead to energetically costless formation of protons

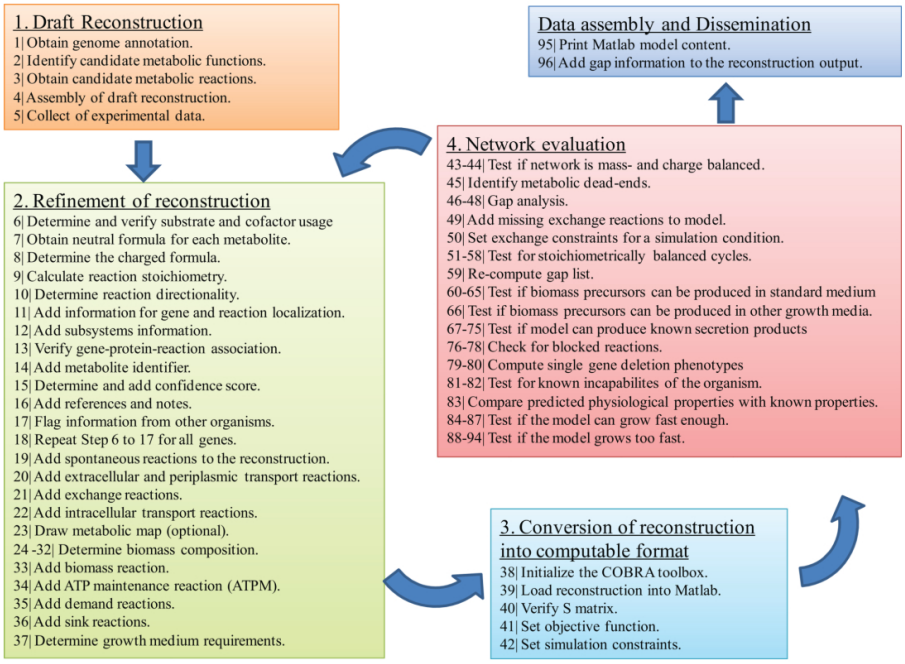


Figure 2.7: An overview of the reconstruction process of GEMs. The steps, corresponding to the numbered blocks in the figure, are described further in Section 2.2.2. Figure reprinted from Thiele & Palsson [27].

and ATP which may significantly affect model predictions.

The reaction directionality is also determined using biochemical data, which is unfortunately often lacking. Determining directionality is often done using estimations based on standard Gibbs free energy, heuristic rules based on network topology, and biochemical literature. It is often so that reactions to which no directionality can be determined are assumed reversible. Unfortunately, models with too many reversible reactions often include so-called futile cycles that may underestimate the cost of maintaining proton gradients across compartments.

Other genetic and biochemical data are included in the model, such as metabolite localization, metabolite/reaction identifiers, and GPR associations. All of these inclusions can be automated but are usually manually curated as well in order to ensure high-quality models. The GPR associations are particularly important to the model's predictive power, as they are the pith of computational gene-knockout simulations.

Missing metabolic functions are also filled in the refinement stage. This is referred to as *gap-filling* and entails identifying reactions that are missing from knowledge bases but which must be included to ensure model quality. These gap-fillings must be accompanied by biological reasoning, as some missing metabolic functions are indeed missing in the

real-world biochemical network. Done correctly, gap-filling may provide hypotheses for the existence of metabolic pathways in the target organism.

It is also in the refinement process that artificial reactions, also dubbed *pseudoreactions*, are created. The biomass reaction is an important pseudoreaction that consumes known biomass constituents and produces an artificial metabolite, or *pseudometabolite*, symbolizing the log phase growth rate. The stoichiometric coefficients of the biomass pseudoreaction are experimentally determined fractions of the overall biomass and must be drawn from experiments investigating biomass under log phase. These fractions are collectively referred to as the biomass composition and compromise the major biomolecular families—protein, carbohydrates, lipids, DNA, and RNA—as well as other soluble species such as polyamines, vitamins, cofactors, and ions. For biomass composition coefficients that are difficult to determine, there exist estimation methods using genomic data. For many reconstructions, the unavailability of experimentally determined biomass compositions necessitates the use of compositions from phylogenetically similar organisms. Lipids, and in particular phospholipids, exist in many different chemically similar forms and experimental methods may not be able to fully map the contribution of each different chemical species to the average composition of fatty acid chains and head groups. Hence, GEMs typically only use averages over lipid chains and head groups. The simplified lipid composition leads to faulty predictions but has been ameliorated by the introduction of pseudoreactions belonging to a formalism known as SLIMEr [74]. It is has been acknowledged that the accuracy of the biomass composition is a major limiting factor in building successful GEMs, as reaction flux predictions and growth-limiting nutrient identification are highly sensitive to changes in the biomass pseudoreaction [75].

Two important additional pseudoreactions are the *growth/nongrowth associated ATP maintenance* (GAM/NGAM) reactions. As the names suggest, these reactions simulate the ATP cost of growth—such as the synthesis of proteins, DNA, and RNA—and nongrowth processes—such as the maintenance of turgor pressure. These pseudoreactions are usually inferred from chemostat experiments under oxygen- and glucose-rich conditions or are in other cases inferred by parameter-fitting procedures. As GAM/NGAM stoichiometric coefficients are inferred experimentally, they are, strictly speaking, only viable estimations for the given media condition.

Extensions to the canonical GEM format has introduced many new flavors of pseudoreactions and pseudometabolites. One example is the GECKO formalism [38], which imposes mass constraints on total enzyme mass using enzyme kinetic data. Extensions such as GECKO and others like it will be described later in Section 2.4.2. Although our discussion will then focus on the mathematical representation of metabolic models, it is worthwhile to contemplate that we will in a sense return to the process of reconstruction refinement.

Conversion to a mathematical model via the stoichiometric matrix

This step is highly automated and centers around instantiating the *stoichiometric matrix*. In this matrix, the rows and columns coincide with the model's metabolites and reactions respectively. Each element in the matrix specifies the stoichiometric coefficient of a metabolite in a given biochemical reaction. Negative elements correspond to reactant substrates whereas positive elements correspond to reactant products. We will discuss the

stoichiometric matrix in greater detail in Sections 2.3.1 and 2.4.2.

We will speak of the computational and mathematical GEM formats fleetingly but will refer to the computational format when focusing on GEMs in their entirety and not only their corresponding stoichiometric matrices. The computational format of the GEM typically includes reaction directionality, GPR relations, reaction identifiers, metabolite identifiers, gene identifiers, relations between reactions and cell compartments, and other data in addition to the stoichiometric matrix. All this information is conveniently stored in the systems biology markup language (SBML), which is an open XML-based format designed to represent biochemical networks [76]. The reconstruction may either be stored directly in SBML format or may be temporarily stored in other formats, such as Excel, before being converted into SBML. Designed to be a computational format standard, SBML is aimed at easing cooperative sharing, evaluation, and development of GEMs. Several software suites exist for common programming languages that facilitate the conversion from SBML format to language-specific data objects. Examples of such software suites are the COBRA [77] (available for MATLAB and Python) and RAVEN [78] (available for MATLAB) toolboxes. These software suites aim to ease further reconstruction refinement and offer multi-purpose platforms for *in silico* analyses.

Model evaluation and validation

During this step, various methods are used to probe the comprehensiveness and predictive power of the GEM. The types of assessments done will depend on the intended use of the model, though some are more common than others.

To probe the comprehensiveness of the model, so-called *dead-end metabolites* are identified. These metabolites have either no influx or efflux and are thus problematic when simulating nonchanging conditions. Related to dead-end metabolites are *blocked reactions*. These are reactions that cannot support flux given nonchanging metabolite concentrations. Both blocked reactions and dead-end metabolites are indicators that gap-filling is necessary to improve model validity.

The quality of the biomass reaction is of utmost importance in model validation. In order to affirm the quality of the biomass reaction, the GEM must be capable of producing each single biomass precursor for conditions in which the species is known to grow. The GEM should also be tested for total biomass production in conditions for which it is known to not grow. For some organisms it may even be known what nutrient is limiting under different media conditions. For such organisms, we may test if the model itself is limiting on that particular nutrient. During this step, growth-related parameters may be fitted according to experimental data.

Simulatory approaches to GEMs may provide opportunities to test if the model recaptures phenotypes properties known in the organism. Inclusion of novel biochemical data may be necessary to do so successfully as will be exemplified later in Section 2.4.2.

Other tests and quality-reassurance methods exist that may together provide the incentive to reiterate back to the refinement stage of the reconstruction process. If on the other hand, the assessments of the GEM are congruent to experimental data and are deemed sufficiently comprehensive, the reconstruction is considered finished. Many metabolic reconstructions are updated regularly and made available in online depositories such as BiGG [79] and GitHub repositories.

2.3 Constraint-based analysis of metabolic models

In this section, we will describe the most common method for constraint-based analysis of metabolic models: FBA [30]. We will also describe a generalization of FBA, called RAMP [45], that serves to handle the uncertainty inherent to modeling parameters found in metabolic reconstructions. As we will see, the deceptively simple FBA formulation gives rise to a variety of applications in both bioengineering and medicine but is built on dubious assumptions that are resolved by the RAMP formalism.

2.3.1 Flux balance analysis

In FBA, a biochemical network reconstruction is used in the formulation of an optimization problem aimed at modeling cellular metabolism. FBA is a versatile and powerful framework for modeling metabolism having proved viable in a diverse range of research fields, including cell biology [42, 66, 69, 72], bioengineering [31, 32, 70], and medicine [33, 34, 71]. We will now motivate the FBA formulation and then state its complete formulation as an LP.

Consider an organism whose metabolism is described by m metabolites and r biochemical reactions. Any biochemical reaction, with index j, of the r reactions may be described by the chemical equation

$$\mathrm{a}_{1,j}\,[\mathrm{x}_1] + \mathrm{a}_{2,j}\,[\mathrm{x}_2] + \ldots \mathrm{a}_{m,j}\,[\mathrm{x}_m] \underset{\mathrm{k}^j_-}{\overset{\mathrm{k}^j_+}{\rightleftharpoons}} \mathrm{b}_{1,j}\,[\mathrm{x}_1] + \mathrm{b}_{2,j}\,[\mathrm{x}_2] + \ldots \mathrm{b}_{m,j}\,[\mathrm{x}_m],$$

where $[x_i]$ represents the concentration of metabolite i, with k^j_+ and k^j_- denoting the reaction rate constants in forward and reverse direction respectively. If a reaction is irreversible, we simply set either rate to zero. Accordingly, if a metabolite is absent in the reaction we set both both coefficients $a_{i,j}$ and $b_{i,j}$ to zero. Let us denote the reaction rate from left to right as v^j_+ and the reaction rate from right to left as v^j_-. The sum of these reaction rates, v, is referred to as the *flux* through reaction j. It is thus clear that the rate of change in concentration of metabolite i is

$$\begin{aligned}\frac{d[x_i]}{dt} &= \sum_{j=1}^{r}(b_{i,j} - a_{i,j})(v^j_+ - v^j_-) \\ &= \sum_{j=1}^{r}(b_{i,j} - a_{i,j})v^j \\ &= S_i^T v,\end{aligned}$$

where $S_i, v \in \mathbb{R}^r$ with S_i being the vector of the stoichiometric coefficients of metabolite i. We may use the stoichiometric coefficient vectors, S_i, as rows in the matrix S. This matrix is called the stoichiometric matrix and is precisely the stoichiometric matrix constructed from genome-scale metabolic models (see section 2.2.2).

The basic principles of FBA are the following. First, assume that the rate of change in concentration of all metabolites is zero. This is called the *steady state* (SS) *assumption*

and is described by the matrix-vector product $Sv = 0$, where we note that the right-hand side denotes the zero vector and not the zero scalar. Second, fluxes are constrained as to confer to reaction directionality and environmental conditions. Reaction directionalities are extracted from the GEM, and fluxes transporting extracellular metabolites into the cell should be constrained as to simulate a given environment. For example, if the modeler wishes to simulate an oxygen-rich culture growing on glucose-rich media, all transport fluxes except for the oxygen and glucose transport should be constrained to zero. Additionally, the limiting nutrient factor transport influx, in this example the glucose influx, should be constrained by an experimentally determined upper value. If not constrained, the cell would simply consume the limiting nutrient at a rate higher than what is biologically possible and hence confer to faulty modeling predictions. Third, we use the preceding constraints—flux bounds and SS assumption—to formulate an optimization problem in which the objective is set for maximized biomass production (see section 2.2.2).

The resulting optimization problem is in LP form and is readily solved by commercial solvers such as the very time-effective simplex method (see section 2.1.1). The rudimentary concept of FBA is thus to extrapolate the stoichiometric matrix from a metabolic reconstruction, set transport flux bounds, assume SS, and optimize biomass production. The idea being that the organism in question has itself, through evolution, employed a set of fluxes that optimize growth and should thus coincide with the solution(s) to the FBA optimization problem. Hence, the biomass objective function is only tenable to simulations of organisms in stable environments to which they are fully optimized for growth. Such situations are typically restricted to experimentally controlled monocultures during exponential growth phase.

The FBA paradigm employing biomass production as optimization objective has been experimentally proven to be conceptually justifiable [80, 81] and has served as the backbone in an abundant number of research endeavors in computational biology [31–34, 66, 69–72] including a multitude of elaborations on the canonical FBA framework [35, 38, 39, 41, 45, 67, 68, 74]. FBA has also been included as a submodule in larger models aiming to concurrently describe multiple aspects of cellular biology [42, 43]. For an overview of the procedure of an FBA simulation see Figure 2.8. We will now proceed to formulate the FBA problem as an optimization program.

The linear program constituting the FBA framework is formulated as

$$\max_{v} \; c^T v \tag{2.8a}$$

$$\text{s.t. } Sv = 0, \tag{2.8b}$$

$$L \leq v \leq U. \tag{2.8c}$$

where r and m denote number of reactions and metabolites respectively, $c \in \mathbb{R}^r$, $L, U, v \in \mathbb{R}^r$, are in units of mmol gDWh^{-1}—that is, millimol per gram dry weight per hour—and $S \in \mathbb{R}^{m \times r}$ is the stoichiometric matrix. The objective vector c specifies the flux(es) being optimized. Typically, the objective vector has only zero-entries except at the entry corresponding to the biomass reaction, which contains the value one, resulting in an optimization program maximizing the rate of biomass production. The first constraint (2.8b) disallows changes in concentration of all metabolites and imposes SS. The second constraint (2.8c)—for which the inequality sign is meant to be interpreted entry-wise—simply

asserts upper and lower bounds on the reaction fluxes so that reaction must adhere to their directionality. In certain cases the upper and lower bounds represent experimentally determined bounds on nutrient uptake rates and cost-associated pseudoreactions such as the GAM/NGAM reactions.

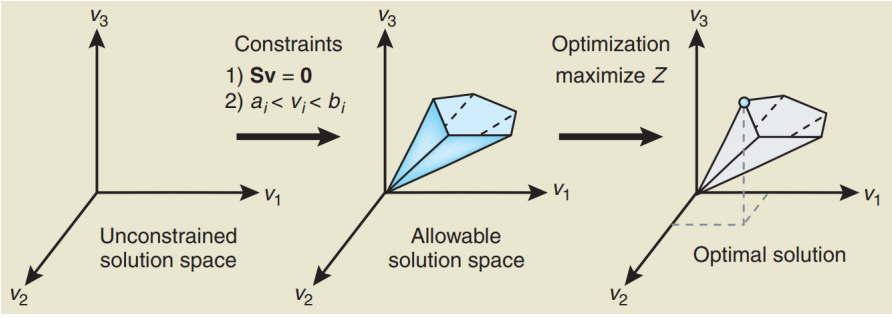


Figure 2.8: An overview of an FBA simulation. By assuming SS and imposing upper and lower bounds on fluxes, we define the solution space. The upper and lower bounds may be used to define the directionality of reactions as well as defining the nutrient environment. Finally, we identify the optimal reaction flux vector, which is more often than not optimized for biomass production. Figure reprinted from Orth *et al.* [30].

2.3.2 Robust analysis of metabolic pathways

In RAMP, we abandon the LP and arrive at a model of cellular metabolism formulated in SOCP form. Although RAMP is computationally unstable, it provides a method that acknowledges the inherent uncertainty of modeling parameters found in GEMs. We will describe the RAMP formalism per MacGillivray *et al.* [45]. In light of our discussion on SOCPs in Section 2.1.1, it is worth mentioning that the following derivation of RAMP follows the stochastic SOCP interpretation. And although the stochastic interpretation provides a more viable lens from a biological perspective, it is also meaningful to interpret RAMP nonstochastically; these different interpretations will be discussed briefly after introducing the RAMP method formulation.

Consider a cell culture for which we define $\mathcal{C}$ to be the collection of cells inhabiting that culture. We assume the cells of the culture to be genetically identical and optimized to their current environment. Let $m \in \mathbb{R}$ be the number of metabolites in the cell and $[x_i^c]$ be the concentration of metabolite i in cell c. Hence, the collection of $d[x_i^c]/dt$ is a random sample of the rate of change of concentration of metabolite i in cell c. Defining the average rate of change of metabolite i of the cell population as $d[x_i^{\mathcal{C}}]/dt$, we have

$$\frac{d[x_i^{\mathcal{C}}]}{dt} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{d[x_i^c]}{dt}.$$

As bacterial cultures typically contain hundreds of millions of cells, we may consider

$d[x_i^{\mathcal{C}}]/dt$ to be approximately normal by the central limit theorem. Now, let us assume the following probabilistic constraints

$$P(d[x_i^{\mathcal{C}}]/dt > M_i) \leq \epsilon \quad \text{and} \quad P(d[x_i^{\mathcal{C}}]/dt < -M_i) \leq \epsilon,$$

where M_i, $\epsilon \in \mathbb{R}$. These constraints combined assert that $P(-M_i \leq d[x_i^{\mathcal{C}}]/dt \leq M_i) \geq 1 - 2\epsilon$, or in other words, that the average rate of concentration of metabolite i is—to an arbitrary probability determined by ϵ—set within the bounds of the absolute value of parameter M_i. Let μ_i and σ_i be the mean and standard deviation of $d[x_i^{\mathcal{C}}]/dt$. Since $d[x_i^{\mathcal{C}}]/dt$ is normal we have that $(d[x_i^{\mathcal{C}}]/dt - \mu_i)/\sigma_i$ is a standard normal variable. We may then rewrite the above stochastic constraints as

$$\frac{M_i - \mu_i}{\sigma_i} \geq \delta_{1-\epsilon} \quad \text{and} \quad \frac{-M_i - \mu_i}{\sigma_i} \leq \delta_\epsilon, \tag{2.9}$$

in which δ_ϵ and $\delta_{1-\epsilon}$ are the ϵ and $1 - \epsilon$ standard percentiles respectively. Using the fact that $\delta_\epsilon = -\delta_{1-\epsilon}$, we may rearrange the preceding inequalities as

$$\sigma_i \leq \min \left\{ \frac{M_i - \mu_i}{\delta_{1-\epsilon}}, \frac{M_i + \mu_i}{\delta_{1-\epsilon}} \right\}. \tag{2.10}$$

We would like to exploit this inequality in constructing the second-order cone constraints, but to do this we will have to find an expression for μ_i. Consider the stoichiometric matrix of cell $c \in \mathcal{C}$ to consist of rows $S_i^c \in \mathbb{R}^r$—where r is the number of reactions—in which the coefficients are defined, in general, to be random variables. Coefficients to which we attribute no uncertainty would be modeled as random variables with a single outcome. We consider the flux vector $v \in \mathbb{R}^r$ to be common among the inhabitants of the culture $\mathcal{C}$. From this, we have the following expression

$$\begin{aligned}
\mu_i &= E\left(\frac{d[x_i^{\mathcal{C}}]}{dt}\right) \\
&= E\left(\frac{1}{|\mathcal{C}|}\frac{d[x_i^c]}{dt}\right) \\
&= E\left(\frac{\sum_{c\in\mathcal{C}} S_i^c}{|\mathcal{C}|} v\right) \\
&= E\left(S_i v\right) \\
&= E\left(S_i\right) v
\end{aligned}$$

where S_i is the (random) matrix row consisting of the average stoichiometric coefficients in a sample of the cells in culture $\mathcal{C}$. We think it is worth reminding the reader that there has been no made no assumptions to the precise value of μ_i besides the inequalities (2.9). To continue our development of RAMP, we will have to discuss the probability distribution of $d[x_i^{\mathcal{C}}]/dt$. As MacGillivray *et al.* [45] assume a general discrete distribution for $d[x_i^{\mathcal{C}}]/dt$, we will follow in this vein for now and will later introduce a simple uniform continuous distribution in Chapter 4. In the development described above, the stochasticity pertains to the stoichiometric coefficients and not the flux vector. Hence, a discrete distribution

of $d[x_i^{\mathcal{C}}]/dt$ implies there to be a finite number of outcomes q—or so-called *scenarios*—for S_i. Each of these scenarios corresponds to a nonrandom row which we name S_{ik} for $k \in \{1 \dots q\}$. Each scenario has a probability of occurrence p_{ik} for $k \in \{1 \dots q\}$ which we collect in the vector $p_i \in \mathbb{R}^q$. Finally, we let $P_i \in \mathbb{R}^{q \times q}$ be the diagonal matrix composed from p_i, and let $\hat{S}_i \in \mathbb{R}^{k \times r}$ be the matrix in which row k corresponds to S_{ik}. Then

$$\mu_i = E\left(S_i\right) v = p_i^T \hat{S}_i v.$$

and

$$\begin{aligned}
\sigma_i^2 &= \text{Var}(S_i v) \\
&= \sum_{k=1}^{q} p_{ik}(S_{ik}v - E(S_{ik}v))^2 \\
&= (\hat{S}_i v)^T (I - e p_i^T)^T P_i (I - e p_i^T) \hat{S}_i v,
\end{aligned}$$

where e is a vector of ones. By defining $R_i = \delta_{1-\epsilon}\sqrt{P_i}(I - e p_i^T)\hat{S}_i$ we simplify the preceding expression to

$$\sigma_i = \frac{||R_i v||}{\delta_{1-\epsilon}},$$

that combined with (2.10) gives us

$$||R_i v|| - M_i \leq p_i^T \hat{S}_i v \leq M_i - ||R_i v||,$$

which is itself a combination of two second-order cone constraints.

