Modeling and Simulation of Enzyme Controlled Metabolic Networks Using Optimization Based Methods

Dissertation
zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

von
Henning Lindhorst
geboren am 10.April 1985 in Paderborn

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik der
Otto-von-Guericke-Universität Magdeburg

Gutachter:
Prof. Dr.-Ing. Joachim Kienle
Prof. Dr.-Ing. Steffen Waldherr
Prof. Dr.rer.nat. Alexander Bockmayr

eingereicht am 14. April 2019
Promotionskolloquium am 22. Januar 2020

Contents

Abstract I

Deutsche Kurzfassung III

1 Introduction 1
- 1.1 Metabolic networks . . . 2
- 1.2 Modeling metabolism . . . 5
  - 1.2.1 Dynamic models . . . 5
  - 1.2.2 Constraint-based methods . . . 7
- 1.3 Goals and contributions . . . 9
- 1.4 Structure of this work . . . 10

2 Resource allocation methods 13
- 2.1 Resource allocation in a self-replicator . . . 13
- 2.2 Resource Balance Analysis . . . 17
  - 2.2.1 ME models . . . 20
- 2.3 deFBA . . . 22
  - 2.3.1 Species . . . 23
  - 2.3.2 Reactions . . . 24
  - 2.3.3 Time-scale separation . . . 24
  - 2.3.4 Enzyme capacity constraint . . . 26
  - 2.3.5 Biomass composition constraints . . . 28
  - 2.3.6 Objective functional . . . 29
  - 2.3.7 Optimization problem . . . 31
  - 2.3.8 Conditional FBA . . . 31
- 2.4 Conclusion . . . 32

3 Building, exchanging and evaluating deFBA models 35
- 3.1 Extending the deFBA formalism . . . 35
  - 3.1.1 Refining the macromolecules . . . 35
  - 3.1.2 Maintenance reactions . . . 36
  - 3.1.3 Extended dynamics and lumped reactions . . . 37
  - 3.1.4 Objective biomass . . . 39
  - 3.1.5 Optimization problem . . . 40

3.2 SBML extension - Resource Allocation Modeling . . . 40
  3.2.1 SBML header . . . 41
  3.2.2 The model . . . 41
  3.2.3 Compartmentalization . . . 42
  3.2.4 Parameters . . . 42
  3.2.5 Species . . . 43
  3.2.6 Encoding genetic information . . . 45
  3.2.7 Reactions . . . 47
  3.2.8 Naming conventions . . . 49
  3.2.9 Future of RAM . . . 49
3.3 Generating deFBA models . . . 50
  3.3.1 Prerequisites . . . 50
  3.3.2 Building enzyme production . . . 50
  3.3.3 Setting up quota . . . 52
  3.3.4 Determining turnover numbers . . . 54
  3.3.5 Molecular weights & objective weights . . . 55
  3.3.6 Model validation . . . 56
3.4 deFBA-Python package . . . 56
  3.4.1 Linopt class . . . 57
  3.4.2 DefbaModel class . . . 57
  3.4.3 RAM interface . . . 59
3.5 Conclusion . . . 61

4 Short-term deFBA **63**
4.1 Why using a receding prediction horizon? . . . 63
  4.1.1 Enzymatic-growth model . . . 64
4.2 Implementing the receding time horizon . . . 66
4.3 Choosing the prediction horizon . . . 68
  4.3.1 Improving on the boundaries for the growth rates . . . 74
4.4 Choosing the iteration time . . . 75
4.5 Short-term deFBA in `deFBA-Python` . . . 78
4.6 Numerical examples . . . 80
  4.6.1 Enzymatic growth model revisited . . . 80
  4.6.2 Carbon-core model . . . 81
    4.6.2.1 Carbon source switch . . . 83
    4.6.2.2 Aerobic - anaerobic switching . . . 86

5 Robust deFBA **89**
5.1 Multi-stage MPC . . . 89
5.2 Constructing a single optimization problem . . . 91
5.3 Uncertainty in the environment . . . 94