We are now ready to formulate the RAMP optimization program. Defining c, L, U as in section 2.3.1, the RAMP method is described by

$$\max c^T v \tag{2.11a}$$

$$\text{s.t. } ||R_i v|| \leq p_i^T \hat{S}_i v + M_i, \quad i \in \{1, \dots, m\} \tag{2.11b}$$

$$||R_i v|| \leq -p_i^T \hat{S}_i v + M_i, \quad i \in \{1, \dots, m\} \tag{2.11c}$$

$$L \leq v \leq U. \tag{2.11d}$$

It is worth mentioning that uncertainty in upper and lower bounds on fluxes could be modeled by treating the stoichiometric coefficients relating to that flux as random.

Rewriting the second-order cone constraints of RAMP (2.11) might present the solution space in more intuitive light. Using the fact that $||R_i v|| = \max\{u^T R_i v : ||u|| \leq 1\}$ we may rewrite constraint (2.11b) as

$$\begin{aligned}
-p_i^T \hat{S}_i + ||R_i v|| \leq M_i &\Leftrightarrow \max\{-p_i^T \hat{S}_i + u^T R_i v : ||u|| \leq 1\} \leq M_i \\
&\Leftrightarrow S_i v \leq M_i, \, \forall S_i \in \{-p_i^T \hat{S}_i + u^T R_i : ||u|| \leq 1\},
\end{aligned}$$

and similarly for constraint (2.11c), we have

$$p_i^T \hat{S}_i + ||R_i v|| \leq M_i \Leftrightarrow S_i v \leq M_i, \, \forall S_i \in \{p_i^T \hat{S}_i + u^T R_i : ||u|| \leq 1\}.$$

We see that the solution of the RAMP problem consists of fluxes that satisfy a relaxed SS assumption for a set of coefficients which are defined by the discrete distribution of the rows of the stoichiometric matrix. Given our development of the model based off stochastic considerations, it is reasonable to interpret the solution space as the space of fluxes almost satisfying the SS assumption given uncertainty in the stoichiometric coefficients. But this is not the only appropriate interpretation, as strictly speaking, the solution space requires the fluxes to satisfy a set of constraints found in the (higher-dimensional) proximity of the average constraint $p_i^T \hat{S}_i$. This could just as easily be interpreted as an infinite set of qualitatively similar rows being added to the stoichiometric matrix. An illustrative example of the difference between an FBA SS constraint and the corresponding relaxed SS constraint is shown in Figure 2.9.

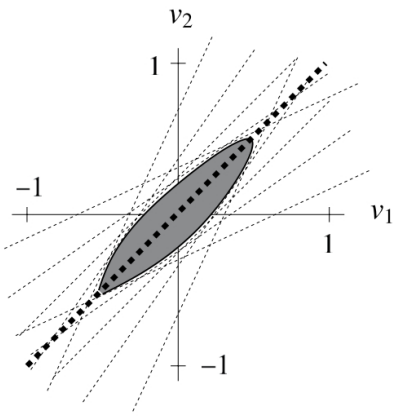


Figure 2.9: Comparison of an FBA SS constraint (bold, dashed line) and its corresponding RAMP constraints (light, dashed lines). The gray area enclosed by the light, dashed lines corresponds to the subspace satisfying the RAMP constraint, whereas the bold, dashed line corresponds to the subspace solving the FBA constraint. Figure reprinted from MacGillivray *et al.* [45].

When comparing the predictive power of RAMP and FBA with an *E. coli* reconstruction, MacGillivray *et al.* [45] found that RAMP's solution space inhabits fluxes closer to experimental flux data than is possible with FBA. The result holds for different conditions as is shown in Figure 2.10. In addition, RAMP was shown to have similar predictive power to that of FBA in the identification of essential genes [45]. Moreover, RAMP possesses key mathematical properties not found in FBA, which we will now go on to discuss.

We briefly state three theorems on RAMP, without proof, as presented by MacGillivray *et al.* [45]. The first theorem—stated below—affirms that the RAMP problem is continuous in its probabilistic parameters. More precisely, it shows that for any feasible flux there will, under sufficiently small changes in its parameters, exist an arbitrarily close flux that is feasible in a slightly perturbed RAMP program. This is in stark contrast to the FBA method, as small changes in the stoichiometric coefficients may lead to abruptly dissimilar solution spaces. An example of such instability in FBA constraints is given by MacGillivray *et al.* [45].

Theorem 1. *For a collection of probability vectors p_i and scenario matrices $\hat{S}_i$ and for the lower and upper bounds U and L, let $\mathcal{F}(\{(p_i, \hat{S}_i) : i = 1, 2, \ldots, m\}.L, U)$ be the nonempty set of feasible fluxes satisfying the constraints of the RAMP model defined by equation set* (2.11)*. Assuming that each $p_i + \triangle p_i$ is a probability vector, we then have for each*

$$v \in \mathcal{F}(\{(p_i, \hat{S}_i) : i = 1, 2, \ldots, m\}.L, U)$$

that there is a $\lambda \geq 0$ such that

$$\min_{v'} ||v - v'|| \leq \lambda \Gamma$$

where

$$\Gamma = \max_i \{||\triangle p_i + ||\triangle \hat{S}_i|| + ||\triangle p_i||||\triangle \hat{S}_i||\},$$
$$v' \in \mathcal{F}(\{(p_i + \triangle p_i, \hat{S}_i + \triangle \hat{S}_i) : i = 1, 2, \ldots, m\}.L - \lambda \Gamma e, U + \lambda \Gamma e),$$
and λ is independent of all $||\triangle \hat{S}_i||$.

The second theorem describes an intimate connection between the RAMP and FBA methods, stating that RAMP problems converge to FBA problems as stochasticity in modeling parameters diminishes. It effectively shows that RAMP encapsulates FBA, and that we may rightfully interpret RAMP as a generalization of FBA.

Theorem 2. *Let p_i^t, $\hat{S}_i{}^t$, M_i^t, be sequences such that for each t the corresponding RAMP model satisfies Slater's interiority condition. Let v^t be an optimal solution of the RAMP model corresponding to p_i^t, $\hat{S}_i{}^t$, and M_i^t, and assume $v^t \to v$ as $t \to \infty$. Assume likewise that a corresponding dual sequence of optimal solutions converges. Further assume that as $t \to \infty$ we have $(p_i^t)^T \hat{S}_i{}^t \to \bar{S}_i$, $R_i^t \to 0$ and $M_i^t \to 0$. Then v is a solution to the FBA model*

$$\begin{aligned} &\max_v \ c^T v \\ \textit{s.t.} \ &\bar{S} v = 0, \\ &L \leq v \leq M, \end{aligned}$$

where $\bar{S}$ is the matrix whose i-th row is $\bar{S}_i$.

The proof of Theorem 2 (not shown) also implies that not all stochastic parameters need to vanish in order to arrive at an FBA problem. This implication is explored further in the third theorem. But first, we define a scenario, $\hat{S}_i$, as *biologically possible* for a solution, $\hat{v}$, of an FBA problem described by stoichiometric matrix of rows $p_i^T \hat{S}_i$ if there exists (positive) probability vector p_i such that $\hat{v}$ is still an FBA solution while satisfying $L \leq \hat{v} \leq U$ and

$$\lim_{M_i \to 0} P(-M_i \leq p_i^T \hat{S}_i \hat{v} \leq M_i) \leq 1 - 2\epsilon, \ \forall i.$$

We also partition $\hat{v}$ into $(\hat{v}', \hat{v}'')$ corresponding to the nonzero and zero entries respectively and define S' to be the submatrix of S whose columns correspond with $\hat{v}'$. The-

orem 3—presented below—provides a description of the allowed range of stochasticity RAMP problems may inhabit whilst converging to the same FBA problem under diminishing flexibility in the relaxed SS assumption.

Theorem 3. *Let $\hat{v} = (\hat{v}', 0)$, with $\hat{v}'_j$ for all j, be a solution to the FBA problem*

$$\begin{aligned} &\max_{v} \; c^T v \\ \textit{s.t.} \;\; &p_i^T \hat{S}_i v = 0, \quad \forall i, \\ &L \leq v \leq U. \end{aligned}$$

Then the scenarios of $\hat{S}_i$ are biologically possible for $\hat{v}$ if and only if for all i we have $\hat{S}_i{}' = \alpha_i e$ for some scalar $\alpha_i \neq 0$.

To summarize, the RAMP method generalizes the FBA method by introducing stochasticity in modeling parameters and relaxing the FBA modeling assumptions. RAMP problems are continuous with regards to their probabilistic modeling parameters and converge to FBA problems as these probabilistic parameters disappear. Not all probabilistic parameters have to disappear to ensure convergence, and Theorem 3 provides a classification of these parameters. While the predictive power of RAMP is similar to that of FBA, RAMP is not as computationally stable as FBA and is mathematically more complicated. In the next section, we will see how the mathematically simpler FBA formulation lends itself to the integration of enzyme kinetic data.

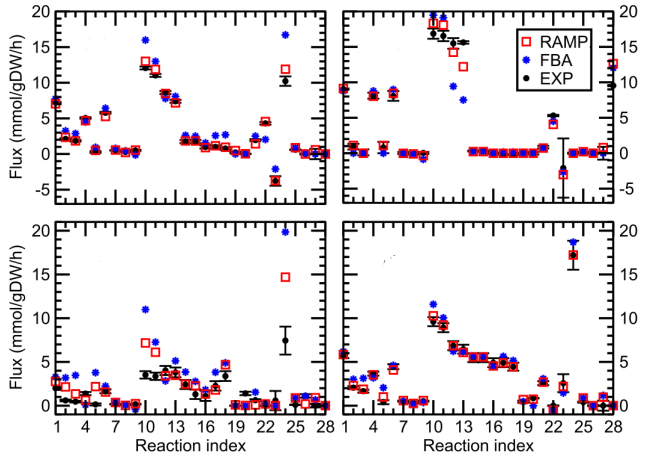


Figure 2.10: Comparison of flux vectors closest in distance to experimental flux data in RAMP and FBA solution spaces for the *E. coli* GEM iJO1366. The different subfigures correspond to different environmental conditions. *Upper left:* Aerobic batch growth. *Upper right:* Anaerobic batch growth. *Lower left:* Carbon-limited chemostat with dilution rate 0.1/h. *Lower right:* Carbon-limited chemostat with dilution rate 0.4/h. Figure reprinted from MacGillivray *et al.* [45].

2.4 Extending beyond canonical flux balance analysis

Numerous extensions to the FBA formulation have been published allowing for additional analytic features as well as higher accuracy in flux and growth rate prediction. Such extensions include dynamic FBA [35] and LK-DFBA [82], which introduce differential equations and hence time evolution to the FBA formalism, gx-FBA [68], which employs transcription data to prune the solution space, cFBA [36], which expand the FBA formalism to microbial communities, SLIMEr [74], which increases the granularity of lipid metabolism in the biomass pseudoreaction, ccFBA [37], which prunes the solution space using mass considerations of carbon uptake, and many more [38–41, 83–85]. In this section, we will only discuss FBA extensions employing enzyme kinetic data and total enzyme mass constraints. Recent extensions in this direction include GECKO [38], MOMENT [39], sMOMENT [40], FBAwMC [41], and the AutoPACMEN software toolbox [40]. One of the key achievements of these extensions is their ability to model overflow metabolism as protein reallocation strategies [38, 40]. But before we move on to describe the introduction of enzyme kinetics into the FBA formalism, we will have to introduce the notion of enzyme *turnover numbers* (TNs).

2.4.1 Michaelis-Menten kinetics

Enzymes are biological catalysts that lower the activation energy necessary for biochemical reactions to occur. Catalysts themselves do not affect the reaction equilibrium, which dictates the reaction direction, but they do affect the *reaction rate*—that is, the rate of substrate conversion—often referred to as the reaction flux in biochemical settings. Enzymes may exhibit extraordinary catalytic capabilities. An example being carbonic anhydrase which increases reaction rates by a factor of 10^7. Enzymes may furthermore exhibit remarkable specificity towards the reactions they catalyze. The effect of enzymes on the reaction rate under different conditions is studied in the field of *enzyme kinetics* [86, Chapter 6].

In general, the reaction rate, v, of a chemical reaction,

$$\mathrm{a}_1\,[\mathrm{S}_1] + \mathrm{a}_2\,[\mathrm{S}_2] + \ldots \mathrm{a}_s\,[\mathrm{S}_s] \xrightarrow{\mathrm{k}} \mathrm{b}_1\,[\mathrm{P}_1] + \mathrm{b}_2\,[\mathrm{P}_2] + \ldots \mathrm{b}_m\,[\mathrm{P}_p],$$

is given by the *rate equation*, $v = k[S_1]^{n_1}[S_2]^{n_2}\ldots[S_s]^{n_s}$, where k is the rate constant and n_i is the *partial order* of reactant $[S_i]$. As usual, the brackets denote the molar concentration of the reaction species. In simple cases, such as a *single-step reaction*, the partial orders equal the stoichiometric coefficients. However, this is not true in general, leaving the partial orders to be most reliably determined experimentally. The rate constant is a function of temperature and depends on the reaction's activation energy. Its units depend on the rate equation and are defined in each case such that the flux, v, is given in molar per second [87, Chapter 14]. As enzymes lower the activation energy, they increase the rate constant. Probing the change imposed on the rate constant using the rate equation proves exceedingly difficult, as the substrates—the reactants involved in an enzymatically catalyzed biochemical reaction—change over time. In order to describe the magnitude of which enzymes change the rate constant, we will have to constrain ourselves to the description of the initial reaction rate and make several assumptions.

The following discussion will borrow heavily from Nelson [86, Chapter 6]. Consider the biochemical reaction

$$\mathrm{E} + \mathrm{S} \underset{\mathrm{k_{-1}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \mathrm{ES} \underset{\mathrm{k_{-2}}}{\overset{\mathrm{k_2}}{\rightleftharpoons}} \mathrm{E} + \mathrm{P},$$

where E is the enzyme catalyzing the reaction with substrate S and product P, and ES denotes the transient enzyme-substrate complex. We imagine that we model an early part of the reaction in which we have an excess of substrate compared to enzyme. We assume that the rate of formation and breakdown of the enzyme-substrate complex is in equilibrium. This is argued by the fact that substrate excess will force the enzyme-substrate reaction to equilibrium quickly. We also assume that the change in substrate concentration is negligible due to its surplus. We also assume k_{-2} to be negligible due to the small amount of product present. Finally, we assume that the final step in the reaction is rate-limiting and that determining the rate of product formation—essentially the total flux of the reaction—is reduced to finding $V_0 = k_2[ES]$, where we add a subscript of zero to remind us we are dealing with the initial reaction rate. We should also stress the fact that we have assumed the partial orders of the reaction equal to their respective stoichiometric coefficients. In other words, we have assumed the kinetics of the reaction to be that of a single-step reaction.

The equilibrium assumption for the enzyme-substrate complex results in the following equation

$$k_1([E_t] - [ES])[S] = k_{-1}[ES] + k_2[ES],$$

where E_t denotes the total concentration of enzyme. Solving the equation for [ES] and defining $K_\mathrm{m} = (k_{-1} + k_2)/k_1$ gives

$$[ES] = \frac{[E_t][S]}{K_\mathrm{m} + [S]}.$$

Substituting [ES] in the rate equation for product formation gives us

$$V_0 = \frac{k_2[E_t][S]}{K_\mathrm{m} + [S]}.$$

Finally, we note that the maximum initial rate is obtained at $[ES] = [E_t]$. Defining $V_\mathrm{max} = k_2[E_t]$ per the rate equation of product formation. we arrive at the *Michaelis-Menten* (MM) *equation*,

$$V_0 = \frac{V_\mathrm{max}[S]}{K_\mathrm{m} + [S]}. \tag{2.14}$$

The constant K_m is known as the *Michaelis constant*, and enzymes adhering to the MM equation (2.14) are said to follow *MM kinetics*. Figure 2.11 depicts the initial rate as a function of substrate concentration. In unchanging conditions, an enzyme-catalyzed reaction slows down as substrate is consumed. Hence, the MM equation provides a maximal reaction rate given an initial substrate concentration under conditions in which the MM assumptions are warranted. Although we have based our discussion on simple enzyme-catalyzed two-step reactions with single substrates, many multi-steps reactions with several substrates still follow MM kinetics. For multi-stepped reactions with single

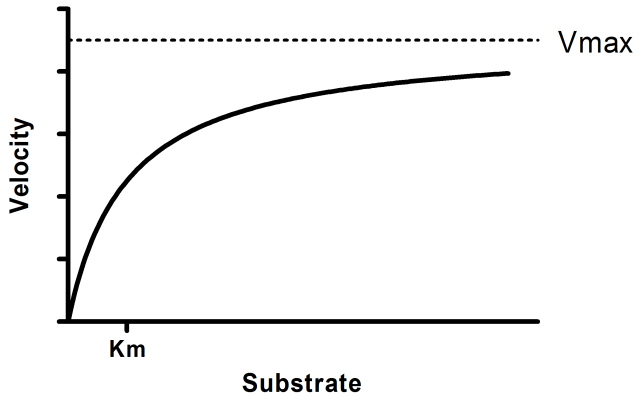


Figure 2.11: The initial reaction rate as a function of substrate concentration given by the MM equation (2.14). Note how the initial rate grows slower as the substrate concentration increases, an effect explained by the gradual saturation of the enzyme population. The value $[S] = K_{\mathrm{m}}$ is shown on the horizontal axis, as it—in general—corresponds to $V_0 = V_{\mathrm{max}}/2$ and is as such often exploited when determining K_{m} and V_{max} experimentally. Figure reprinted from Motulsky [88].

rate-limiting constants and multiple substrates, the derivation of the MM equation is analogous to ours above. But even under instances of reactions for which the MM conditions are problematic to justify, the enzyme-catalyzed reaction may follow a hyperbolic dependency like that depicted in Figure 2.11. We may thus consider these enzymes to follow MM kinetics as well. And although regulated enzymes do not adhere to MM kinetics in general, they still approach a maximum initial rate upon enzyme saturation. This observation leads to the introduction of the TN denoted by k_{cat}, which describes the limiting-rate of a reaction at saturation per enzyme concentration. The TN is also at times defined as the limiting-rate per enzyme concentration per active site. Enzymes possess a vast range in magnitude of TNs, as illustrated by Table 2.1. We must be careful to stress that TNs are not related to enzymes per se but rather to substrate-enzyme pairs.

In practice, the determination of TNs for any enzyme, regardless of its kinetics, will employ the MM equation under *in vitro* environmental conditions. Several methods for TN determination exist with varying degrees of accuracy. For example, Uludag-Demirer *et al.* [89] compared two methods for determining TNs and showed disparate errors in TN calculation with 35% and 5% errors respectively. Additionally, as is well-established in enzymology, enzyme efficiency is highly condition-dependent. Allosteric regulators, covalent modifications, pH, and temperature are examples of factors that affect enzyme catalytic efficacy. It is therefore important that TNs are determined in conditions that accurately recapture *in vivo* environments should they be used to infer meaningful biological knowledge. Despite the apparent obstacles, Davidi *et al.* [90] showed, using FBA-reliant computational analysis, that TNs determined *in vitro* are reasonable, though far from perfect, representations of *in vivo* TNs.

Table 2.1: Examples of TN values for various enzyme-substrate pairs. Table reprinted from Nelson [86, Chapter 6].

TABLE 6–7 Turnover Number, k_{cat}, of Some Enzymes

Enzyme	Substrate	k_{cat} (s^{-1})
Catalase	H_2O_2	40,000,000
Carbonic anhydrase	HCO_3^-	400,000
Acetylcholinesterase	Acetylcholine	14,000
β-Lactamase	Benzylpenicillin	2,000
Fumarase	Fumarate	800
RecA protein (an ATPase)	ATP	0.5

2.4.2 Enzymatically constrained FBA

Several extensions to FBA has been developed aiming to prune the solution space of biologically improbable fluxes using enzyme-related data [38–41]. These extensions typically introduce constraints to the FBA formulation (2.8) using individual enzyme turnover rates and total enzyme mass restrictions. We will now describe, in general, an *enzymatically constrained* FBA (ecFBA) formulation drawing primarily from the formulation presented by Sánchez *et al.* [38].

In order to accommodate enzyme kinetics into the FBA formulation, we will first define the *turnover matrix*, T, which is designed to specify the relational and quantitative dependencies between reaction fluxes and enzyme concentration. We describe three relational reaction-enzyme dependencies. Each row of T corresponds to an enzyme and each column to a reaction, hence the enzyme-reaction relationships will determine the inequalities that the rows of the turnover matrix represent.

First, we consider a simple one-to-one relationship between reaction and enzyme. Such a relationship is modeled as

$$\frac{1}{k_{cat}^{i,j}} v_j \leq e_i \tag{2.15}$$

where $k_{cat}^{i,j}$ is the TN of enzyme i in reaction j, and e_i is the concentration of enzyme i. The inequality (2.15) states that the reaction flux v_j cannot surpass the limit imposed by the enzyme concentration available to catalysis. The $(i,j)^{\text{th}}$-entry of T is then $1/k_{cat}^{i,j}$, while the other entries of that row are zeroes. We also note that the time unit of v must equal the time unit of k_{cat}.

More generally, an enzyme complex—consisting of enzymes $i^1, i^2, \ldots, i^n$—may catalyze a single reaction j. In this case, the enzyme flux v_j is bounded by the limiting subunit, which is mathematically equivalent to

$$v_j \leq \min \left\{ \frac{k_{cat}^{i^1,j}}{\rho_{i^1,j}} e_{i^1}, \frac{k_{cat}^{i^2,j}}{\rho_{i^2,j}} e_{i^2}, \ldots, \frac{k_{cat}^{i^n,j}}{\rho_{i^n,j}} e_{i^n} \right\} \tag{2.16}$$

where $\rho_{i^s,j}$ is the number of subunits of i^s in one unit of enzyme complex. Note that

constraint (2.16) encapsulates constraint (2.15) by setting $n = 1$ and $\rho_{i,j} = 1$. To preserve the linear nature of the constraints, we state constraint (2.16) as

$$T_{i^s,j} v_j \leq e_{i^s}, \forall s \in \{1 \dots n\} \tag{2.17}$$

where we have defined $T_{i,j} := \rho_{i,j}/k_{cat}^{i,j}$ to ease notation. Each of these n constraints correspond to a single row in T, as they represent separate enzymes.

Second, a single enzyme—alone or as part of one or several complexes—may catalyze several reactions; say enzyme i catalyzes reactions $j^1, j^2, \dots, j^n$. Let e_i^s be the portion of enzyme i participating in the catalysis of reaction j^s such that $e_i^1 + \cdots + e_i^n = e_i$. Simply adding the inequalities arising from these relationships will give us the following inequality

$$T_{i,j^1} v_{j^1} + \cdots + K_{i,j^n} v_{j^n} \leq e_i^1 + \cdots + e_i^n = e_i. \tag{2.18}$$

The preceding inequality will correspond to a single row in T; the enzyme portion variables were merely introduced to justify the constraint formulation.

Third, several enzymes—or enzyme complexes even—may independently catalyze a shared reaction. Such enzymes are often referred to as *isozymes*. Precaution has to be made in order to model isozyme relationships that preserve upper and lower reaction flux bounds. Let reaction j be catalyzed (independently) by enzymes $i^1, i^2, \dots, i^n$. We then introduce an intermediate pseudometabolite, dubbed x', as well as n number of pseudoreactions, named j^s, each corresponding to an isozyme i^s. We have illustrated the case of two isozymes in Figure 2.12. Each pseudoreaction is quantitatively linked to its corresponding isozyme by one of the inequalities presented above. No link is established between the original flux j with any enzyme. By not upheaving the lower and upper bound on the original reaction j, the SS assumption (2.19b) ensures that the sum of the set of fluxes $v_{j^1}, v_{j^2}, \dots, v_{j^n}$ does not exceed the upper bound of the original flux v_j. We stress that the introduction of the T matrix effectively requires the introduction of said pseudometabolites and pseudoreactions into the S matrix. Additionally, the identification of TNs can be quite time-expensive and laborious. Consequently, an automated effort has been made to the construction of GEMs accommodating the ecFBA formalism [40].

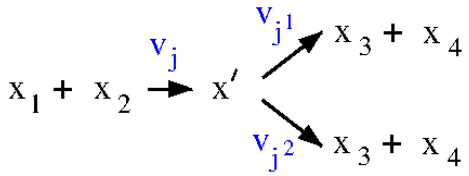


Figure 2.12: A modeling schematic for a reaction that can be catalyzed independently by two enzymes.

The turnover matrix by itself does not impose any restrictions on the flux solution space. Given available and relevant proteomic data, we may impose upper bounds on enzyme concentrations. Collecting the enzyme concentrations and their upper bounds in vectors e and E respectively, we impose the following constraints

$$\begin{aligned} Tv &\leq e, \\ 0 \leq e &\leq E, \end{aligned}$$

setting unknown enzyme bounds to positive infinity. These two constraints will together constrain the flux solution space. In reality, we may only identify upper bounds on a handful of enzymes if any; in order to further impose the solution space, we may additionally limit the total sum of the enzyme mass.

Let w be a vector containing the enzyme weights per concentration. We will refer to w as the *weight vector*. Accordingly, we may limit the dot product, $w^T e$, by an upper bound, W, to impose a maximum weight for the sum of enzyme mass. We will refer to the constraint

$$w^T e \leq W$$

as the *enzyme pool constraint* and the upper bound, W, as the *enzyme pool bound*. In contrast, we will refer to constraints defined by the turnover matrix rows as *kinetic constraints*. The turnover matrix, enzyme concentration bounds, and enzyme pool constraint are the central components of the ecFBA formalism.

Sánchez *et al.* [38] refer to the vector, e, as the *enzyme usage vector*. This is due to the fact that introduced enzyme kinetic constraints are more aptly interpreted when we consider e_i to be the active enzyme concentration. Likewise, the enzyme pool bound, W, should then be correspondingly modeled as the maximum mass amount of active enzymes. Before moving on we would like to stress that the amount of enzyme included need not be exhaustive for the formalism to be viable. We are now ready to fully state the ecFBA formulation.

The ecFBA problem—which is indeed an LP—is stated as

$$\begin{aligned} \max_v \; & c^T v && \text{(2.19a)} \\ \text{s.t. } & Sv = 0, && \text{(2.19b)} \\ & Tv \leq e, && \text{(2.19c)} \\ & w^T e \leq W, && \text{(2.19d)} \\ & L \leq v \leq U, && \text{(2.19e)} \\ & 0 \leq e \leq E, && \text{(2.19f)} \end{aligned}$$

where k and r denote the number of enzymes and reactions respectively, $T \in \mathbb{R}^{k \times r}$ denotes the turnover matrix, the enzyme usage and its concentration bounds, $e, E \in \mathbb{R}^k$, are in units mmol gDW^{-1}, the weight vector, $w \in \mathbb{R}^k$, is in units g mmol^{-1}, and the enzyme pool bound, $W \in \mathbb{R}$, in units of g gDW^{-1}. The enzyme pool constraint (2.19d) therefore represents a restriction on the total mass amount of active enzyme in grams per gram dry weight. The SS constraint (2.19b) and flux bounds (2.19e) are defined as for the FBA problem.

Though often requiring extensive manual parameter curation, several studies have demonstrated the potential of the ecFBA method. Sánchez *et al.* [38] correctly predicted the Crabtree effect—an example of overflow metabolism in baker's yeast not captured by FBA—and reported significant improvements in growth rate predictions for *S. cerevisiae*. In similar fashion, Bekiaris & Klamt [40] were able to simulate overflow metabolism and improve growth rate predictions in *E. coli*, while Massaiu *et al.* [91] found improved flux and gene-essentiality predictions for *Bacillus subtilis*. These studies effectively show that the incorporation of enzyme kinetic data may greatly improve the predictive power of the FBA formalism.

Chapter 3

Material & Methods

In this chapter, we will describe the metabolic reconstruction—as well as the tools and modules—utilized in our computational efforts. All computational endeavors were written in the Python programming language [92] and made extensive use of the Gurobi solver [51] as well as the Python packages COBRApy [93] and Pyomo [94]. The computational efforts can be assorted into three main parts. First, we implemented the RAMPER formalism for general-purpose use. Second, we investigated global TN characteristics across taxa using a comprehensive enzyme database. Third, we performed sensitivity analysis of RAMPER's TN uncertainty modeling parameters using a baker's yeast metabolic reconstruction. The scripts and GEM employed in our computational efforts have been made available at GitHub (`https://github.com/tutentaten/RAMPER`).

3.1 Genome-scale metabolic models containing enzyme kinetic data

Due to the computational instability experienced by MacGillivray *et al.* [45] using *E. coli* models with the RAMP method, we chose to focus on *S. cerevisiae* GEMs [59] for which there were easily available integrated enzymatic data. Luckily, we experienced these baker's yeast models to be more stable than their *E. coli* counterparts. The enzyme kinetic data already integrated into the baker's yeast reconstruction were extracted from the enzyme database BRENDA [95]. And as we will make further use of the BRENDA enzyme repository, we will describe the database briefly before moving on to the relevant metabolic reconstruction.

3.1.1 BRENDA

Initiated in 1987, BRENDA is an effort to provide a comprehensive database on functional and molecular knowledge of enzymes [95]. The database contains, as of 2019, a total of ~4.3 million manually-curated data entries for ~84,000 enzymes collected from

~140,000 primary literature references with an additional ~1.6 million data entries from ~3.6 million references obtained through text-mining [96]. BRENDA systematizes its repository based on EC numbers and contains enzyme-specific data on catalyzed reactions, enzyme–ligand interactions, nomenclature, taxonomy, inhibitions, mutants, protein sequence, protein structure, disease-related data, enzyme kinetic parameters, and much more [95, 96]. The content of the BRENDA database can be freely downloaded in text format at `https://www.brenda-enzymes.org/download_brenda_without_registration.php`.

3.1.2 The ecYeast8 reconstruction

The Yeast8 is a recently published consensus GEM for *S. cerevisiae* [59]. It was published concurrently with multiple Yeast8-derived model variants in addition to a database consisting of 3D yeast metabolic protein structures. These derived models include GEMs tailored for the creation of strain-specific models as well as GEMs accommodated with enzyme kinetic data. Respective to its predecessor Yeast7, the Yeast8 model has extended the biomass equation by nine metal ions and eight cofactors, added 37 transport reactions, updated GPR associations, and introduced the SLIMEr formalism [74]. The Yeast8 models showed significant improvement over their predecessors in GEM quality assurance tests per the MEMOTE software suite [97]. The Yeast8 model and its ecosystem are made freely available at GitHub (available at `https://github.com/SysBioChalmers`), which provides a version control system allowing users to follow and access updates to the Yeast8 modeling ecosystem. The newest version, Yeast8 v8.3.4, contains 2691 metabolites, 3991 reactions, and 1149 genes.

One of the GEMs equipped with enzyme kinetic data in the Yeast8 ecosystem is designed for ecFBA simulations under oxygen- and glucose-rich batch conditions (available at `https://github.com/SysBioChalmers/ecModels/blob/master/ecYeastGEM/model/ecYeastGEM_batch.xml`). It is precisely this reconstruction we will employ in our computational analyses. In order to ease reference, we will simply refer to it as the *ecYeast8 model*. The model contains an enlarged stoichiometric matrix according to the GECKO formalism [38]—containing both the stoichiometric and turnover matrix—and introduces 968 enzymes as well as 521 pseudometabolites to account for enzyme-reaction relationships. The TNs were automatically curated from BRENDA [95] and were then fitted to accommodate the experimental growth rate in minimal glucose media under oxygen-rich batch conditions. The automatic curation picks the highest TN for each enzyme, whereas the fitting procedure iteratively changes TNs of enzymes found to heavily affect the predicted growth rate. Finally, the enzyme pool bound is fitted under the same oxygen- and glucose-rich conditions.

3.2 Computational tools

Extraction of enzymatic data from the BRENDA database [95], implementation of the RAMPER model, and subsequent computational experiments were all performed using the Python programming language [92]. Of special importance to these implementations are the Pyomo [94] and COBRApy [93] packages, where the former is a software suite for

constructing and solving optimization problems and the latter is a package for constraint-based reconstruction and analysis of metabolic reconstructions. As the COBRApy package is a Python extension of the COBRA toolbox [77] for MATLAB [98], we will briefly describe the MATLAB toolbox before introducing COBRApy.

3.2.1 Python

Python [92] is a very popular, high-level, multi-paradigm programming language known for its clear, intuitive syntax. As a general-purpose programming language, Python has a large set of native data types and classes. Python is a portable programming language available in Windows (2000 and onwards), Mac, and several Unix variants [99]. It is open-source and enjoys a large community following. Thus, numerous Python packages are available and shared readily among its users. Two such packages were used extensively in the implementation of RAMPER and subsequent computational experiments, namely, COBRApy and Pyomo.

The COBRA Toolbox and COBRApy

The COBRA Toolbox [77] is a MATLAB software package for the reconstruction and analysis of GEMs. The Toolbox allows the user to input SBML-formatted models, perform FBA, gene deletion experiments, network module identification, and much more. The COBRA Toolbox converts the SBML files into MATLAB data types containing the necessary matrices and vectors needed to define the FBA optimization problem. Additionally, the user may access GPR associations and metabolite/reaction/gene identifiers as MATLAB data structures. In general, the COBRA toolbox is a software suite designed for basic and advanced methods in the quantitative modeling of metabolic networks and is tailored specifically for GEMs. The COBRA Toolbox is part of the openCOBRA project—which includes COBRApy—and serves as a repository for genome-scale metabolic modeling methods.

COBRApy is an object-oriented implementation of the basic COBRA Toolbox utilities for the Python programming language. The object-oriented implementation was designed to accommodate the growing incorporation of complex omics data to constraint-based methodologies. The classes found in COBRApy are Model, Metabolite, Reaction, and Gene. These classes form a hierarchy and are interconnected by their class methods. COBRApy also reads SBML-formatted GEMs but in a more general-purpose, intuitive-driven manner per its class hierarchy. COBRApy is part of the openCOBRA project and contains a module for direct interaction with COBRA Toolbox utilities [93].

Pyomo

Pyomo is an open-source Python package for composing and solving mathematical optimization problems. The key goal of the Pyomo package is to provide a similarly intuitive tool for writing and solving optimization programs as does algebraic modeling languages, such as AMPL and GAMS, but having the advantage of being embedded within a general-purpose, extensible, high-level programming language. One of the key advantages of algebraic modeling languages is the separation of abstract models and concrete model

instances. That is to say, classes symbolically representing optimization programs are defined separately from actual, parameterized optimization problems. Hence in Pyomo, abstract models are constructed first before instantiating concrete models by supplying such abstract models with parameter data. Another key aspect of Pyomo is the separation of model construction and problem solving. Pyomo supports a wide range of both open-source and commercial optimization solvers, allowing Pyomo models to be solved using a variety of different solvers. This is in contrast to other optimization programming languages that may only provide an internal optimization solver. Pyomo is among the most versatile optimization tools available and allows the user to define a wide range of optimization problems ranging from linear to nonconvex optimization problems. As Pyomo can be integrated with a repertoire of solvers, the user may identify a solver handling the specific type of optimization problem to be solved. This is important, as several solvers only handle certain categories of optimization programs. The efficacy of solving the problems themselves will naturally depend upon the chosen solver [94].

3.2.2 Optimization solvers

A wide range of open-source and commercial optimization solvers are available each specialized to certain classes of optimization programs. Examples of solvers equipped with Python APIs include CPLEX [53], FICO EXPRESS [52], MOSEK [57], and Gurobi [51]. Due to its intuitive documentation, free academic license, support of SOCPs, and previous user experience, we chose to utilize the Gurobi Optimizer in our computational efforts.

Gurobi Optimizer

The Gurobi Optimizer [58] is a popular optimization solver supporting a wide range of programming languages, such as Python, C++, Java, C, MATLAB, R, AMPL, and GAMS. Although the Gurobi Optimizer solves a broad spectrum of optimization problems, such as LPs, mixed-integer linear programs, quadratic programs, mixed-integer quadratic programming, QCPs, and mixed-integer quadratically constrained programs, its most efficient algorithms are designed for LPs, quadratic programs, and mixed-integer programs. The Gurobi Optimizer is covered by a free academic license and includes an extensive reference manual. The Gurobi solver also provides the user with the possibility of tuning numerous solver algorithm parameters. As the implementation of the RAMPER method allows the user to specify such solver parameters, we will very briefly describe some of the Gurobi solver's parameter settings.

The RAMPER method is formulated as an SOCP but is in practice written as a QCP (see Section 2.1.1). The FBA and ecFBA problems are, on the other hand, LPs. Our implementation of the RAMPER method handles both deterministic FBA/ecFBA problems and stochastic RAMP/RAMPER problems. Hence, both LP and QCP solver settings will be of relevance to us.

Gurobi's LP and QCP solver algorithms share a few tuning parameters. The *Numeric Focus* attribute changes the numerical accuracy in the solver's calculations and may affect numerical stability. The *Aggregate* and *Presolve* attributes both change the presolve settings and may provide a trade-off between computational time and tractability. For some programs such as the LP, the user may have a number of solver algorithms to choose from.

In Gurobi, LPs may be solved either by the primal simplex method, dual simplex method, or by an interior-point method. The interior-point method may or may not include variants of so-called *crossover methods*. The crossover method is used to find a vertex solution, as is found in the simplex method, from an interior-point solution, which in theory will not be a vertex itself. If not applied, the interior-point LP method will, strictly speaking, always find a suboptimal solution. The QCP solver provides an interior-point method. The interior-point methods of both LPs and QCPs rely on the duality gap (see Section 2.1) to identify solutions. A solution is identified should the duality gap become zero, and the user may alter the absolute value tolerance in primal and dual solution difference. Such tolerances are referred to as *convergence tolerances*. Other parameters may also tune the interior-point method. The *Barrier Corrections* attribute determines the number of computations performed at each iteration step in interior-point methods, whereas the *Barrier Homogeneous* attribute may enforce another variant of interior-point methods that in principle is better at differentiating between infeasible and unbounded problems but may suffer from computational instability [51].

3.3 Computational implementation

The RAMPER model was implemented as to be able to convert COBRApy Model objects into concrete Pyomo models which are then solved using the Gurobi Optimizer. The code consists primarily of two functions—one parsing metabolic reconstruction data and another constructing and solving Pyomo model instances—and may perform simulations with and without parameter uncertainty. Additionally, computation times for the RAMPER implementation were calculated for both RAMPER and ecFBA problems. We also probed the enzyme database BRENDA [95] for TN entries and performed a sensitivity analysis of stochasticity in kinetic constraints using the RAMPER implementation.

3.3.1 Implementing RAMPER

The scripts we will describe shortly input GEMs and performs simulations of the metabolic reconstructions modeled either with uncertainty, as SOCPs (RAMP/RAMPER), or without, as LPs (ecFBA/FBA). Hence, our code can perform simulations applying the FBA, ecFBA, RAMP, and RAMPER methods. We decided to implement our code in Python using the Pyomo package and used the Gurobi Optimizer as our default solver. Note that it would not be difficult to implement the option of choosing different solvers due to Pyomo's diverse compatibility with optimization solvers. Two primary functions were written, called *createUncModel()* and *solveUncModel()*, and both rely on functions imported from the module *uncUtils.py*. Both functions input a dictionary structure—referred to as the *model settings*—containing the tuning parameters for the Gurobi solver (described above in Section 3.2.2) and parametric uncertainty specifications. The overall process is illustrated by Figure 3.1. We will now describe the two primary functions and then briefly describe the functions found in the *uncUtils.py* module.

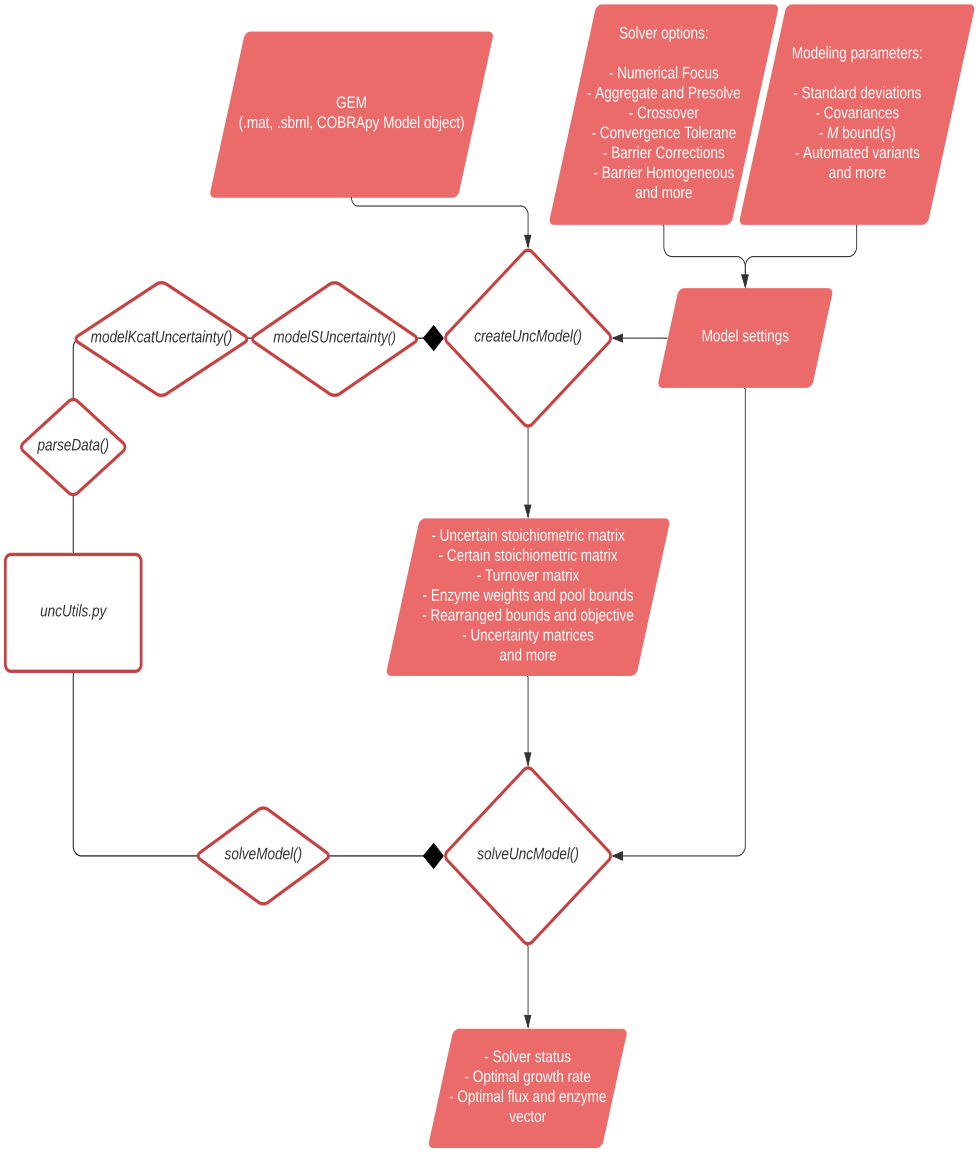


Figure 3.1: Flowchart illustrating the implementation of the RAMPER formalism. The diamond-shaped boxes correspond to functions, diamond-shaped arrows show destinations of imported functions, squares correspond to modules, and parallelograms correspond to both input and output. On the whole, GEMs are used to perform FBA/ecFBA/RAMP/RAMPER simulations per user-set modeling parameter specifications. The user need only specify the modeling parameters and must not specifically state the constraint-based method to be utilized.

createUncModel()

The *createUncModel()* function inputs GEMs—either as SBML files, MATLAB structures per the COBRA Toolbox, or as Model objects per COBRApy—and the model settings. The precise parametric uncertainty may either be handled automatically by the code or set specifically by the user.