5.4 Implementing the robust deFBA 97
5.5 Solving rdeFBA with `deFBA-Python` 98
5.6 Numerical examples 100
    5.6.1 Carbon-core model 101
        5.6.1.1 Carbon source switching 101
        5.6.1.2 Aerobic-anaerobic switching 102
    5.6.2 Yeast model 103
5.7 Conclusion 109

6 Conclusions **111**

A Models **115**
A.1 Resalloc model 115

B Results for deFBA_yeast model **119**

Bibliography **121**

List of symbols and abbreviations **131**

Abstract

This PhD thesis contributes to the modeling and the numerical simulation of metabolic reaction networks coupled with gene expression via the inclusion of gene products as catalysts for the reactions. These networks describe the foundation of all (bio-)chemical processes in any living cell.

A widespread method for modeling and analyzing these networks is the Flux Balance Analysis, which uses a stoichiometric description of the network and experimentally determined reaction rates to predict pathway usage, growth rates and by-products. This method helps in understanding the inner workings of metabolic networks but is restrained to static environments and relies on experimentally measured reaction rates.

The basis for this work is an extension of this method called dynamic enzyme-cost Flux Balance Analysis (deFBA). It models the metabolism as a resource allocation problem with the goal to maximize biomass accumulation in a dynamic environment. As this method is quite young, we start this work by introducing new standards for the deFBA with regards to the creation, saving and evaluation of the models. A result of this standardization is a software package in Python allowing simple access and evaluation to the models.

The deFBA is limited by its reliance on a deterministic description of the environmental dynamics. But single-celled organisms have learned to prepare for unforeseen and sudden shifts in the availability of nutrients. To be able to model this robustness we developed a new simulation environment using a combination of the deFBA with the multi-stage Model Predictive Control. This shifts the objective of the modeled cells from optimal adaptation to the known environment to the more general goal of survival while retaining as much growth rate as possible. We expect this method to support the identification of the mechanisms leading to the robustness against environmental changes.

Deutsche Kurzfassung

Diese Doktorarbeit beschäftigt sich mit der Modellierung und numerischen Simulation von metabolischen Netzwerken, welche die (bio-)chemischen Vorgänge innerhalb von einzelligen Organismen beschreiben.

Eine weit verbreitete Modellierungsmethode für diese Netzwerke ist die Flux Balance Analysis, welche alleine über eine grundlegende stoichiometrische Beschreibung des Netzwerks und experimentell gesessene Reaktionsraten Aussagen über Wachstumsraten und Nebenprodukte treffen kann. Während diese Methodik bereits tiefe Einblicke in die Struktur und Funktionsweise von metabolischen Netzwerken geben kann, ist sie beschränkt auf statische Umgebungen und die Ergebnisse sind stark von den eingesetzten Reaktionsraten abhängig.

Die Grundlage dieser Arbeit ist eine Erweiterung dieser Methode namens dynamic enzyme-cost Flux Balance Analysis (deFBA). Diese versteht den Metabolismus als ein Ressourcenallokations-Problem mit dem Ziel die Akkumulation von Biomasse in einer dynamischen Umgebung zu maximieren. Da dieser Ansatz noch sehr jung ist, beginnt unseren Beitrag damit die Handhabe dieser neuen Methodik zu standardisieren in Bezug auf Erstellung, Speicherung und Evaluation der Modelle. Das Resultat dieser Vereinheitlichungen ist unter anderem ein Software Paket in Python, welches einen einfachen Zugang zu diesen Modellen bereit stellt.

Der Nachteil der deFBA liegt darin, dass die durch mathematische Optimierung erzielten Ergebnisse sehr genau auf die aktuelle Nahrungssituation zugeschnitten sind. Tatsächlich sehen wir allerdings in Wildtypen von Einzellern, dass ein großer Anteil der verfügbaren Ressourcen in die Vorbereitung auf mögliche Störungen in der Nährstoffversorgung investiert werden. Um diese sogenannte Robustheit zu modellieren, entwickeln wir in dieser Arbeit eine neue Simulationsumgebung, welche die deFBA mit der multi-stage Model Predictive Control kombiniert. Diese neue Methode erlaubt es Vorherzusagen, wie sich Einzeller an unsichere Nährstoffverfügbarkeiten anpassen, und kann uns dabei helfen zu verstehen wie diese Robustheit durch die Regelung der Genexpression umgesetzt wird.