The user may specify the uncertainty with a matrix—of same dimensions as the enlarged stoichiometric matrix per the GECKO formulation—denoting the uncertainty in each coefficient. Additionally, the user may input a list specifying the covariance among coefficients. The matrix specifying the coefficients' uncertainty will later be referred to as the *deviation matrix*, as its elements correspond to standard deviations (to be discussed more precisely in Chapter 4).

The automatic uncertainty specification will attempt to identify uncertain stoichiometric coefficients using simple heuristics and model them uncorrelated with default deviation magnitude. The default magnitude is different for stoichiometric coefficients and TNs, and can also be set manually.

The user may also specify TN uncertainty inputting upper and lower percent bounds. The percent range is then used to assign the uncertainty magnitude for each TN assuming a linear correlation between turnover magnitude and uncertainty, with the largest TN receiving the largest percent uncertainty and the lowest TN receiving the lowest. Note that this is not the same model of uncertainty we will describe in Section 4.4.1 but is rather a modeling relic.

The uncertainty model of TNs as described in Chapter 4 were rather introduced using the deviation matrix input, as it is not yet introduced into *createUncModel()* as an optional, automatically generated uncertainty setting.

The output is a dictionary containing the extracted data needed to instantiate the concrete Pyomo model, which is then either an LP or SOCP dependent on the specified uncertainty parameters. The SBML input is simply read as a COBRApy Model object. Hence, the script does not "read" SBML files in the strict sense. A great advantage of using COBRApy objects is that we may use the COBRApy utilities on the metabolic models prior to creating the corresponding Pyomo model. Changing the Pyomo concrete model can be done using Pyomo functions, but scripts written with constraint-based analysis in mind should be implemented as to intuitively perform gene-knockouts, change simulation environments, and tune model parameters without having to change the original GEM input and subsequently rerun *createUncModel()*. Of the two main functions, *createUncModel()* is clearly the most time expensive. This is not surprising as the extraction of necessary data is indeed somewhat convoluted. We will now briefly describe this process and elaborate on it later when addressing the *uncUtils.py* module.

First, the format of the GEM is determined automatically. As mentioned, the SBML format is converted into a COBRApy Model should that be the case. The first parts of the data extraction depend on whether we are dealing with a MATLAB data structure or COBRApy Model, but these are technical differences and will not be discussed. Second, if TNs are included in the enlarged stoichiometric matrix, it is split into a turnover matrix and a stoichiometric matrix. Further, if coefficients in the stoichiometric matrix are to be modeled with uncertainty, then the stoichiometric matrix is further split into certain and uncertain matrix counterparts. The resulting right-hand side of the constraints—as well

as the objective and bound vectors—must then be changed accordingly. The enzyme pool constraint is handled separately by extracting the corresponding row in the enlarged stoichiometric matrix. All these matrices are handled as sparse matrices saving significant computation time. The matrices and vectors describing the constraints, both SOCP and LP, are set as output. Third, the specified parametric uncertainty is read and initializes the construction of uncertainty matrices, which themselves end up part of the output. As the uncertain stoichiometric and kinetic constraints are handled separately, we create two corresponding separate collections of uncertainty matrices. These collections are themselves large matrices separated by indices stored in separate lists during the matrices' construction.

During the whole process a log file is being written. If an error is caught while calling *createUncModel()*, a message will be printed to screen and the log file will specify which function in *uncUtils.py* caught it. The user may themselves—by configuring the model settings—decide whether or not a log file will be written at all, the name of the file, and whether or not messages will be printed to screen.

solveUncModel()

The *solveUncModel()* inputs both the output of *createUncModel()* and the model settings (see Figure 3.1). Initially an abstract Pyomo model is constructed—in QCP form, as explained in Section 2.1.1—after which input is used to create a concrete Pyomo model.

The concrete Pyomo model is then solved according to input parameters specified in the model settings, and the solution provided by the Gurobi Optimizer is then used to extract the resulting output. Note that the Gurobi solver will recognize the QCP to be equivalent to an LP should that be the case and then subsequently employ the more efficient LP solving methods instead. The user may themselves set the desired LP solver algorithm in the model settings should they not desire to apply the default simplex method.

The output of the script can be adjusted by the user but is typically the resulting objective value and flux solution. Other provided outputs include the certain and uncertain stoichiometric matrix as generated by *createUncModel()* and other model data, though these outputs were originally meant to be used for debugging purposes. A log file is generated in this function analogously to *createUncModel()*.

uncUtils.py

The script *uncUtils.py* contains six functions each of which we will now briefly describe. All functions input the model settings in addition to some function-specific inputs which will be mentioned whenever necessary.

The functions *declareNone()* and *mesg()* are small functions that simply aid us in initializing function output and updating the log file instantiated by *createUncModel()* and *solveUncModel()*. The two functions are used ubiquitously by *createUncModel()*, *solveUncModel()*, and the other functions in *uncUtils.py*.

The function *parseData()* inputs the metabolic reconstruction and parses the potentially enlarged stoichiometric matrix into its corresponding canonical stoichiometric matrix and turnover matrix. The upper and lower bounds on the enzyme and flux variables as well as the objective vector must be changed accordingly. Reactions arising from the

SLIMEr formalism [74] are identified, as well as the enzyme pool constraint row. The function has different pipelines for handling different metabolic reconstruction formats. The metabolite GEM identifiers are used to identify the kinetic and enzyme pool constraints of the enlarged stoichiometric matrix. It does so by finding relevant keywords often used to describe these pseudometabolites. The user may, and should, verify that the reference keywords in fact recognize the correct constraints. Should that not be the case, the user may reconfigure these reference keywords manually in the model settings. Additionally, should the user have submitted a deviance matrix, maps correlating the rows of the enlarged stoichiometric matrix to the corresponding rows of the deviance matrix are instantiated. These are mandatory should we be able to assign the deviance matrix coefficients to the correct parameters of the stoichiometric and kinetic constraints.

The function *modelKcatUncertainty()* inputs the turnover matrix constructed by *parseData()* and creates the uncertainty matrices corresponding to the kinetic constraint per the specified uncertainty parameters. The function is of course skipped should there either be no turnover matrix produced or should there be no uncertainty specified for any TN. Due to the different ways in which the user may specify uncertainty in enzyme kinetic parameters, the function relies on if-else branching based on these specifications.

The function *modelSUncertainty()* inputs the sparsed stoichiometric matrix and the corresponding flux bounds and subsequently splits the two into certain and uncertain counterparts. Finally, uncertainty matrices for the uncertain stoichiometric matrix are constructed. The very simple heuristic designed to find uncertain coefficients—should the automatic stoichiometric uncertainty option be selected—locates decimal coefficients, except for the common rational $1/2$, and skips SLIME pseudoreactions, as these are not constructed using uncertain data.

The function *solveModel()* first assembles an abstract Pyomo model corresponding to the assigned parametric uncertainty. Then, the concrete Pyomo model resulting from the certain and uncertain stoichiometric matrices, turnover matrix, uncertainty matrices, weight vector, and enzyme pool bound is finally instantiated. The concrete Pyomo model is then subsequently solved by the Gurobi Optimizer with the solver parameters specified by the model settings. Importantly, the solution provided by the Gurobi Optimizer is identified as optimal, suboptimal, or infeasible/unsolvable. In the case of optimal and suboptimal solves, the resulting predicted growth rate is printed to screen unless instructed otherwise by the user.

Note on computational stability

When employing the ecYeast8 model—and other baker's yeast GEMs for that matter—we experienced less computational instability than what has been reported previously for RAMP [45]. We did however suffer similar instability upon using *E. coli* reconstructions, but as we focused our efforts on ecYeast8, computational instability did not present itself as an insurmountable problem. We did nonetheless experience instability performing FVA (not presented) and TN uncertainty sensitivity analysis. More precisely, we would experience that the Gurobi solver failed for some problems when solving hundreds of RAMPER problems with varying parameters. By enforcing the Homogeneous Barrier solver attribute, we could often verify that the instability stemmed from either empty solution spaces or unsolvable problem instances. Yet at other times, we could not determine the

source of computational instability. For these reasons, the default Gurobi settings as set by the RAMPER implementation is somewhat more lenient than the original default Gurobi settings, as we impose the strictest presolve settings and set maximum numerical accuracy. Additionally, we increased the convergence tolerance to 10^{-4} from 10^{-8} when performing large-scale simulations.

But more importantly, we experienced incompatibility with certain versions of the Gurobi Optimizer. That is, certain Gurobi versions were not unable to solve any given RAMPER problem. Whether this is caused by more global problems shared among SOCPs or is specific to RAMPER problems is not known to us. We did nevertheless identify a Gurobi version stable enough for our intended use. Therefore, all subsequent analyses and computations with the RAMPER method were performed with the Gurobi Optimizer version 7.5.2 released in 2017. By utilizing said version we did not experience any instability beyond what has been described in the above paragraph.

3.3.2 Computational speed

The computational speed of the RAMPER implementation was compared between ecFBA and RAMPER formulations (see Section 4.3) and was performed using the *computation_times.py* script. Computation times were determined for the *createUncModel()* and *solveUncModel()* functions as well as the Pyomo model instance construction and the Gurobi solver. The execution times were determined using the Python time module and were calculated as the average over 100 simulations for RAMPER and ecFBA each. All computations were performed with the ecYeast8 reconstruction.

3.3.3 Retrieving and investigating turnover numbers

After extracting TNs from the enzyme repository BRENDA [95], we performed regression analysis on the mean and standard deviation of enzyme-substrate pair TNs, where we included entries from all taxa represented in the enzyme repository (see Section 4.4.1). Enzyme data extraction was performed using the function *retrieveTN()*, whereas the regression analysis and subsequent plotting of the data were done using the *plotTN.py* script.

retrieveTN()

The function *retrieveTN()* retrieves enzyme data from the freely available BRENDA text file [95] and extracts TN entry values along with other pertinent data. The text file sorts enzymes by their EC number and contains diverse information on each enzyme, including kinetic data. Each enzyme entry contains a TN catalog—composed of TNs gathered from various sets of sources—and each entry of the TN catalogs contains descriptions of experimental conditions, relevant substrate names, and enzyme-dependent organism reference codes. The *retrieveTN()* function sorts the enzyme data by EC numbers and removes TNs indicated to be determined under impertinent experimental conditions. It does so by probing the condition descriptions for certain keywords. For instance, the function omits TNs that do not include ”wild”, ”wild-type”, or ”native” in their description and skips entries containing ”mutation”, ”recombinant”, ”mM”, ”DTT”, ”cleaved”, and other similar keywords. As entry descriptions are prone to typing errors, we also account for certain

misspelled keywords. The user may additionally include further restrictions on extracted TNs such as temperature and pH intervals. If specified, the pH and temperature values are subsequently identified in the condition description entries. The user may also limit the search to a particular organism, effectively excluding TNs determined in other species. As the enzyme entries contain enzyme-specific organism reference codes, these references must be resolved for every single enzyme. The extracted entries of *retrieveTN()* are ultimately stored in a two-dimensional list, where row entries correspond to enzymes and column entries specify substrate names, condition descriptions, and TN values. Note that in the computational analysis presented in Section 4.4.1, we included TN entries from all taxa determined in pH interval 6–8 and temperature interval 20–30°C.

plotTN.py

The script *plotTN.py* plots the mean over the inverse enzyme-substrate TN entries against their standard deviation and performs linear regression using the Scipy Python package. The script uses the extracted BRENDA data to sort TN entries according to enzyme-substrate pairs and excludes pairs with single TN entries. The script also provides two plots. The first of these plots depicts the mean against standard deviation on log-log axes and includes a regression line. The second plot depicts the logarithmic ratio of standard deviation to mean against standard deviation and does not include a regression line. The second regression line performs very poorly on statistical tests and is as such deemed superfluous.

3.3.4 Probing sensitivity in turnover number uncertainty

The scripts *geckoKcat.py*, *ramperKcat.py*, and *regRamperKcat.py* utilize the RAMPER implementation to simulate gradually stricter TN modeling parameters (see Section 4.5). The *geckoKcat.py* script incrementally increases enzyme TNs applying the ecFBA method, while *ramperKcat.py* and *regRamperKcat.py* gradually increase uncertainty in enzyme TNs employing the RAMPER method. The *ramperKcat.py* script selectively imposes uncertainty on single enzymes one at a time, whereas *regRamperKcat.py* assumes uncertainty across all enzymes using the regression analysis performed by *plotTN.py* (the regression TN uncertainty model will be described in Section 4.4.1). As these scripts are highly similar, we will only describe the components of *regRamperKcat.py* in detail. The minor differences between the scripts will be summarized briefly afterward.

The *regRamperKcat.py* script consists of the *findMaxStd()* function and a *main()* function. The former identifies the maximum standard deviation applicable to a given enzyme's TN collection without predicting growth rates below a user-defined cutoff. The main function instantiates and applies the deviation matrix constructed per the regression model found using *plotTN.py*, subsequently runs the *findMaxStd()* function in parallel, and concludes by saving the results for later analysis. The script *plotKcatStd.py* plots the results produced by *regRamperKcat.py*, whereas the *comprMaxDev.py* script compares the results of distinct *regRamperKcat.py* experiments. The *comprMaxDev.py* script orders the enzymes by their maximum acceptable TN deviation and prints the ordered lists to screen for manual comparison. The scripts *plotKcatStd.py* and *comprMaxDev.py* are trivial in their implementation and will not be discussed in any more detail.

findMaxStd()

The function *findMaxStd()* uses the parameter output from *createUncModel()*, a growth rate cutoff, a list of constants, a maximum iteration number, a list of indices corresponding to the protein pseudometabolite rows of the stoichiometric matrix, and an iteration number indicating the index of an individual enzyme. The last argument is essential in order to parallelize the function using the Python multiprocessing package. The *findMaxStd()* function uses the protein row indices to map the iteration number unto the deviation matrix and multiply the TN parameters of that row by the next number in the list of constants. For each constant the growth rate is computed, and the function returns the first constant causing the growth rate to drop below the cutoff. The function returns the last constant computed should no constants make the cutoff or should the maximum number of iterations have been met. It can later be verified whether or not the cutoff was exceeded by considering the metadata saved by the *main()* function as described below. We would like to note that although the implementation of an upper bound on the number of iterations is useful for general purposes, we did in fact not specify a maximum iteration number in the computational experiments presented in Section 4.5.

main()

In the *main()* function the specified metabolic reconstruction is read, and the corresponding deviation matrix is generated according to the regression analysis performed by *plotTN-.py*. During the construction of the deviation matrix, the protein pseudometabolite rows of the stoichiometric matrix are identified and stored in an index list to be accessed by *findMaxStd()*. The script then parallelizes the *findMaxStd()* function over every enzyme. Metadata regarding the specific computational experiment are saved separately from the results obtained from the *findMaxStd()* parallelization and include the list of constants, maximum number of iterations, GEM name, and regression model parameters. It is worth mentioning that due to incompatibilities with the multiprocessing package the metabolic model must be read by COBRApy each time the *findMaxStd()* is called. Computation time could be significantly reduced were we to circumvent this incompatibility, but our efforts in doing so proved ultimately unsuccessful.

Differences between *geckoKcat.py*, *ramperKcat.py*, and *regRamperKcat.py*

The *ramperKcat.py*, as opposed to *regRamperKcat.py*, assumes TN uncertainty for only a single, given enzyme at a time. Therefore, we instead instantiate the deviation matrix as the zero matrix and consequently add the iterate constants to the row coefficients as opposed to multiplying them. On the other hand, *geckoKcat.py* employs the ecFBA method, and hence the iterate constant must be added to the enlarged stoichiometric matrix as opposed to the deviation matrix. In all three scripts, the model settings are reset to default in each iteration of *findMaxStd.py* so that the effect of adding, or multiplying, iterate constants not accumulate.

Chapter 4

Results

We begin this chapter motivating the consideration of enzyme kinetic parameter uncertainty before proceeding to introduce RAMPER using three different formulations. We then briefly describe the issue of interpreting RAMPER solutions in light of these different formulations. Furthermore, we prove that RAMP's continuity and convergence theorems extend into RAMPER, effectively demonstrating that RAMPER in fact generalizes ecFBA per the RAMP formalism. Next, we briefly compare the computational speed of the RAMPER implementation under deterministic and stochastic modeling assumptions. We then probe a comprehensive enzyme data repository and harness its data to establish a provisional model for TN uncertainty. Thereafter, we perform sensitivity analyses of TN uncertainty in the RAMPER method, which we conduct both with and without applying said model of TN uncertainty. Finally, by comparing the sensitivity analysis results with a similar analysis employing the ecFBA method, we argue that the RAMPER method indeed does provide a distinct and novel link between proteomics and constraint-based metabolic modeling.

4.1 Robust analysis of metabolic pathways under enzymatic regulation

Naturally, the RAMPER method borrows heavily from RAMP (2.11) differing in its integration of enzyme kinetic data analogous to the ecFBA method (see Section 2.4.2). But in contrast to the development of RAMP per MacGillivray *et al.* [45], we are going to consider continuous rather than discrete probability distributions in modeling parameters. We will throughout this section follow the notational scheme of the ecFBA formulation (2.19) as to avoid unnecessary repetitions of definitions.

Before formulating the RAMPER method, we will first motivate the introduction of continuous stochasticity in modeling parameters. We then go on to discuss interpretations of the method under parameter stochasticity assumptions. Finally, we will briefly consider whether available experimental data are amenable to our RAMPER interpretation.

4.1.1 Stochasticity in modeling parameters

Turnover numbers and the enzyme pool bound

Every enzyme-catalyzed biochemical reaction is, as described by the ecFBA method, limited by maximum enzyme catalytic efficacies and enzyme concentrations. But other biological factors may in fact constrain reaction fluxes even further. As pointed out by Davidi *et al.* [90], factors such as substrate concentration, substrate localization, enzyme localization, and post-translational enzyme regulation may impose stricter flux bounds than what the ecFBA kinetic constraints suggest. Moreover, Zotter *et al.* [100] found large cell-to-cell variability when measuring the *in vivo* catalytic efficiency of TEM1-β-lactamase, effectively showing that enzyme reaction-limiting factors may exhibit intercellular variability even within homogeneous cell cultures.

Motivated by the apparent intercellular variability of enzyme efficiency, we define the *effective TN* of enzymes to represent their *in vivo* maximum substrate conversion rate. Correspondingly, we define the *effective turnover* (ET) matrix analogously to the turnover matrix having replaced the inverse TN coefficients by inversed effective TNs.

Consider now a homogeneous cell culture optimized to its environment. At any given point in time, the cell culture possesses a collection of ET matrices in which each ET matrix belongs to a single cell of the culture. As cell cultures commonly accommodate millions of cells, we may apply the central-limit theorem and assume the sample mean of the ET matrix collection to be normally distributed. More precisely, we assume the sample means of the ET matrix coefficients to be distributed normally.

Define now the *random effective turnover* (rET) matrix, K, as the sample mean of the ET matrices. In other words, we define the rET matrix as the matrix whose coefficients are the corresponding random variable sample means of the ET matrix coefficients. Note that that for a given flux vector, v, the matrix-vector product Kv represents a vector of enzyme concentrations, but as we have established K to be a matrix of normal variables, the product Kv is, in fact, a vector of enzyme concentration distributions. Due to the linearity of normal distributions, it follows readily that the predicted concentration distributions are normal as well. As we now argue, the consideration of enzyme concentration distributions, as opposed to numerical enzyme concentrations, is more concurring to biological reality.

Fluctuating gene-expression [101] and asymmetric partitioning of molecular species during cell division [102] are examples of biological mechanisms that introduce stochasticity in the subcellular localization and concentration of enzymes. Soltani *et al.* [103] employed a computational approach to investigate mechanisms of protein copy number stochasticity and indicate approximately normal distributions of protein levels throughout the cell cycle. Hence, it appears that the introduction of uncertainty into the turnover matrix uncertainty confers to predictions of protein concentrations that are consistent with our understanding of intercellular protein variability. In contrast, the interpretation of the enzyme usage vector per Sánchez *et al.* [38] mandates that the ecFBA method predicts numerical lower bounds on enzyme concentrations. Following observations made by MacGillivray *et al.* [45], the lower enzyme concentration bound is more aptly interpreted as the average lower bound over the cell culture. Therefore, the ecFBA method does not acknowledge the periodic change in protein concentration provoked by cell cycle progression and fails thus at capturing populational enzyme characteristics.

Extending the above discussion from single enzyme concentrations to the cell proteome, it is clear that the overall enzyme content is prone to intercellular variability as well. We may additionally consider the total enzyme content to be approximately normal, as it is represented by the sum over normally distributed enzyme concentrations, and therefore assume normality in the enzyme pool bound as well. In contrast, the enzyme molecular weights are calculated summing the amino acid molecular weights of the protein primary structures and are as such not considered stochastic.

Let us reiterate our motivation for introducing stochasticity into enzyme kinetic parameters. First, cell-to-cell variations exist implying variance in the upper bounds of enzyme-substrate conversion rates. Second, by reinterpreting the turnover matrix as the ET matrix, we note that all cells in any given culture possess their own ET matrix. Third, by considering the sample mean of TNs across the culture, we shift focus towards the rET matrix composed by the sample mean of the ET matrix population. Four, having introduced stochasticity in the turnover matrix in terms of the rET matrix, we provide a more biologically apt interpretation of predicted enzyme concentrations.

Stoichiometric coefficients and flux bounds

It is not unreasonable to assume that the different cells in a culture will possess distinct biomass precursor needs. At the very least, it would be deemed by most biologists unreasonable to assume the cells to need precisely the same amount of precursors to produce the exact same amount of biomass. Indeed, biomass precursors consumption changes during the cell cycle, and at any given point in time the cells of cell cultures exist at different phases of the cell cycle. We therefore argue that biomass precursor coefficients are reasonably modeled as stochastic modeling parameters in a similar fashion to the enzyme kinetic parameters described above. Flux bounds of metabolic reconstructions are most commonly imposed to limit the maximal uptake rate of nutrients. Stochastically fluctuating protein concentrations and post-translational regulation may affect membrane transport proteins and may lead to changes in nutrient uptake rates. Additionally, cell cultures may experience fluctuations in concentration, localization, and diffusion properties of extracellular nutrients which can cause cells to be subject to different immediate microenvironments. The upper bound of nutrient uptake may then be prone to intercellular variability and is perhaps more aptly interpreted stochastically. Moreover, flux bounds may be imposed for reasons other than the modeling of nutrient uptake. For example, the GAM/NGAM pseudoreaction fluxes are typically restricted and are in some cases even fixed to a given numerical value. As we believe it is quite reasonable to presume differences in ATP-cost across cell cultures, we deem it sensible to confer uncertainty to the GAM/NGAM flux bounds as well.

Having motivated our introduction of stochasticity into certain categories of canonical FBA coefficients, we argue that—in a similar vein to the turnover matrix—we may consider a stochastic counterpart to the stoichiometric matrix composed of the sample mean of stoichiometric coefficients. Again, arguing by the central-limit theorem, we assume the stochastic sample mean coefficients to be normally distributed. Equivalently, we assume certain coefficients of the flux bound vectors to be normally distributed as well.

Now that we have advocated the stochastic consideration of stoichiometric and kinetic modeling parameters, we may proceed to introduce the RAMPER method formulation.

4.1.2 Stochastic formulation

We will now reconsider the ecFBA formulation (2.19) in light of stochastic modeling parameters. As has been described in the previous section, we would like to instill stochasticity in the experimentally inferred coefficients of the stoichiometric matrix, the enzyme kinetic data of the turnover matrix, the enzyme pool bound, and eventually a subset of the flux and enzyme concentration bounds. As we motivated above, the stochastic parameters are assumed normal and are interpreted as parameter sample means across cell cultures.

We split the stoichiometric matrix into two matrices, S^c and S^u, in which the former consists of rows absent of random variable parameters and the latter containing rows that do. We additionally replace the turnover matrix by the rET matrix denoted by K. Consider now the following probabilistic program

$$\max_v \; c^T v \tag{4.1a}$$

$$\text{s.t. } S^c v = 0, \tag{4.1b}$$

$$P(-M_i \leq S_i^u v \leq M_i) \geq 1 - 2\epsilon, \; \forall i \in \mathcal{S}, \tag{4.1c}$$

$$P(K_i v \leq e_i) \geq 1 - \epsilon, \; \forall i \in \mathcal{E}, \tag{4.1d}$$

$$P(w^T e \leq W) \geq 1 - \epsilon, \tag{4.1e}$$

$$P(L \leq v \leq U) \geq 1 - \epsilon, \tag{4.1f}$$

$$P(0 \leq e \leq E) \geq 1 - \epsilon, \tag{4.1g}$$

where $\mathcal{E}$ is an index set of enzymes, $\mathcal{S}$ is an index set of the rows of the uncertain stoichiometric matrix, K_i and S_i^u denote rows of their respective matrices, and $\epsilon \in \mathbb{R}$ is a (small) positive scalar.

These constraints mandate that the cells optimize biomass under metabolic states satisfying the kinetic constraints under SS to high probability. It is important to be fully aware that is the parameters and not variables that possess stochasticity. In fact, RAMPER suggests an optimization in which we decide the coefficients (fluxes and enzyme concentration) of parameter normal distributions (biomass coefficients, TNs, enzyme pool bound, and variable bounds). We will come back to this after introducing the RAMPER's SOCP formulation. Finally, we would like to point out that the constraint, $S^c v = 0$, is nonstochastic precisely because neither the parameters nor variables are stochastic.

4.1.3 Deterministic formulations

As we demonstrated in Section 2.1.1, stochastic linear inequalities lead either to second-order cone constraints or deterministic linear inequalities. We now recast the probabilistic RAMPER constraints as deterministic constraints and finally formulate RAMPER as an SOCP.

The rows of S^u are modeled to have normal random variable coefficients. As such, we have $S_i^u v$ normal with mean $\bar{S}_i^u v$ and corresponding uncertainty matrix R_i^s. The equivalent SOCP constraints are then $\bar{S}_i^u v + \|\delta_{1-\epsilon} R_i^s v\| \leq M_i$ and $-\bar{S}_i^u v + \|\delta_{1-\epsilon} R_i^s v\| \leq M_i$ for every row $i \in \mathcal{S}$. As for the kinetic constraints, we similarly model K_i to be normally distributed, and hence $K_i v$ is normal with mean $\bar{K}_i v$ and uncertainty matrix R_i^k. The

resulting SOCP constraints are then $\bar{K}_i v + \|\delta_{1-\epsilon} R_i^k v\| \leq e_i$ for all rows $i \in \mathcal{E}$. As for the enzyme pool constraint, we have already shown in Section 2.1.1 that stochastic right-hand coefficients simply produce stricter linear inequalities. It is therefore not of any use to recast the enzyme pool constraint stochastically but rather, as the stochastic interpretation suggests, set the enzyme pool bound one or two standard deviations below the mean. In a similar vein, we may recast probabilistic variable bound constraints but doing so would merely lead to stricter bounds and again not truly change the method formulation. The enzyme pool constraint and variable bounds will therefore remain unchanged from the ecFBA formulation in the following deterministic RAMPER description. For simplicity's sake, we continue using the same notation for the unaltered ecFBA constraints.

The SOCP formulation of RAMPER is given as

$$\max_v \, c^T v \tag{4.2a}$$
$$\text{s.t. } S^c v = 0, \tag{4.2b}$$
$$S_i^u v + \|\delta_{1-\epsilon} R_i^s v\| \leq M_i, \, \forall i \in \mathcal{S}, \tag{4.2c}$$
$$-S_i^u v + \|\delta_{1-\epsilon} R_i^s v\| \leq M_i, \, \forall i \in \mathcal{S}, \tag{4.2d}$$
$$K_i v + \|\delta_{1-\epsilon} R_i^k v\| \leq e_i, \, \forall i \in \mathcal{E}, \tag{4.2e}$$
$$w^T e \leq W, \tag{4.2f}$$
$$L \leq v \leq U, \tag{4.2g}$$
$$0 \leq e \leq E. \tag{4.2h}$$

As was discussed in Section 2.1.1, we may also reformulate the RAMPER SOCP using uncertainty sets in the following way

$$\max_v \, c^T v \tag{4.3a}$$
$$\text{s.t. } S^c v = 0, \tag{4.3b}$$
$$-M_i \leq S_i^u v \leq M_i, \, \forall S_i^u \in \mathcal{U}_i, \, \forall i \in \mathcal{S}, \tag{4.3c}$$
$$K_i v + \leq e_i, \, \forall K_i \in \mathcal{V}_i, \, \forall i \in \mathcal{E}, \tag{4.3d}$$
$$w^T e \leq W, \tag{4.3e}$$
$$L \leq v \leq U, \tag{4.3f}$$
$$0 \leq e \leq E, \tag{4.3g}$$

where $\mathcal{U}_i \coloneqq \{\bar{S}_i^u + \delta_{1-\epsilon} u^T R_i^u : \|u\| \leq 1\}$ and $\mathcal{V}_i \coloneqq \{\bar{K}_i + \delta_{1-\epsilon} u^T R_i^k : \|u\| \leq 1\}$.

4.1.4 Interpreting solutions and parameters

The stochastic interpretation of kinetic RAMPER constraints states that we may only consider metabolic fluxes, given an ET matrix sample, predicting enzyme concentration distributions in accordance with the enzyme pool bound. Another yet equivalent interpretation is the following. Consider a culture of homogeneous cells in which each cell has its own distinct array of effective TNs. RAMPER then only allows flux vectors such that, if imposed on all the cells, sample means of the corresponding enzyme concentrations have a

probability $1 - \epsilon$ of being below the enzyme pool bound. This is indeed equivalent since the flux vector multiplied by the rET matrix sample is precisely a sample of enzyme concentrations. Hence each RAMPER flux, together with the rET matrix, provides sample mean distributions of enzyme concentrations. We may thus bypass the flux vector and rather state the following: RAMPER only permits the cell culture to possess an enzyme concentration distribution among its cells so that a sample of the enzyme content is—with a high degree of probability—below the permitted total enzyme mass.

Correspondingly, the stochastic SS constraints will have similar stochastic interpretations. RAMPER will only permit fluxes that, if imposed on all cells in the culture, predict metabolite concentration change distributions across the culture that when sampled has a high probability of being close to zero.

To summarize, every flux vector corresponds to distributions of enzyme concentrations and metabolite concentration change across the cell culture. The RAMPER method constrains fluxes to distributions that satisfy desired properties relating to the corresponding sample mean distributions. Specifically, RAMPER constrains the cell culture to metabolic states under which sample means of enzyme concentrations meet total mass requirements and sample means of metabolite concentration change are close to zero. Finally, RAMPER identifies the flux vector maximizing the mean of the stochastic biomass production. Note however that the populational predictions stem from the stochasticity of modeling parameters and not the variables. Hence, we essentially assume all cells of the culture to adhere to the same flux vector and that intercellular variance arises from differences in effective TNs and growth requirements.

It is obviously unwarranted to assume equivalent flux vectors across cell cultures. The assumption is in fact conceptually contradictory to the RAMPER formalism. In contrast however, MacGillivray *et al.* [45] argue that the FBA approach implicitly enforces the same assumption. As such, the RAMPER method is not worse off than the FBA approach. Nevertheless, considering distributions of flux vectors themselves seems beyond the capability of optimization program applications. We may wonder however if the RAMPER solution is aptly interpreted as the sample mean of the metabolic fluxes. To answer this question, we would have to deal with products of random variables of dependent normal distributions and is hence outside the scope of our discussion.

We now turn to the question of whether said interpretations of stochastic RAMPER kinetic constraints are amenable to available experimental data. The TNs provided by enzyme databases such as BRENDA are typically determined *in vitro* and intend to capture the maximum enzyme turnover rate. Hence, these data do not necessarily agree with our definition of effective TNs. But sets of TN calculations from various experiments may provide viable approximations to variances of effective TN sample means. This is argued by the fact that such different experiments provide a glimpse into the variation of maximum conversion rates, and this variance is hopefully similar for effective conversion rates. In contrast, we would expect the expected average sample mean of experimentally inferred TNs to overestimate effective TNs—though underestimation is also possible in the case of heavily regulated enzymes. Fortunately however, Davidi *et al.* [90] reported a decent correlation between traditional and effective TN calculations when comparing the effective and traditional TN values of 132 metabolic reactions in *E. coli*. Experimental TN calculations may thus be helpful in the modeling of effective TN uncertainty after all.

4.2 Mathematical properties

We will now show that Theorem 1 and 2 extends from RAMP to RAMPER. These extended theorems will effectively show that solutions to RAMPER are Lipschitz continuous with regards to change in uncertain parameters and that solutions to ecFBA problems (2.19) are limiting cases of RAMPER problem (4.2) solutions.

4.2.1 Solution continuity in parametric uncertainty

RAMPER provides continuous flux states with respect to parameter data variation. Such continuity is congruent to biological reality, as minor parametric perturbations are abundant yet withstood by cellular metabolism. The result is stated mathematically by Theorem 4; the proof of which is provided below.

Theorem 4. *Let*

$$\mathcal{R} = \left\{R_i^S : i \text{ indexes a row of } S\right\} \cup \left\{R_i^K : i \text{ indexes a row of } K\right\},$$

and set $\mathbf{V}(S, K, \mathcal{R}, L, U)$ *to be the collection of feasible flux vectors for the RAMPER problem formulation in* (4.2) *with parameters* $(S, K, \mathcal{R})$ *and flux bounds* (L, U)*. Then for any* $v \in \mathbf{V}(S, K, \mathcal{R}, L, U)$*, there is a* $\lambda > 0$ *so that*

$$\begin{aligned}\lambda\,\Gamma \geq \min\{\|v - v'\| :\\ v' \in \mathbf{V}((S^c, S^u + \Delta S^u), K + \Delta K, \mathcal{R} + \Delta\mathcal{R}, L - \lambda\Gamma\mathbb{1}, U + \lambda\Gamma\mathbb{1})\},\end{aligned}$$

where

1. $\mathcal{R} + \Delta\mathcal{R} = \left\{R_i^S + \Delta R_i^S : i \text{ indexes a row of } S\right\}$
   $\cup \left\{R_i^K + \Delta R_i^K : i \text{ indexes a row of } K\right\}$*;*
2. λ *is independent of* $\|\Delta S_i^u\|$, $\|\Delta K_i\|$, $\left\|\Delta R_i^S\right\|$, *and* $\left\|\Delta R_i^K\right\|$*;*
3. $\Gamma = \max\left\{\max_i\left\{\|\Delta S_i^u\| + \|\Delta R_i^u\|\right\},\ \max_i\left\{\|\Delta K_i\| + \left\|\Delta R_i^K\right\|\right\},\right.$
   $\left.\|w^T\Delta K\| + \sum_i \left\|w_i\,\Delta R_i^K\right\|\right\}$*; and*
4. $\mathbb{1}$ *is a vector of ones.*

The independence of λ from $\|\Delta S_i^u\|$, $\|\Delta K_i\|$, $\|\Delta R_i^S\|$, and $\|\Delta R_i^K\|$ ensure that RAMPER solutions are Lipschitz continuous with respect to model parameters. This is a strong sense of continuity that can often replace bounded differentiability in theoretical arguments, and it guarantees that flux deviations are bounded linearly in the magnitude of parameter perturbations; the theorem asserts that optimal flux deviations grow at most linearly as catalytic rates, as well as their models of uncertainty, are perturbed. We may extend Theorem 4 to include enzyme concentrations by simply setting

$$e_i = \max\left\{0,\ K_i v + \left\|R_i^K v\right\|\right\}.$$

Then e_i inherits the continuity of v, as maxima of Lipschitz functions are themselves Lipschitz continuous.

Proof of Theorem 4

We must first establish extensions of Lemma 1 and Corollary 1 as presented for RAMP by MacGillivray *et al.* [45] before proceeding to prove Theorem 4 itself.

Lemma 1. *Let A be an n-element row vector, b be a strictly positive scalar, and $\mathcal{R}$ be a collection of $q_j \times n$ matrices, each denoted by R_j, where $j = 1 \dots p$. Let*

$$\mathcal{F}(A, \mathcal{R}) = \left\{ v : Av + \sum_{j=1}^{p} \|R_j v\| \leq b \right\}.$$

Then for any $v \in \mathcal{F}(A, \mathcal{R})$ there are $\lambda \geq 0$ and $0 < z \leq 1$ such that

$$\min \{\|v - v'\| : v' \in \mathcal{F}(A + \Delta A, \mathcal{R} + \Delta\mathcal{R})\}$$
$$\leq \|v - zv\| \leq \lambda \left(\|\Delta A\| + \sum_{j=1}^{p} \|\Delta R_j\| \right),$$

where

- $\mathcal{R} + \Delta\mathcal{R} = \{R_j + \Delta R_j : j = 1 \dots p\}$,
- $z\,v \in \mathcal{F}(A + \Delta A, \mathcal{R} + \Delta\mathcal{R})$, *and*
- λ *is independent of* ΔA *and* $\Delta\mathcal{R} = \{\Delta R_j : j = 1, \dots p\}$.

Proof. Let $v \in \mathcal{F}(A, \mathcal{R})$. The lemma holds with $z = 1$ and $\lambda = 0$ if $v \in \mathcal{F}(A + \Delta A, \mathcal{R} + \Delta\mathcal{R})$, so we assume to the contrary that $v \notin \mathcal{F}(A + \Delta A, \mathcal{R} + \Delta\mathcal{R})$. We then have $(A + \Delta A)v + \sum_{j=1}^{p} \|(R_j + \Delta R_j)v\| > b$. And by applying the Intermediate Value Theorem, there is a $\gamma \in \mathbb{R}$ satisfying $0 < \gamma \leq 1$ such that we also have

$$0 < (A + (1 - \gamma)\Delta A)v + \sum_{j=1}^{p} \|(R_j + (1 - \gamma)\Delta R_j)v\| = b.$$

Now let $A' = A + (1 - \gamma)\Delta A$, $R'_j = R_j + (1 - \gamma)\Delta R_j$ for $j = 1 \dots p$, and set

$$z = \frac{b}{b + \gamma(\|\Delta A\| + \sum_{j=1}^{p} \|\Delta R_j\|)\|v\|}.$$

We then have the following,

$$\begin{aligned}
&(A+\Delta A)(zv)+\sum_{j=1}^{p}\|(R_j+\Delta R_j)(zv)\|-b\\
&= (A'+\gamma\Delta A)(zv)+\sum_{j=1}^{p}\|({R_j}'+\gamma\Delta R_j)(zv)\|-b\\
&\leq z\left(A'v+\gamma\Delta Av+\sum_{j=1}^{p}\left(\|{R_j}'v\|+\gamma\|\Delta R_jv\|\right)\right)-b\\
&\leq z\left(A'v+\gamma\|\Delta A\|\|v\|+\sum_{j=1}^{p}\left(\|{R_j}'v\|+\gamma\|\Delta R_j\|\|v\|\right)\right)-b\\
&= z\left(A'v+\sum_{j=1}^{p}\|{R_j}'v\|+\gamma\left(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|\right)\|v\|\right)-b\\
&= z\left(b+\gamma\left(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|\right)\|v\|\right)-b\\
&= 0.
\end{aligned}$$

Hence $z\,v\in\mathcal{F}(A+\Delta A,\mathcal{R}+\Delta\mathcal{R})$ and

$$\min\{\|v-v'\|:v'\in\mathcal{F}(A+\Delta A,\mathcal{R}+\Delta\mathcal{R})\}\leq\|v-zv\|.$$

By setting $\lambda=\|v\|^2/b$, we have that

$$\begin{aligned}
\|v-zv\| &= \left(1-\frac{b}{b+\gamma(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|)\|v\|}\right)\|v\|\\
&= \frac{\gamma\|v\|^2}{b+\gamma(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|)\|v\|}\left(\|\Delta A+\sum_{j=1}^{p}\|\Delta R_jv\|\right)\\
&\leq \lambda\left(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|\right),
\end{aligned}$$

and hence

$$\min\{\|v-v'\|:v'\in\mathcal{F}(A+\Delta A,\mathcal{R}+\Delta\mathcal{R})\}\leq\|v-zv\|\leq\lambda\left(\|\Delta A\|+\sum_{j=1}^{p}\|\Delta R_j\|\right),$$

where λ is independent from ΔA and $\Delta\mathcal{R}$. □

Lemma 1 can be extended to finite collections of constraints per the following corollary.

Corollary 1. *For $i \in \{1 \dots m\}$, let A_i be an n-element row vector, let b_i be a strictly positive scalar, and let $\mathcal{R}_i$ be a finite collection of $q_{ij} \times n$ matrices each denoted by R_{ij}, where $j = 1 \dots p_i$. Let*

$$\mathcal{F}\{(A_i, \mathcal{R}_i) : i = 1 \dots m\} = \left\{ v : A_i + \sum_{j=1}^{p_i} \|R_{ij}\, v\| \leq b_i, \text{ for } i = 1 \dots m \right\}.$$

Then, for any $v \in \mathcal{F}\{(A_i, \mathcal{R}_i) : i = 1 \dots m\}$, there are scalars z and λ satisfying $0 < z \leq 1$ and $\lambda \geq 0$ so that

$$\min\left\{\|v - v'\| : v' \in \mathcal{F}\{(A_i + \Delta A_i, \mathcal{R}_i + \Delta\mathcal{R}_i) : i = 1 \dots m\}\right\} \leq \|v - zv\| \leq \lambda \max_i \left\{ \|\Delta A_i\| + \sum_{j=1}^{p_i} \|\Delta R_{ij}\| \right\},$$

where

- $\mathcal{R}_i + \Delta\mathcal{R}_i \coloneqq \{R_{ij} + \Delta R_{ij} : j = 1 \dots p_i\}$,
- $z\, v \in \mathcal{F}\{(A_i + \Delta A_i, \mathcal{R}_i + \Delta\mathcal{R}_i) : i = 1 \dots m\}$, *and*
- λ *is independent of* ΔA_i *and* $\Delta\mathcal{R}_i = \{\Delta R_{ij} : j = 1 \dots p_i\}$ *for* $i = 1 \dots m$.