1 Introduction

> It seems to me that the view toward which we are tending is that the specificity in gene action is always a chemical specificity, probably the production of enzymes which guide metabolic processes along particular channels. A given array of genes thus determines the production of a particular kind of protoplasm with particular properties - such, for example, as that of responding to surface forces by the formation of a special sort of semipermeable membrane, and that of responding to trivial asymmetries in the play of external stimuli by polarization, with consequent orderly quantitative gradients in all physiologic processes. Different genes may now be called into play at different points in this simple pattern, either through the local formation of their specific substrates for action, or by activation of a mutational nature. In either case the pattern becomes more complex and qualitatively differentiated. Successive interactions of differentiated regions and the calling into play of additional genes may lead to any degree of complexity of pattern in the organism as a largely self-contained system. The array of genes, assembled in the course of evolution, must of course be one which determines a highly self regulatory system of reactions. On this view the genes are highly specific chemically, and thus called into play only under very specific conditions; but their morphological effects, if any, rest on quantitative influences of immediate or remote products on growth gradients, which are resultants of all that has gone on before in the organism.
>
> Sewall Wright, in *'Genetics of Abnormal Growth in the Guinea Pig', Cold Spring Harbor Symposia on Quantitative Biology (1934)*

When reading the words of Mr. Wright roughly 90 years later, it is fascinating to see how the understanding of the genome and its expression in all living cells has evolved. Of course, Mr. Wright was mostly hypothesizing at this point as the experimental methods to prove his words were simply not invented yet. Nowadays, we access to modern methods to analyze the genome including high data throughput methods like whole genome sequencing. These allow us to study the deoxyribonucleic acid (DNA) of individual cells quickly and at a low cost per experiment. Following the words from Mr. Wright, all gene actions follow a ”chemical specificity” and are called into play at a ”simple pattern”. Therefore, it should easily be possible to derive a full set of the cellular reactions for a cell to any outside disturbances just from its DNA alone.

Unfortunately, the more we learned about how gene expression works, what the gene products are, and which role they play in the metabolism of cell, the more we faced how little we really know about the process. While it is true that many of these gene products are enzymes, which enable or regulate specific metabolic reactions inside the cell, there exists whole families of proteins Mr. Wright could have never known of. There are, for example, chaperone proteins working to stabilize and fold other synthesized proteins. They regulate enzymes such that reaction products only are created where wanted. Other proteins are merely transporters tasked with the transport of metabolites and other proteins to their destination. Membrane proteins regulate the uptake or secretion of chemicals. Others are used as messages which can be sent to other cells via the medium the cells live in. And these are just some examples of functions of gene products inside the cells.

Even if we knew all functions of the products, we still could not state with full certainty how the cells regulate themselves completely as other factors like epigenetics and the existence of phenotypes are not included yet. While there exist projects to include every detail of a single cell into very sophisticated Whole-Cell models [54], the creation of the models takes years and large amounts of manpower to even have a chance of being successful.

In this work, we want to present and develop methods, which bypass the problem of unknown regulations inside the cell. Instead these methods assume that the internal regulation of cells has evolved over millennia to optimize certain aspects, like achieving maximal growth rates under specific conditions or surviving in very dynamic environments. The idea is to translate this assumption into a mathematical optimization problem and predict the cells' behavior via the solution of the optimization. We place a focus on dynamic environments in this work, meaning availability of nutrients can change quickly with or without the influence of the cells. Our final goal is to predict how cells adapt to these rapidly changing environments and locate the cellular functions making the cells robust against the constant pressure to adapt.

At the core of these *constraint based methods* lies always a metabolic network and the chemical reactions enabling the cell to live. We give a short introduction into these in the next section.