Proof. The proof follows by defining z_i and λ_i to be the scalars guaranteed by Lemma 1 for each constraint $i \in \{1 \dots m\}$ and setting $z = \min_i\{z_i\}$ and $\lambda = \max_i\{\lambda_i\}$. □

Having established Lemma 1 and its corollary, we now proceed to prove the RAMPER continuity theorem.

Proof of Theorem 4. Let $\mathbf{V}(S, K, \mathcal{R}, L, U)$ be the collection of feasible flux states for the RAMPER model, meaning that there is a vector e for each $v \in \mathbf{V}(S, K, \mathcal{R}, L, U)$ so that

$$\begin{gathered} S^c\, v = 0,\ L \leq v \leq U, \\ -M_k + \left\|R_i^S v\right\| \leq S_i\, v \leq M_k - \left\|R_i^S v\right\|, \\ K_i v + \left\|R_i^K v\right\| \leq e_i, \\ 0 \leq e \leq E,\ w^T e \leq W. \end{gathered}$$

We show that $\mathbf{V}(S, K, \mathcal{R}, L, U) = \mathbf{V}'(S, K, \mathcal{R}, L, U)$, where

$$v \in \mathbf{V}'(S, K, \mathcal{R}, L, U)$$

satisfies

$$\begin{gathered} S^c\, v = 0,\ L \le v \le U, \\ -M_k + \left\|R_i^S v\right\| \le S_i\, v \le M_k - \left\|R_i^S v\right\|, \\ K_i v + \left\|R_i^K v\right\| \le E_i \\ w^T K_i v + \sum_i \left\|R_i^K v\right\| \le W. \end{gathered}$$

The fact that $\mathbf{V}(S, K, \mathcal{R}, L, U) \subseteq \mathbf{V}'(S, K, \mathcal{R}, L, U)$ holds trivially, and the reverse containment of $\mathbf{V}'(S, K, \mathcal{R}, L, U) \subseteq \mathbf{V}(S, K, \mathcal{R}, L, U)$ follows by defining e_i, for each $v \in \mathbf{V}'(S, K, \mathcal{R}, L, U)$, to be

$$e_i = \max\left\{0,\ K_i v + \left\|R_i^K v\right\|\right\}.$$

The SOCP constraints defining $\mathbf{V}'(S, K, \mathcal{R}, L, U)$ are of the form required by Lemma 1 under the tacit assumption that M_i, E_i, and W are strictly positive. We may thus apply Corollary 1 to their shared collection and let z' and λ' be the scalars from Corollary 1. Then

$$v \in \mathbf{V}(S, K, \mathcal{R}, L, U) \Rightarrow S^c\,(z'v) = z'\,(S^c v) = 0,$$

and

$$v \in \mathbf{V}(S, K, \mathcal{R}, L, U) \Rightarrow L - \lambda'\Gamma\mathbb{1} \le z\, v \le U + \lambda'\Gamma\mathbb{1},$$

with

$$\begin{aligned} \Gamma \quad = \quad & \max\left\{\max_i\left\{\|\Delta S_i^u\| + \|\Delta R_i^u\|\right\},\ \max_i\left\{\|\Delta K_i\| + \left\|\Delta R_i^k\right\|\right\},\right. \\ & \left.\left\|w^T \Delta K\right\| + \sum\nolimits_i \left\|w_i \Delta R_i^k\right\|\right\}. \end{aligned}$$

We conclude that

$$z'v \in \mathbf{V}((S^c, S^u + \Delta S^u),\ K + \Delta K, \mathcal{R} + \Delta\mathcal{R}),\ L - \lambda'\Gamma\mathbb{1},\ U - \lambda'\Gamma\mathbb{1}),$$

and hence,

$$\begin{aligned} \lambda'\,\Gamma \ge \|v - zv'\| \ge \\ \min\{\|v - v'\| : v' \in \mathbf{V}((S^c, S^u + \Delta S^u), K + \Delta K, \mathcal{R} + \Delta\mathcal{R}), \\ L - \lambda'\Gamma\mathbb{1},\ U - \lambda'\Gamma\mathbb{1})\}. \end{aligned}$$

□

4.2.2 Convergence under diminishing uncertainty

Theorem 5 shows that limiting trajectories of RAMPER solutions terminate at solutions of corresponding ecFBA problems (2.19) as parameter uncertainty disappear. A proof of the theorem is provided below.

Theorem 5. *Let $(S^t, K^t, \mathcal{R}^t, M^t)$ be a sequence of RAMPER modeling parameters so that*

$$(S^t, K^t, \mathcal{R}^t, M^T) \to (S, K, \mathcal{Z}, 0) \quad \text{as } t \to \infty,$$

where $\mathcal{R}$ has the same definition as Theorem 4, $\mathcal{Z} = \mathcal{R}$ with every R_i^S and R_i^K being zero, and M is a vector whose i-th component is M_i. Then, any convergent primal and dual sequence of RAMPER solutions has a limit that solves the linear ecFBA program (2.19).

Proof. The RAMPER SOCP formulation is an instance of the general form convex optimization problem,

$$\max \left\{ c^T x : A_i x + \|R_i\, x\| \le b_i, \text{ for all } i \right\}, \tag{4.4}$$

where

$$\begin{aligned} A^T &= \begin{bmatrix} (S^c)^T & -(S^c)^T & (S^u)^T & -(S^u)^T & I & -I & K & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -I & I & -I & w \end{bmatrix}^T, \\ b^T &= \left(0,\, 0,\, M^T,\, -M^T,\, U,\, L,\, 0,\, E,\, 0,\, T_\varepsilon\right), \text{ and} \end{aligned}$$

the R_i matrices for the non-red constraints are zero.

The necessary and sufficient conditions of optimality are

$$\begin{aligned} A_i x + \|R_i x\| &\le b_i, \ \forall\, i \\ c_j + \sum_{i=1}^{m} \lambda_i\, v_j^T \left(w + A_i^T\right) &= 0, \ \forall\, j \\ \sum_{i=1}^{m} \lambda_i \left(b_i - \|R_i x\| - A_i x\right) &= 0, \ \text{and} \\ \lambda &\ge 0, \end{aligned}$$

where v_j is a vector of zeros except for a 1-element at position j and

$$w = \begin{cases} R_i^T \left(\dfrac{R_i x}{\|R_i x\|} \right), & R_i x \neq 0 \\ 0, & R_i x = 0. \end{cases}$$

Suppose that (x^t, λ^t, ρ^t) is a convergent sequence that satisfies these conditions for the parameter collection $(A^t, \mathcal{R}^t, b^t)$, where the definition of $\mathcal{R}$ is analogous to Theorem 4 and all R_i matrices originally zero remain zero for all t. The convergence assumptions

of the modeling parameters in Theorem 5 imply that we have, for all t where $R_i^t x^t$ not identically zero, both

$$0 \leq \|R_i^t\, x^t\| \leq \|R_i^t\|\, \|x^t\| \to 0 \text{ as } t \to \infty$$

and

$$\left\| R_i^t \left(\frac{R_i^t\, x^t}{\|R_i^t\, x^t\|} \right) \right\| \leq \|R_i^t\| \left\| \left(\frac{R_i^t\, x^t}{\|R_i^t\, x^t\|} \right) \right\| = \|R_i^t\| \to 0 \text{ as } t \to \infty.$$

The convergence assumptions of Theorem 5 also ensure that

$$\begin{aligned} &(x^t, \lambda^t) \to (\hat{x}, \hat{\lambda}), \quad A^t \to \hat{A}, \quad \text{and} \\ &b^t \to \hat{b} = (0,\, 0,\, 0,\, 0,\, U,\, L,\, 0,\, E,\, 0,\, T_\varepsilon) \quad \text{as} \quad t \to \infty. \end{aligned}$$

Hence, we conclude that

$$\begin{aligned} \hat{A}\hat{x} &\leq \hat{b}, \\ c^T + \hat{\lambda}^T \hat{A} &= 0, \\ \hat{\lambda}^T \left(\hat{A}\, \hat{x} - \hat{b} \right) &= 0, \quad \text{and} \\ \hat{\lambda} &\geq 0. \end{aligned}$$

These necessary and sufficient conditions show that $\hat{x}$ solves the linear program

$$\max\{c^T x : \ \hat{A}x \leq \hat{b}\},$$

which equates the ecFBA problem formulation (2.19). □

4.3 Computational speed

The computation times of RAMPER and ecFBA problems as executed by the RAMPER implementation (see Section 3.3.1) were calculated and compared. The RAMPER implementation consists of two primary functions named *createUncModel()* and *solveUncModel()*. The former inputs metabolic reconstructions and parses modeling data, whereas the latter inputs the parsed reconstruction data, constructs Pyomo model instances (described in Section 3.2.1), and solves the resulting optimization problems utilizing the Gurobi solver (described in Section 3.2.2). We computed the execution times of the two primary functions, the Pyomo model instance construction, and the Gurobi solver. The results are summarized in Table 4.1 and show that the model data parsing is by far the most time-expensive portion of the RAMPER implementation. The time difference seen in *solveUncModel()* is solely due to the Gurobi solver, as the instantiating of Pyomo instances is equally time-costly in RAMPER and ecFBA. We conclude that RAMPER problems do indeed take longer to set up and solve than their ecFBA counterparts, though the difference should be bearable when solving moderate numbers of problems.

Table 4.1: Computation times for the creation and solving of RAMPER and ecFBA problems employing the RAMPER implementation. All results are presented in seconds and include execution times for the primary implementation functions, the Pyomo instance construction, and the Gurobi solver.

Method	*createUncModel()*	*solveUncModel()*	Pyomo instance	Gurobi solver
RAMPER	6.52	3.81	1.40	1.92
ecFBA	1.15	2.12	1.40	0.72

4.4 Designing probability distributions of enzyme kinetic parameters

Whereas a lack of experimental data on biomass composition parameters hinders discussions of their distributional characteristics, this is not the case for TNs. In this section, we probe an exhaustive enzyme database and subsequently identify a near-linear relationship between TN mean and standard deviation. Conclusively, we discuss the application of the near-linear relation in the construction of kinetic uncertainty matrices.

4.4.1 Uncertainty in turnover numbers

Inspecting the BRENDA repository [95], we examined relationships between TN means and standard deviations. After removing TN entries found in aberrant experimental conditions and ignoring enzyme-substrate pairs related to fewer than two TN measurements, we sorted the TN values according to their corresponding enzyme-substrate pairs. As the TNs themselves are inversed in their respective kinetic constraints, we subsequently calculated the mean and standard deviation of the inversed TN measures per enzyme-substrate pair. We finally analyzed the results by linear regression as depicted in Figure 4.1.

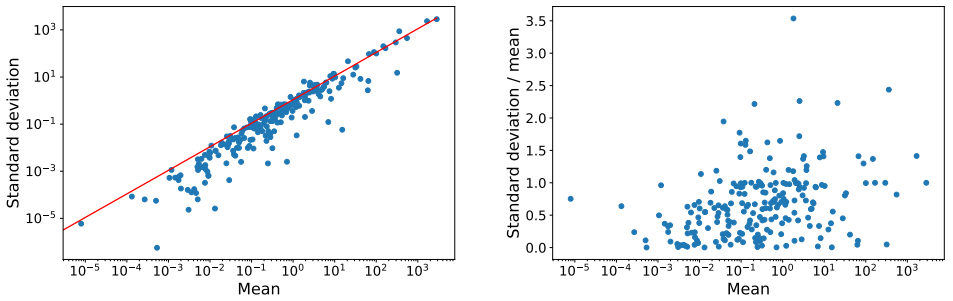


Figure 4.1: *Left:* Log-log plot of inversed TN means against standard deviations of enzyme-substrate pairs per the BRENDA database [95] together with a linear regression line. *Right:* Semi-log plot of the ratio of mean to standard deviation against standard deviation for the same measurements.

Linear regression on the logarithmic mean against logarithmic standard deviation indicates an almost linear relationship between mean and standard deviation (R-value $\simeq 0.94$, slope $a \simeq 1.1$, and intercept $b \simeq -0.73$). We may use the preceding regression analysis to postulate an effective model for TN distribution statistics for the RAMPER method. The resulting approximate relation between TN sample mean and standard deviation is

$$\sigma \approx e^b \mu^a, \tag{4.5}$$

where μ is the mean and σ is the standard deviation. In contrast, the logarithmic ratio of mean to standard deviation against logarithmic standard deviation does not appear to be as closely related (R-value $\simeq 0.32$, not shown) with similar conclusions drawn for semi-log and regular linear regression as well.

4.4.2 Composing uncertainty matrices for kinetic parameters

Having assumed normality in the sample mean of effective TNs with corresponding means and standard deviations coupled by the equation (4.5), we may now proceed to construct kinetic uncertainty matrices. The covariance matrix Q is related to the uncertainty matrix by $R^T R = Q$. We note that, since the formulation of the RAMPER SOCP (4.2) includes the matrix-vector product $R^k v$, the uncertainty matrices must have column dimensions of the size of the flux vector, whereas the size of the row dimension is still undecided.

In our subsequent analysis, we will assume the TN sample means to be uncorrelated. The assumption allows a simple, intuitive composition of the uncertainty matrix. We let R_i^k contain as many rows as there are uncertain parameters in K_i. Each row will then correspond to an uncertain parameter, and we let the first indexed parameter correspond to the first row, the second indexed uncertain parameter to the second row, and so on. Each row is constructed as to contain zeros except in all entries except for the column index corresponding to the row's uncertain parameter index, which will contain its standard deviation. It is straightforward to verify that said uncertainty matrix definition results in $R^T R$ becoming the covariance matrix of uncorrelated random variables.

We may additionally motivate our construction of uncertainty matrices by the mathematical discussion in Appendix A.1. We see that our construction is indeed the uncertainty matrix, of the kind described in Appendix A.1, with the fewest dimensions and the least nonzero entries. Hence, we are essentially constructing the simplest uncertainty matrix possible using linearly independent column vectors.

In order to ease our further discussion, we will set ϵ so that $\delta_{1-\epsilon} = 1$. In other words, we set $1 - \epsilon \approx 0.841$, as we may then implement our composed uncertainty matrices directly into the RAMPER SOCP formulation. In the following computational analyses, the kinetic uncertainty matrices are constructed using the kinetic coefficients found in the ecYeast8 model (see Section 3.1.2). We set these kinetic coefficients as the expected average of the TN sample means and subsequently determine the corresponding standard deviations by applying the regression equation (4.5). Then, we construct the uncertainty matrices using these standard deviations whilst assuming all stochastic kinetic coefficients to be uncorrelated. We will refer to the construction of said kinetic uncertainty matrices as the TN *regression model*.

4.5 Analyzing the impact of stochasticity in enzyme kinetic parameters

Probing the significance of increasing uncertainty in kinetic data, we may be able to attribute a degree of importance to individual enzymes. The key idea being that metabolically essential enzymes will more abruptly affect RAMPER growth rate predictions should they be assigned greater uncertainty. In deterministic settings, we may similarly determine the extent of enzymes' importance by incrementally decreasing TNs. Likewise, we expect metabolically crucial enzymes to impose greater changes on ecFBA predicted growth rates under diminishing TN values. In this section, we assess whether or not the RAMPER and ecFBA methods yield similar predictions of said enzyme importance. In other words, we are interested in whether or not the consideration of enzyme kinetic uncertainty in fact offers novel modeling premises. At first, we compare ecFBA and RAMPER simulations in which we introduce stochasticity in individual enzymes one at a time. Afterward, we apply the TN regression model to the RAMPER method—effectively introducing stochasticity in all enzymes concurrently—and similarly compare the simulation results with those of previous experiments.

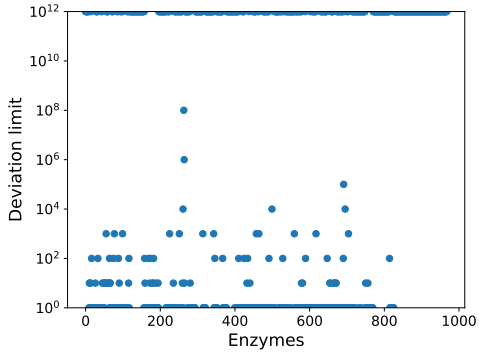


Figure 4.2: The deviation limit of enzymes in ecYeast8 using the RAMPER method. The vertical axis is scaled logarithmically in base 10. The iterations of standard deviation values started at 10 and ended in 10^{12} increasing by a factor 10 at each step. Note that enzymes which we could only determine to have deviation limits below 10 are found at position 0 with regards to the deviation limit axis.

Using the ecYeast8 (see Section 3.1.2) imposed under oxygen- and glucose-rich environments, we gradually increased the magnitude of uncertainty in individual enzyme's TNs. More precisely, we iterated through each enzyme and found the greatest term we could concurrently set as standard deviations in the enzyme's corresponding kinetic uncertainty matrix without predicting growth rates below half the reference growth rate—which we computed initially under the assumption of no uncertainty. Keep in mind that enzymes were modeled stochastically one at a time and that their TN coefficients were assumed uncorrelated. Fortunately, the stochastic RAMPER description sheds light on how to interpret said computational experiment: As we are increasing the standard deviations of an enzyme's uncorrelated TNs by the same factor, we are essentially increasing the stan-

dard deviation of the vector product $K_i v$ by that same factor. We are thus increasing the standard deviation of the enzyme's concentration uniformly across the entire flux solution space. Therefore, we will refer to the maximum standard deviation factor as the enzyme's *deviation limit*.

As mentioned, we would like to perform a similar computational experiment applying the ecFBA method. In doing so, we analogously iterate through each enzyme and correspondingly divide its TN values by a common factor. We decrease the TN coefficients by the same factor for a number of iterations stopping whenever the growth rate falls below half the reference growth rate—which we computed initially using the original TN coefficients. We will refer to the largest such division factor as the enzyme's *mean factor*.

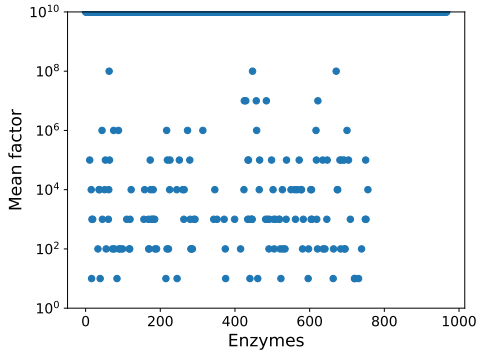


Figure 4.3: The mean factor of enzymes in ecYeast8 employing the ecFBA method. The vertical axis is scaled logarithmically in base 10. The iterations of mean factors started at 10 and ended in 10^{10} increasing by a factor 10 in each step.

We would like to stress the fact that the deviation limit and mean factor cannot be compared quantitatively. We may, on the other hand, compare the order in which the deviation limit and mean factor rank the enzymes' importance.

At each iteration of our experiments, we compute the growth rate after having imposed a change in either TN mean or standard deviation. The granularity of the analyses will therefore depend on the differences in these changes. Figures 4.2 and 4.3 illustrate large-scale, coarse experiments employing the RAMPER and ecFBA method respectively. The experiments demonstrate considerable spans in deviation limits and mean factors but suffer clearly from inadequate granularity. Nevertheless, the results may still prove insightful. Noticing there are 338 enzymes with deviation limits below 10, we do not know the order of these enzymes among themselves. Yet we do know they are the 338 smallest ones. As for the mean factor, we see that there are 13 enzymes with mean factors below 10^3. We do not know the order of these either, but we do know they are the 13 enzymes with the smallest mean factors.

Consider now the set of 338 most important enzymes as determined by their deviation limit and likewise the set of 13 most important enzymes determined by their mean factor. We identified the set of 13 enzymes not to be a subset of the set of 338 enzymes. More precisely, exactly 2 of the 13 enzymes with the lowest mean factors were not even among the 338 enzymes with the lowest deviation limits. That is, relatively among the enzymes,

increasing standard deviation is not equivalent to decreasing the mean. It is implied from this that RAMPER, even without presumed uncertainties in kinetic coefficients, provides a formalism that may investigate aspects of enzyme characteristics not possible with ecFBA. Furthermore, we have shown that RAMPER predicts certain enzymes to be more sensitive to increases in standard deviation than reductions in mean, at least in comparison to other enzymes.

We also repeated the analysis for RAMPER mounted with the TN regression model (introduced in Section 4.4.1). In this computational experiment, the deviation limit was found by multiplying the already-set standard deviations of the given enzyme's uncertainty matrix by a set factor for a number of iterations. At each iteration the growth rate was computed and compared to a reference growth rate, which was computed initially applying the TN regression model. Applying the TN regression model amounted to a 15% decrease in reference growth rate.

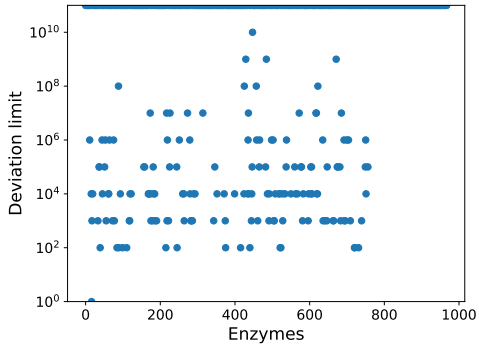


Figure 4.4: The maximum deviation of enzymes in ecYeast8 applying the RAMPER method with the TN regression model. The vertical axis is scaled logarithmically in base 10. The iteration of deviation scaling factors started at 100 and ended in 10^{11} increasing by a factor of 10 at each step. Note that enzymes which we could only determine to have deviation limits below 10 are found at position 0 with regards to the deviation limit axis.

We see that the results, illustrated in Figure 4.4, appear to reflect, to some degree, the results of the ecFBA experiment. Nonetheless, an analysis of enzyme importance rankings again yields differences from the ecFBA method. The 13 smallest mean factor enzymes found in ecFBA are not a subset of the 16 smallest deviation limits found using RAMPER with the TN regression model. Indeed, 3 enzymes found using ecFBA are not part of the 16 enzymes identified using RAMPER with the TN regression model. Neither do these 3 enzymes overlap with the 2 aberrant enzymes found comparing ecFBA and RAMPER lacking the TN regression model. Hence, we see again that some enzymes, at least in comparison to others, are not affected equally by changes in deviation and mean. Imposing uncertainty across all enzymes or not, the incorporation of uncertainty in enzyme kinetic parameters provide RAMPER with distinct enzyme characteristic predictions not possible using its deterministic counterpart.

Chapter 5

Discussion

Having introduced the RAMPER method, we now turn to discuss its potential strengths and weaknesses. We contemplate the validity of the deterministic RAMPER description, discuss parameter-related modeling issues, and describe alternative method descriptions accounting for correlated enzymatic stochasticity. Moreover, we discuss RAMPER's compatibility with the ecYeast8 reconstruction and discuss recent trends in the incorporation of enzyme kinetic parameters in metabolic modeling. We furthermore consider the inferior computational speed and stability observed in RAMPER, before finally contemplating conceivable applications of our method.

Although the stochastic RAMPER formulation is arguably well-warranted, the deterministic formulation may appear less so. At first glance, it seems that we simply have formulated a stricter FBA problem in which each constraint is replaced by a set of constraints. One could argue that a cell must not satisfy all these constraints at once but rather that the cell must satisfy an instance of constraints drawn from the uncertainty set. Hence applying all these constraints at once is ill-advised, as it would exceedingly constrain the solution space. Though otherwise a well-reasoned argument, it incorrectly assumes that RAMPER attempts to recapture the metabolism of an individual cell. As is the case with the stochastic interpretation, it is only reasonable to interpret the RAMPER formalism in terms of cell cultures and not single cells. When introducing RAMP, MacGillivray *et al.* [45] argued that FBA likewise implicitly models cell cultures and not individual cells, effectively arguing that these approaches are more aptly interpreted as modeling cultures. In that sense, FBA is incorrectly assuming nonchanging biological factors across cell cultures, whereas RAMPER does not.

We could have nonetheless considered an alternative approach based on stochastic programming. Employing such classes of optimization programs, we could model individual cells maximizing biomass withstanding the uncertainty inherent to biological processes. Such an approach would be applicable for species that are evolutionarily adapted to changing conditions and for which the fluxes are optimized to maximize biomass anticipating stochasticity in growth-related parameters. Stochastic programming is, on the other hand, notoriously difficult to formulate and implement. Additionally, we consider RAMPER's

ability to model cultures to be valuable in and of itself. Employing stochastic programming could potentially lose the populational aspect of RAMPER.

According to our discussion on SOCPs in Section 2.1.1, probabilistic kinetic constraints containing but one TN coefficient will simply result in linear constraints accommodating the worst-case value of those TNs. This might feel wrong to some readers who may ask themselves how the stochastic RAMPER interpretation could end up describing the same ecFBA linear constraints. We remind those readers that the constraints possessing more than two uncertain kinetic parameters will not be reduced to linear constraints but rather to proper second-order cone constraints. Importantly, the solution space is contingent on the entire collection of constraints and not the individual constraints in isolation. Hence, it should be clear that the entirety of the RAMPER solution space is not equivalent to the correspondingly stricter ecFBA solution space. We will return to this discussion later when describing an alternative RAMPER description.

We have as of yet discussed the importance of modeling parameter ϵ. Modifying the ϵ parameter not only changes simulation outcomes but also their interpretation. Decreasing ϵ effectively tightens the probabilistic RAMPER constraints and hence narrows the solution space subsequently decreasing growth rate predictions. MacGillivray *et al.* [45] found however that the ϵ parameter had an insignificant impact on RAMP gene essentiality predictions. Exactly how much the ϵ modeling parameter affects RAMPER method predictions remains to be examined. It is also problematic to interpret changes in ϵ. Even more worrying is the fact that a uniform change in standard deviations across uncertainty matrices would impose equivalent effects on the RAMPER SOCP formulation as corresponding changes in ϵ. Although causing the same changes to the RAMPER solution space, modifications to ϵ and parameter standard deviations are interpreted quite distinctly.

As is mentioned in Section 2.1.1, it is not possible to model correlated random variable coefficients across separate constraints using SOCP formulations. We only expect correlations in uncertainty between TNs of the same enzyme, and as each row of the ET matrix corresponds to distinct enzymes, we do not expect this to be an issue. On the other hand, we do expect the literature of enzymatically constrained metabolic modeling approaches to grow, and it is not unforeseeable that future GEMs will include turnover matrices containing novel classes of "pseudo-enzymes" which may in fact necessitate stochastic correlations across constraints. In that case, new modeling approaches are needed. One such approach is briefly discussed next, which may also potentially circumvent the interpretive issue of linear kinetic constraints arising from singular TNs as discussed above.

The proof of Theorem 4 presents two equivalent flux solution spaces both described in terms of SOCP constraints. The differences are based solely on substituting e_i for $K_i v$. That same substitution applied to an ecFBA problem (2.19) will similarly result in a distinct model formulation describing an equivalent solution space. As opposed to the canonical ecFBA formulation, the problem formulation resulting from the variable substitution will have replaced both the enzyme pool constraint and the turnover matrix by a variant of the pool enzyme mass constraint, $w^T T v \leq W$. We could—were we to extrapolate a second-order cone constraint from the pool enzyme mass constraint variant—construct a single uncertainty matrix for the vector $w^T T$. One possibility, though unorthodox, would be to create an uncertainty matrix with rows corresponding to each uncertain parameter in T in which case we would be able to model correlations of any kind. Such a RAM-

PER variant would also avoid the issue of having kinetic constraints remodeled as worst-case linear constraints. We will refer to this alternative approach as *substituted* RAMPER (sRAMPER). Interesting questions arise when considering the sRAMPER formulation. Will sRAMPER and RAMPER be equivalent? If not, are there conditions we may impose on their uncertainty matrices so that they are? Should the two problems be equivalent under reasonable restrictions on the uncertainty matrices, we could rest assured that RAMPER can be reformulated to handle correlations across kinetic constraints. It would also provide an additional argument that the RAMPER solution space is not justly interpreted as a further constriction of the ecFBA solution space. In contrast, the more orthodox choice for the sRAMPER uncertainty matrix would contain rows for each element in $w^T K$ though this is biologically nonsensical. Doing so, we would only be able to model correlations between all TNs related to one reaction to all TNs related to another.

Moving over to the construction of uncertainty matrices for the RAMPER method (see Section 4.4.2), we did not cover the issue of correlated random variable coefficients. This is a pressing issue, as certain biochemical reactions in metabolic reconstructions are split into similar reactions—catalyzed by the same enzyme—using different substrates. For some of these reactions, the substrates are expected to be converted with an equal turnover rate. Consider for example an enzyme catalyzing a reaction with long-chain lipid substrates for which only the head group is chemically active during catalysis. These lipid substrates may consist of the same head group yet differ in their chain structure. But as the chain structure is biochemically irrelevant to the catalyzing enzyme, we do not expect the corresponding TN values to differ either. In GEMs, such reactions are handled by creating separate reactions for each substrate but for which their TN is the same. These kinetic coefficients should therefore be correlated with covariance factors equal to one. In the computational implementation of RAMPER (described in Section 3.3.1), we handle correlated uncertain parameters by constructing uncertainty matrices in which the rows corresponding to the correlated TNs have the same elements in their corresponding nonzero column indices. In other words, the rows corresponding to correlated TNs are set equal to each other. It can be showed by straightforward matrix manipulations that such uncertainty matrices do indeed result in the desired covariance matrices. The ellipsoid describing the uncertainty set will then only contain points in which the magnitude of the dimensions of the correlated variable coefficients are equal. That is to say, uncertainty in one of the coefficients necessitates the same amount of uncertainty in its correlated coefficients. In the current implementation of RAMPER, correlated coefficients of other values are implemented similarly, although a more intuitive user interface for modeling correlations is certainly needed.

The ecYeast8 GEM used in our computational efforts includes TNs that may not be congruent to the RAMPER modeling formalism. The majority of the TNs are best-case turnover rates from BRENDA, whereas others are manually curated and fitted to experimental growth rates. Ideally, we would want to have TNs determined as the average over various experiments. Fortunately, such a pipeline exists—named AutoPACMEN [40]—-but was inconveniently made available after our computational efforts were concluded. The AutoPACMEN method produces ecFBA-accommodating GEMs using the averages over TN entries extracted from both BRENDA [95] and SABIO-RK [104]. The resulting enzymatically constrained GEM is not of the form introduced in Section 2.4.2 but has

rather substituted e_i for $T_i v$ analogously to sRAMPER. Extending RAMPER to GEMs produced by AutoPACMEN should be straightforward and is integral to the RAMPER implementation should it be made available to the constraint-based modeling community.

The TN regression model reveals incredibly large standard deviations within enzyme-substrate pairs even for narrow ranges of experimental conditions. As we intended a large-scale probing of TN measurements, we did not pursue to scrutinize the individual TN data entries. Strikingly, it should be noted that previous research [90] has reported inconsistent units in BRENDA TN entries, which might help explain the large variance observed in kinetic data. We would also like to point out that including the TN regression model amounted to a 15% reduction in predicted growth rate for the ecYeast8 reconstruction. Should we in fact have overestimated the variance in kinetic data, the predicted growth rate ought to suffer a smaller reduction. In either case, we note that decreases in growth rate predictions are easily ameliorated by increases in the enzyme pool bound.

As was discussed above, we would ideally use averages over TN determination experiments as coefficients in the ET matrix. Alternatively, we could use the largest TN entries—which are readily accessible in many ecFBA-accommodating GEMs—to predict sample standard deviations using the TN regression model and then subsequently set their differences as ET matrix coefficients. We could also potentially predict effective TNs employing experimental flux measurements and proteomic data or even tune TN standard deviations to match protein measurement variances. At any rate, the TN regression is merely presented as a tool for assessing the RAMPER method under global kinetic uncertainty and is by no means meant to be interpreted as a rigorous and viable option for modeling kinetic data statistics—at least not without further scrutiny.

There is no doubt that the RAMPER approach suffers from computational instability. As was experienced with RAMP, consensus *E. coli* models prove highly unstable when applying the RAMPER method. We cannot tell yet why yeast models—not only ecYeast8 but other yeast reconstructions as well—appear to be considerably more stable than their *E. coli* counterparts. It is tempting to claim that *E. coli* GEMs are more prone to over-optimization in modeling parameters than yeast models, though we struggle to find evidence supporting the supposition. We do not expect *E. coli* enzymatically constrained reconstructions produced by AutoPACMEN to be more stable than canonical *E. coli* models, as these GEMs are already very unstable.

RAMPER's computational speed is satisfactory for smaller applications. However, the small increment in execution time makes RAMPER far more time-expensive than ecFBA when performing large-scale computational experiments. In probing the sensitivity of enzyme TN uncertainty, we relied on parallel computing in order to execute the experiment within an acceptable timeframe. We expect that other large-scale computational experiments using RAMPER will have to rely on parallelization techniques as well. The construction of Pyomo model instances from metabolic reconstructions usually takes ~8 seconds and is far slower than the actual solving of the instances. Making small changes in the RAMPER problem formulation is therefore far more time-consuming than solving the problem itself, as per date these changes must be made to the COBRApy model object before reinitializing the Pyomo instance construction. It is therefore imperative that utilities to perform routine modeling modifications are implemented to accommodate gene-knockout simulations, environment condition changes, and other common modeling

adjustments. Such utilities should be easily implemented using existing Pyomo functions that facilitate alterations of model instance structures.

Despite its inferior stability and speed, the RAMPER method opens a revenue of computational analyses inaccessible by canonical constraint-based approaches. As RAMPER solutions implicitly solve for expected means and standard deviations of metabolite concentration change and enzyme concentration samples, the method effectively predicts populational cell culture properties. Such distributional predictions are far stricter than numerical value predictions, making RAMPER far easier to disprove than its deterministic counterparts. It is also foreseeable that data on enzyme concentration variance can impose further bounds on the RAMPER solution space, and that enzyme concentrations lacking such data could be inferred from RAMPER solutions. The predicted standard deviations could also find their way into the objective function, enabling modelers to probe potential variations on the biomass objective.

As uncertain RAMPER modeling parameters are assumed to follow normal sample mean distributions, the biomass objective function is more aptly interpreted as the maximization of the expected mean of biomass production. We reckon to be able to introduce the standard deviation of the biomass sample mean into the objective, effectively resulting in a quadratic objective function. The biological relevance and use of such an objective, however, is up to debate. At any rate, RAMPER solutions should, in theory, be able to predict variances in sample means of biomass production. Additionally, RAMPER could be used to probe the impact of strain-engineering on intercellular variance, hopefully enabling the optimization of populational cell properties for engineering purposes. The stochastic interpretation of RAMPER also sheds light on its possible application in the prediction and engineering of metabolite pooling. Although the viability of the RAMPER formalism remains to be rigorously established, the scope of its potential applicability and predictive power is nonetheless intriguing.

Chapter 6

Outlook

We introduced the RAMPER method to acknowledge the parametric uncertainty found in enzyme kinetic data utilized in the constraint-based analysis of metabolic models. We found that the RAMP framework, developed to alleviate the strict modeling assumptions of FBA, can be extended to include enzyme kinetic constraints similar to ecFBA. Importantly, we showed that RAMPER preserves the essential mathematical properties of RAMP. We may, therefore, assert that the RAMPER method indeed encapsulates ecFBA and exhibits continuity under parametric perturbations. Unfortunately, RAMPER suffers from an unstable and slow computational implementation. Nonetheless, we conclusively showed that RAMPER predicts distributional enzyme properties unattainable by previous methods.

Looking ahead, establishing the predictive accuracy of the RAMPER method is of foremost importance. Not only should we verify RAMPER's capability for correctly predicting growth rates under distinct environmental conditions but also its proficiency at describing phenotypic alterations due to protein allocation—such as the Crabtree effect observed in baker's yeast. We should also consider approaches to corroborate modeling predictions of distributional parameters with experimental measurements. Additionally, we should further examine the impact of stochastic modeling parameters. An understanding of the parameters' effect on model accuracy is vital are we to harness RAMPER's predictive power. The potential synergy between modeling parameters may, however, severely complicate sensitivity analyses. Optimizing the RAMPER implementation may prove necessary, as large-scale computational experiments rely on parallelization to finish in acceptable timeframes. Another pressing issue is the method's computational instability, especially considering the significant differences observed for distinct metabolic reconstructions. Ideally, we should also clarify the mathematical relations between RAMPER and sRAMPER, additionally incorporating AutoPACMEN-generated reconstructions to our computational implementation.

Finally, we should exercise RAMPER's capacity for modeling metabolite pooling and enzyme distribution statistics; we should explore the uncharted modeling premises made accessible by its application as a framework for constraint-based metabolic modeling.

Bibliography

1. Regenmortel, M. H. V. V. Reductionism and complexity in molecular biology. *EMBO Reports* **5,** 1016–1020. doi:10.1038/sj.embor.7400284 (2004).
2. Trewavas, A. A Brief History of Systems Biology. *The Plant Cell* **18,** 2420–2430. doi:10.1105/tpc.106.042267 (2006).
3. Von Bertalanffy, L. An Outline of General System Theory. *The British Journal for the Philosophy of Science* **I,** 134–165. doi:10.1093/bjps/I.2.134 (1950).
4. Fang, F. C. & Casadevall, A. Reductionistic and holistic science. *Infection and Immunity* **79,** 1401–1404. doi:10.1128/IAI.01343-10 (2011).
5. Kolch, W. *et al.* Systems biology primer: the basic methods and approaches. *Essays in Biochemistry* **62,** 487–500. doi:10.1042/EBC20180003 (2018).
6. Kitano, H. Systems Biology: A Brief Overview. *Science* **295,** 1662–1664. doi:10.1126/science.1069492 (2002).
7. Hartwell, L. H., Hopfield, J. J., Leibler, S. & Murray, A. W. From molecular to modular cell biology. *Nature* **402,** C47–C52. doi:10.1038/35011540 (1999).
8. Misra, B. B., Langefeld, C., Olivier, M. & Cox, L. A. Integrated omics: tools, advances and future approaches. *Journal of Molecular Endocrinology* **62,** R21–R45 (2019).
9. Ge, H., Walhout, A. J. M. & Vidal, M. Integrating 'omic' information: a bridge between genomics and systems biology. *Trends in Genetics* **19,** 551–560. doi:10.1016/j.tig.2003.08.009 (2003).
10. Facciotti, M. T. Training Interdisciplinary Scientists for Systems Biology. *Journal of Investigative Medicine* **57,** 471–473. doi:10.2310/JIM.0b013e31819825e3 (2009).
11. Oliveira, J. S. *et al.* A Computational Model for the Identification of Biochemical Pathways in the Krebs Cycle. *Journal of Computational Biology* **10,** 57–82. doi:10.1089/106652703763255679 (2003).
12. Purves, D. & Pacala, S. Predictive Models of Forest Dynamics. *Science* **320,** 1452–1453. doi:10.1126/science.1155359 (2008).

13. Bruggeman, F. J. & Westerhoff, H. V. The nature of systems biology. *Trends in Microbiology* **15,** 45–50. doi:10.1016/j.tim.2006.11.003 (2007).

14. Das, M., Patra, P. & Ghosh, A. Metabolic engineering for enhancing microbial biosynthesis of advanced biofuels. *Renewable and Sustainable Energy Reviews* **119,** 109562. doi:10.1016/j.rser.2019.109562 (2020).

15. Garcia, S. & Trinh, C. T. Modular design: Implementing proven engineering principles in biotechnology. *Biotechnology Advances* **37,** 107403. doi:10.1016/j.biotechadv.2019.06.002 (2019).

16. Biz, A. *et al.* Systems biology based metabolic engineering for non-natural chemicals. *Biotechnology Advances* **37,** 107379. doi:10.1016/j.biotechadv.2019.04.001 (2019).

17. Wang, E. Understanding genomic alterations in cancer genomes using an integrative network approach. *Cancer Letters* **340,** 261–269. doi:10.1016/j.canlet.2012.11.050 (2013).

18. Li, X.-T., Yang, J.-J., Wu, Y.-L. & Hou, J. Toward innovative combinational immunotherapy: A systems biology perspective. *Cancer Treatment Reviews* **68,** 1–8. doi:10.1016/j.ctrv.2018.05.003 (2018).

19. Masoudi-Nejad, A. & Asgari, Y. Metabolic Cancer Biology: Structural-based analysis of cancer as a metabolic disease, new sights and opportunities for disease treatment. *Seminars in Cancer Biology* **30,** 21–29. doi:10.1016/j.semcancer.2014.01.007 (2015).

20. Naimo, G. D. *et al.* A Systems Biology Approach for Personalized Medicine in Refractory Epilepsy. *International Journal of Molecular Sciences* **20,** 3717. doi:10.3390/ijms20153717 (2019).

21. Dovrolis, N. Systems biology in inflammatory bowel diseases: on the way to precision medicine. *Annals of Gastroenterology* **32,** 233–246. doi:10.20524/aog.2019.0373 (2019).

22. Mulder, S., Hamidi, H., Kretzler, M. & Ju, W. An integrative systems biology approach for precision medicine in diabetic kidney disease. *Diabetes, Obesity and Metabolism* **20,** 6–13. doi:10.1111/dom.13416 (2018).

23. Filipp, F. V. Precision medicine driven by cancer systems biology. *Cancer and Metastasis Reviews* **36,** 91–108. doi:10.1007/s10555-017-9662-4 (2017).

24. Otwell, A. E., López García De Lomana, A., Gibbons, S. M., Orellana, M. V. & Baliga, N. S. Systems biology approaches towards predictive microbial ecology. *Environmental Microbiology* **20,** 4197–4209. doi:10.1111/1462-2920.14378 (2018).

25. Purdy, K. J. *et al.* in *Integrative Ecology: From Molecules to Ecosystems* (ed Woodward, G.) 87–149 (Academic Press, 2010). doi:https://doi.org/10.1016/B978-0-12-385005-8.00003-4. http://www.sciencedirect.com/science/article/pii/B9780123850058000034.

26. Gu, C., Kim, G. B., Kim, W. J., Kim, H. U. & Lee, S. Y. Current status and applications of genome-scale metabolic models. *Genome Biology* **20,** 121. doi:10.1186/s13059-019-1730-3 (2019).

27. Thiele, I. & Palsson, B. Ø. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature Protocols* **5,** 93–121. doi:10.1038/nprot.2009.203 (2010).

28. Price, N. D., Reed, J. L., Papin, J. A., Wiback, S. J. & Palsson, B. O. Network-based analysis of metabolic regulation in the human red blood cell. *Journal of Theoretical Biology* **225,** 185–194. doi:https://doi.org/10.1016/S0022-5193(03)00237-6 (2003).

29. Stelling, J., Klamt, S., Bettenbrock, K., Schuster, S. & Gilles, E. D. Metabolic network structure determines key aspects of functionality and regulation. *Nature* **420,** 190–193. doi:10.1038/nature01166 (2002).