1.1 Metabolic networks

The metabolism of a cell can be explained via *metabolic networks* which are directed hyper-graphs describing all reactions between the metabolites. We consider (bio-)chemical reactions in the stoichiometric form, e.g. the formation of water

$$2\ \mathrm{H} + 1\ \mathrm{O}_2 \leftrightarrow 1\ \mathrm{H}_2\mathrm{O}, \tag{1.1}$$

where the given amounts of each species are called the *stoichiometric coefficients* of the reaction. For easier reading coefficients with value 1 are usually omitted. The Figure 1.1 depicts a simple metabolic network and the transport of matter along the system border. The corresponding reaction system is given as $\mathrm{A}\leftrightarrow\mathrm{A}_{\mathrm{ext}}$, $\mathrm{B}\leftrightarrow\mathrm{B}_{\mathrm{ext}}$, $\mathrm{C}\leftrightarrow\mathrm{C}_{\mathrm{ext}}$, $\mathrm{D}_{\mathrm{ext}} \rightarrow\mathrm{D}$, $\mathrm{F}\leftrightarrow\mathrm{F}_{\mathrm{ext}}$, $\mathrm{G}\rightarrow\mathrm{G}_{\mathrm{ext}}$, $\mathrm{A}\rightarrow\mathrm{B}+\mathrm{C}$, $\mathrm{B}\leftrightarrow\mathrm{E}$, $\mathrm{C}\rightarrow\mathrm{D}$, $\mathrm{D}\rightarrow\mathrm{E}$, $\mathrm{E}\leftrightarrow\mathrm{F}$, $\mathrm{E}\rightarrow\mathrm{G}$ with the *external metabolites* $\mathrm{A}_{\mathrm{ext}}$, $\mathrm{B}_{\mathrm{ext}}$, $\mathrm{C}_{\mathrm{ext}}$, $\mathrm{D}_{\mathrm{ext}}$, $\mathrm{F}_{\mathrm{ext}}$, $\mathrm{G}_{\mathrm{ext}}$ and the *internal metabolites* A, B, C, D, E, F, G. We distinguish between *exchange reactions* transporting matter along the system border and *internal reactions* only involving internal metabolites.

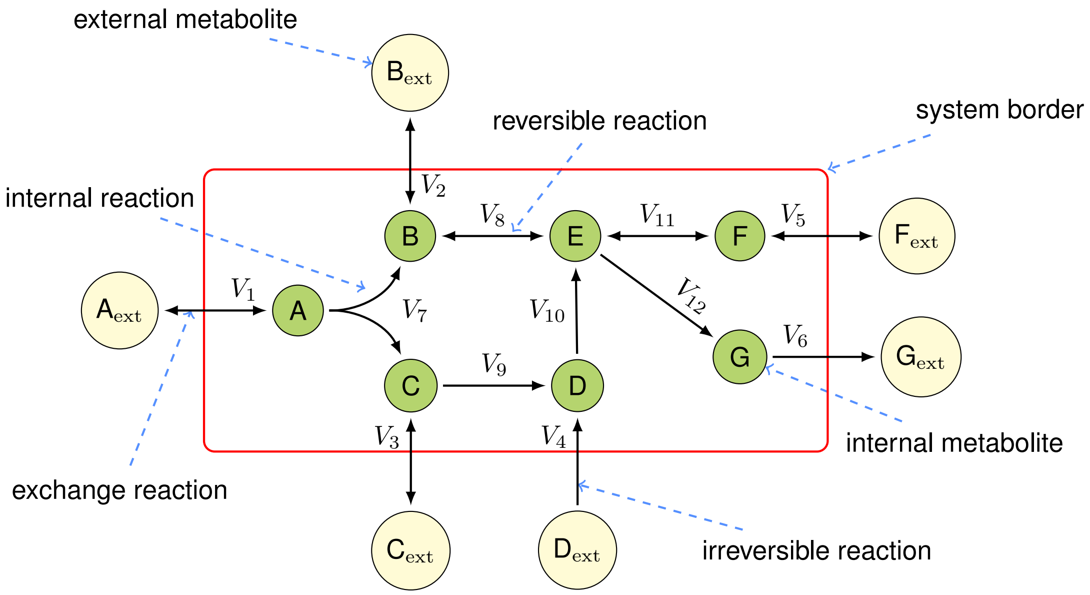


Figure 1.1: Simple representation of a metabolic network.