30. Orth, J. D., Thiele, I. & Palsson, B. Ø. What is flux balance analysis? *Nature Biotechnology* **28,** 245–248. doi:10.1038/nbt.1614 (2010).

31. Wang, X., Xiong, X., Sa, N., Roje, S. & Chen, S. Metabolic engineering of enhanced glycerol-3-phosphate synthesis to increase lipid production in Synechocystis sp. PCC 6803. *Applied Microbiology and Biotechnology* **100,** 6091–6101. doi:10.1007/s00253-016-7521-9 (2016).

32. J. Seppälä, J. *et al.* Prospecting hydrogen production of Escherichia coli by metabolic network modeling. *International Journal of Hydrogen Energy* **38,** 11780–11789. doi:10.1016/j.ijhydene.2013.07.002 (2013).

33. Lee, D.-S. *et al.* Comparative Genome-Scale Metabolic Reconstruction and Flux Balance Analysis of Multiple Staphylococcus aureus Genomes Identify Novel Antimicrobial Drug Targets. *Journal of Bacteriology* **191,** 4015–4024. doi:10.1128/JB.01743-08 (2009).

34. Tobalina, L., Pey, J., Rezola, A. & Planes, F. J. Assessment of FBA Based Gene Essentiality Analysis in Cancer with a Fast Context-Specific Network Reconstruction Method. *PLOS One* **11,** 1–17. doi:10.1371/journal.pone.0154583 (2016).

35. Mahadevan, R., Edwards, J. S. & Doyle, F. J. Dynamic flux balance analysis of diauxic growth in Escherichia coli. *Biophysical Journal* **83 3,** 1331–40 (2002).

36. Khandelwal, R. A., Olivier, B. G., Röling, W. F. M., Teusink, B. & Bruggeman, F. J. Community Flux Balance Analysis for Microbial Consortia at Balanced Growth. *PLOS One* **8,** 1–10. doi:10.1371/journal.pone.0064567 (2013).

37. Lularevic, M., Racher, A. J., Jaques, C. & Kiparissides, A. Improving the accuracy of flux balance analysis through the implementation of carbon availability constraints for intracellular reactions. *Biotechnology and Bioengineering* **116,** 2339–2352. doi:10.1002/bit.27025 (2019).

38. Sánchez, B. J. *et al.* Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. *Molecular Systems Biology* **13,** 935. doi:10.15252/msb.20167411 (2017).

39. Adadi, R., Volkmer, B., Milo, R., Heinemann, M. & Shlomi, T. Prediction of Microbial Growth Rate versus Biomass Yield by a Metabolic Network with Kinetic Parameters. *PLOS Computational Biology* **8,** 1–9. doi:10.1371/journal.pcbi.1002575 (2012).

40. Bekiaris, P. S. & Klamt, S. Automatic construction of metabolic models with enzyme constraints. *BMC Bioinformatics* **21,** 19. doi:10.1186/s12859-019-3329-9 (2020).

41. Beg, Q. K. *et al.* Intracellular crowding defines the mode and sequence of substrate uptake by Escherichia coli and constrains its metabolic activity. *Proceedings of the National Academy of Sciences* **104,** 12663–12668. doi:10.1073/pnas.0609845104 (2007).

42. Karr, J. R. *et al.* A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell* **150,** 389–401. doi:10.1016/j.cell.2012.05.044 (2012).

43. O'Brien, E. J., Lerman, J. A., Chang, R. L., Hyduke, D. R. & Palsson, B. Ø. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular Systems Biology* **9,** 693–693. doi:10.1038/msb.2013.52 (2013).

44. Alzoubi, D., Desouki, A. A. & Lercher, M. J. Flux balance analysis with or without molecular crowding fails to predict two thirds of experimentally observed epistasis in yeast. *Scientific Reports* **9,** 11837. doi:10.1038/s41598-019-47935-6 (2019).

45. MacGillivray, M. *et al.* Robust Analysis of Fluxes in Genome-Scale Metabolic Pathways. *Scientific Reports* **7,** 268–268. doi:10.1038/s41598-017-00170-3 (2017).

46. Nocedal, J. & Wright, S. J. *Numerical Optimization* 2nd edition (Springer, New York, NY, 2006).

47. Boyd, S. P. & Vandenberghe, L. *Convex Optimization* (Cambridge University Press, Cambridge, England, 2004).

48. Schweinzer, P. *Mathematical Methods for Economic Analysis* https://pdfs.semanticscholar.org/c90d/acecc1c968cfc32cdfd01a039190d0afd52d.pdf (2011).

49. Lundgren, J., Ronnqvist, M. & Varbrand, P. *Optimization* (Studentlitteratur, Lund, Sweden, 2010).

50. Niu, S.-C. *The simplex method* https://personal.utdallas.edu/~scniu/OPRE-6201/documents/LP4-Simplex.html.

51. Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual* http://www.gurobi.com (2020).

52. Xpress Optimization, FICO. *Xpress-Optimizer Reference Manual 27.01* https://www.msi-jp.com/xpress/learning/square/optimizer-2015.pdf (2015).

53. IBM Corporation. *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual 12.7* https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/pdf/usrcplex.pdf (2016).

54. Ben-Tal, A. & Nemirovski, A. Robust solutions of uncertain linear programs. *Operations Research Letters* **25,** 1–13. doi:https://doi.org/10.1016/S0167-6377(99)00016-4 (1999).

55. Potra, F. A. & Wright, S. J. Interior-point methods. *Journal of Computational and Applied Mathematics* **124,** 281–302. doi:10.1016/s0377-0427(00)00433-7 (2000).

56. Ben-Tal, A. & Nemirovski, A. Robust Solutions of Uncertain Linear Programs. *Operations Research Letters* **25,** 1–13. doi:10.1016/S0167-6377(99)00016-4 (1999).

57. MOSEK ApS. *MOSEK Modeling Cookbook 3.2.1* https://docs.mosek.com/modeling-cookbook/index.html (2019).

58. Gurobi Optimization, LLC. *Gurobi Optimizer Webpage* https://www.gurobi.com/products/gurobi-optimizer/ (2020).

59. Lu, H. *et al.* A consensus S. cerevisiae metabolic model Yeast8 and its ecosystem for comprehensively probing cellular metabolism. *Nature Communications* **10,** 3586 (2019).

60. Norsigian, C. J. *et al.* BiGG Models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic Acids Research* **48,** D402–D406. doi:10.1093/nar/gkz1054 (2019).

61. Angione, C. Human Systems Biology and Metabolic Modelling: A Review–From Disease Metabolism to Precision Medicine. *BioMed Research International* **2019,** 1–16. doi:10.1155/2019/8304260 (2019).

62. Nielsen, J. Systems Biology of Metabolism. *Annual Review of Biochemistry* **86,** 245–275. doi:10.1146/annurev-biochem-061516-044757 (2017).

63. Stelling, J., Klamt, S., Bettenbrock, K., Schuster, S. & Gilles, E. D. Metabolic network structure determines key aspects of functionality and regulation. *Nature* **420,** 190–193. doi:10.1038/nature01166 (2002).

64. Schilling, C. H., Letscher, D. & Palsson, B. Ø. Theory for the Systemic Definition of Metabolic Pathways and their use in Interpreting Metabolic Function from a Pathway-Oriented Perspective. *Journal of Theoretical Biology* **203,** 229–248. doi:10.1006/jtbi.2000.1073 (2000).

65. Barrett, C. L., Herrgard, M. J. & Palsson, B. Decomposing complex reaction networks using random sampling, principal component analysis and basis rotation. *BMC Systems Biology* **3,** 30. doi:10.1186/1752-0509-3-30 (2009).

66. Li, G., Cao, H. & Xu, Y. Structural and functional analyses of microbial metabolic networks reveal novel insights into genome-scale metabolic fluxes. *Briefings in Bioinformatics* **20,** 1590–1603. doi:10.1093/bib/bby022 (2019).

67. Segrè, D., Vitkup, D. & Church, G. M. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences* **99,** 15112–15117. doi:10.1073/pnas.232349399 (2002).

68. Navid, A. & Almaas, E. Genome-level transcription data of Yersinia pestis analyzed with a new metabolic constraint-based approach. *BMC Systems Biology* **6,** 150–150. doi:10.1186/1752-0509-6-150 (2012).

69. Almaas, E., Kovács, B., Vicsek, T., Oltvai, Z. N. & Barabási, A.-L. Global organization of metabolic fluxes in the bacterium Escherichia coli. *Nature* **427,** 839–843. doi:10.1038/nature02289 (2004).

70. Yim, H. *et al.* Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol. *Nature Chemical Biology* **7,** 445–452. doi:10.1038/nchembio.580 (2011).

71. Mardinoglu, A. *et al.* Genome-scale metabolic modelling of hepatocytes reveals serine deficiency in patients with non-alcoholic fatty liver disease. *Nature Communications* **5.** doi:10.1038/ncomms4083 (2014).

72. Zelezniak, A. *et al.* Metabolic dependencies drive species co-occurrence in diverse microbial communities. *Proceedings of the National Academy of Sciences* **112,** 6449–6454. doi:10.1073/pnas.1421834112 (2015).

73. Lewis, N., Nagarajan, H. & Palsson, B. Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods. *Nature Reviews Microbiology* **10,** 291–305. doi:10.1038/nrmicro2737 (2012).

74. Sánchez, B. J., Li, F., Kerkhoven, E. J. & Nielsen, J. SLIMEr: probing flexibility of lipid metabolism in yeast with an improved constraint-based modeling framework. *BMC Systems Biology* **13,** 4. doi:10.1186/s12918-018-0673-8 (2019).

75. Dikicioglu, D., Kirdar, B. & Oliver, S. G. Biomass composition: the "elephant in the room" of metabolic modelling. *Metabolomics* **11,** 1690–1701. doi:10.1007/s11306-015-0819-2 (2015).

76. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19,** 524–531. doi:10.1093/bioinformatics/btg015 (2003).

77. Becker, S. A. *et al.* Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. *Nature Protocols* **2,** 727–738. doi:10.1038/nprot.2007.99 (2007).

78. Agren, R. *et al.* The RAVEN Toolbox and Its Use for Generating a Genome-scale Metabolic Model for Penicillium chrysogenum. *PLOS Computational Biology* **9,** 1–16. doi:10.1371/journal.pcbi.1002980 (2013).

79. King, Z. A. *et al.* BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research* **44,** D515–D522. doi:10.1093/nar/gkv1049 (2016).

80. Fong, S. S. & Palsson, B. Ø. Metabolic gene-deletion strains of Escherichia coli evolve to computationally predicted growth phenotypes. *Nature Genetics* **36,** 1056–1058. doi:10.1038/ng1432 (2004).

81. Ibarra, R. U., Edwards, J. S. & Palsson, B. O. Escherichia coli K-12 undergoes adaptive evolution to achieve in silico predicted optimal growth. *Nature* **420,** 186–189. doi:10.1038/nature01149 (2002).

82. Dromms, R. A., Lee, J. Y. & Styczynski, M. P. LK-DFBA: a linear programming-based modeling strategy for capturing dynamics and metabolite-dependent regulation in metabolism. *BMC Bioinformatics* **21.** doi:10.1186/s12859-020-3422-0 (2020).

83. Motamedian, E., Mohammadi, M., Shojaosadati, S. A. & Heydari, M. TRFBA: an algorithm to integrate genome-scale metabolic and transcriptional regulatory networks with incorporation of expression data. *Bioinformatics* **33,** 1057–1063. doi:10.1093/bioinformatics/btw772 (2016).

84. Marmiesse, L., Peyraud, R. & Cottret, L. FlexFlux: combining metabolic flux and regulatory network analyses. *BMC Systems Biology* **9,** 93. doi:10.1186/s12918-015-0238-z (2015).

85. Birch, E. W., Udell, M. & Covert, M. W. Incorporation of flexible objectives and time-linked simulation with flux balance analysis. *Journal of Theoretical Biology* **345,** 12–21. doi:https://doi.org/10.1016/j.jtbi.2013.12.009 (2014).

86. Nelson, D. L. *Principles of Biochemistry* 6th edition (W.H. Freeman and Company, New York, NY, 2013).

87. Chang, R. & Goldsby, K. *General Chemistry: The Essential Concepts* 7th edition (McGraw-Hill, New York, NY, 2014).

88. Motulsky, H. *Equation: Michaelis-Menten Model* https://www.graphpad.com/guides/prism/7/curve-fitting/reg_michaelis_menten_enzyme.htm.

89. Uludag-Demirer, S., Duran, J. & Tanner, R. D. Estimating the turnover number in enzyme kinetic reactions using transient and stationary state data. *Brazilian Journal of Pharmaceutical Sciences* **45,** 635–642 (2009).

90. Davidi, D. *et al.* Global characterization of in vivo enzyme catalytic rates and their correspondence to in vitro kcat measurements. *Proceedings of the National Academy of Sciences* **113,** 3401–3406. doi:10.1073/pnas.1514240113 (2016).

91. Massaiu, I. *et al.* Integration of enzymatic data in Bacillus subtilis genome-scale metabolic model improves phenotype predictions and enables in silico design of poly-γ-glutamic acid production strains. *Microbial Cell Factories* **18,** 3. doi:10.1186/s12934-018-1052-2 (2019).

92. Python. *Python programming language—official website* https://www.python.org/ (2019).

93. Ebrahim, A., Lerman, J. A., Palsson, B. O. & Hyduke, D. R. COBRApy: COnstraint-Based Reconstruction and Analysis for Python. *BMC Systems Biology* **7,** 74. doi:10.1186/1752-0509-7-74 (2013).

94. Hart, W., Watson, J.-P., Woodruff, D. & Watson, J.-P. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation* **3,** 219–260. doi:10.1007/s12532-011-0026-8 (2011).

95. Schomburg, I. *et al.* The BRENDA enzyme information system–From a database to an expert system. *Journal of Biotechnology* **261,** 194–206. doi:https://doi.org/10.1016/j.jbiotec.2017.04.020 (2017).

96. Jeske, L., Placzek, S., Schomburg, I., Chang, A. & Schomburg, D. BRENDA in 2019: a European ELIXIR core data resource. *Nucleic Acids Research* **47,** D542–D549. doi:10.1093/nar/gky1048 (2018).

97. Lieven, C. *et al.* MEMOTE for standardized genome-scale metabolic model testing. *Nature Biotechnology* **38,** 272–276. doi:10.1038/s41587-020-0446-y (2020).

98. MathWorks. *MATLAB programming language—official website* https://se.mathworks.com/products/matlab.html (2019).

99. Python Software Foundation. *What is Python?* https://docs.python.org/3/faq/general.html#what-is-python (2020).

100. Zotter, A., Bäuerle, F., Dey, D., Kiss, V. & Schreiber, G. Quantifying enzyme activity in living cells. *The Journal of Biological Chemistry* **292,** 15838–15848. doi:10.1074/jbc.M117.792119 (2017).

101. Raser, J. M. & O'Shea, E. K. Noise in Gene Expression: Origins, Consequences, and Control. *Science* **309,** 2010–2013. doi:10.1126/science.1105891 (2005).

102. Huh, D. & Paulsson, J. Random partitioning of molecules at cell division. *Proceedings of the National Academy of Sciences* **108,** 15004–15009. doi:10.1073/pnas.1013171108 (2011).

103. Soltani, M., Vargas-Garcia, C. A., Antunes, D. & Singh, A. Intercellular Variability in Protein Levels from Stochastic Expression and Noisy Cell Cycle Processes. *PLOS Computational Biology* **12,** 1–23. doi:10.1371/journal.pcbi.1004972 (2016).

104. Wittig, U. *et al.* SABIO-RK—database for biochemical reaction kinetics. *Nucleic Acids Research* **40,** D790–D796. doi:10.1093/nar/gkr1046 (2011).

Appendix A

A.1 Nonuniqueness of uncertainty matrices

We will here show that diagonal covariance matrices, $Q_{\text{diag}} \in \mathbb{R}^{n\times n}$, can be factorized by an infinite number of matrices, R, such that $Q_{\text{diag}} = R^T R$.

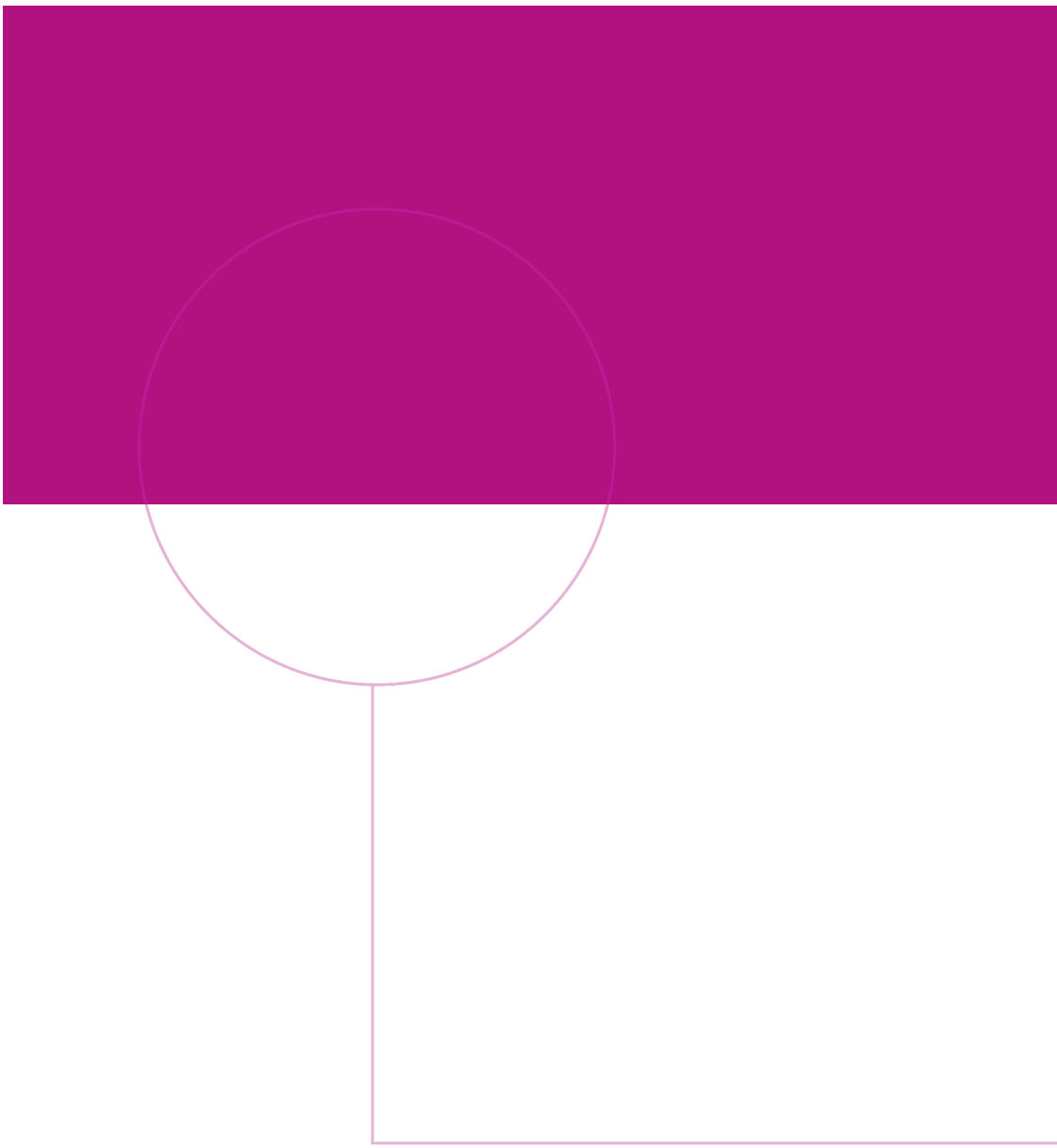
First, we denote the index set of nonzero diagonal entries of Q_{diag} as $\mathcal{I}$. That is, we define $\mathcal{I} = \{i : \text{ the } (i,i)^{\text{th}} \text{ entry of } Q_{\text{diag}} \text{ is nonzero}\}$. As $|\mathcal{I}| \leq n$, we are guaranteed the existence of a set of linearly independent vectors, $\{v_i\}_{i\in\mathcal{I}}$, in $\mathbb{R}^n$. Now perform Gram-Schmidt orthogonalization on that set to obtain a set of orthonormal vectors $\{e_i\}_{i\in\mathcal{I}}$. Now define, for $i = 1 \ldots n$, the following vectors

$$r_i = \begin{cases} \sigma_i e_i & \text{if } i \in \mathcal{I} \\ 0 & \text{if } i \notin \mathcal{I}, \end{cases}$$

where σ_i is the $(i,i)^{\text{th}}$ element of Q_{diag} and 0 is the zero vector. Now define R to be the n-by-n matrix with r_i as its i^{th} column. Let $A = R^T R$ and denote the $(i,j)^{\text{th}}$ entry of A by a_{ij}. We then have

$$a_{ij} = r_i^T r_j = \begin{cases} \sigma_i^2 & \text{if } i = j \\ 0 & \text{if } i \neq j, \end{cases}$$

hence $A = Q_{\text{diag}}$, and consequently $Q_{\text{diag}} = R^T R$. Our uncertainty matrix may thus contain any collection of column vectors corresponding to an orthogonal set of $|\mathcal{I}|$ vectors of lengths equaling the random variable variances. For any dimension $|\mathcal{I}|$, we clearly have an infinite number of such sets, and hence an infinite number of distinct R matrices that satisfy $R^T R = Q_{\text{diag}}$.



NTNU

Kunnskap for en bedre verden