With the given stoichiometry, we can translate the information about the structure of the metabolic network to the *stoichiometric matrix* $S \in \mathbb{R}^{13,12}$ as

$$S = \begin{array}{c|cccccccccccc|}
 & V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & V_9 & V_{10} & V_{11} & V_{12} \\
A_{\text{ext}} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B_{\text{ext}} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_{\text{ext}} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D_{\text{ext}} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F_{\text{ext}} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
G_{\text{ext}} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
A & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
D & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 & -1 \\
F & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
G & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}, \tag{1.2}$$

which reflects the incidence matrix of the underlying graph with the rows of S corresponding to the species (nodes) and columns to reactions (edges). But instead of being limited to the entries $s_{i,j} \in \{-1, 0, 1\}$ the stoichiometric coefficients are used as entries, e.g. reaction (1.1) has the stoichiometric matrix

$$S = \begin{array}{cc} & V \\ \begin{array}{c} \mathrm{H} \\ \mathrm{O}_2 \\ \mathrm{H}_2\mathrm{O} \end{array} & \begin{pmatrix} -2 \\ -1 \\ 1 \end{pmatrix} \end{array} . \tag{1.3}$$

For a general notion we collect metabolites in the vector $X \in \mathbb{R}^{n_x}_{\geq 0}$ and the external species in $Y \in \mathbb{R}^{n_y}_{\geq 0}$, $[X] = [Y] = \mathrm{mol}$. We address the *reaction rates* or *fluxes* of the network as $V \in \mathbb{R}^m$, $[V] = \mathrm{mol/h}$. The reactions are either defined as *irreversible* with $V_{\mathrm{irr}} \geq 0$ or *reversible* without additional bounds. To differentiate between different biological functions of the fluxes we use the index sets y for exchange reactions as $V_y \in \mathbb{R}^{m_y}$ and x for metabolic fluxes $V_x \in \mathbb{R}^{m_x}$ acting only on metabolites inside the cell.

In the same manner, we introduce the sets $\mathcal{Y}$, $\mathcal{X}$ to identify individual species types. This way we can easily access submatrices of S, e.g., the example network consists of the submatrices

$$S = \begin{pmatrix} S_{\mathcal{Y},y} & S_{\mathcal{Y},x} \\ S_{\mathcal{X},y} & S_{\mathcal{X},x} \end{pmatrix} . \tag{1.4}$$

Additionally, we use a dot $(\cdot)$ to access all rows (columns) at once, e.g., the matrix $S_{\mathcal{Y},\cdot} \in \mathbb{R}^{7,12}$ describes all rows of S with respect to the external species. During this work we will gradually introduce more variables and according index sets. A complete list can be found in Appendix B.

This work mainly uses variables expressed in molar amounts, but in rare cases we will also use molar concentrations. They are depicted in bold fonts, e.g. $\mathbf{X}$, $[\mathbf{X}] = \mathrm{mol/L}$ and resp. $[\mathbf{V}] = \mathrm{mol/(L\ h)}$. Please note, that the formulation of fluxes in molar concentrations should usually be avoided, especially if multiple compartments are involved. As this is not the case in our applications, we stick to this simple formulation.

The main concern of this work is the dynamic modeling of metabolic networks. Time is depicted as t, $[t] = \mathrm{h}$ and we refer to single points of trajectories for species and reactions via $X(t) \in \mathbb{R}^{n_x}$. Time slices are written as $X([t_0, t_1]) = \{X(t)\ | t \in [t_0, t_1]\} \subset \mathbb{R}^{n_x}$. For time intervals we use square brackets for closed sets $[t_0, t_1] = \{t \mid t_0 \leq t \leq t_1\}$, rounds ones for open sets $(t_0, t_1) = \{t \mid t_0 < t < t_1\}$, and a mix for half-open sets $(t_0, t_1] = \{t \mid t_0 < t \leq t_1\}$.

The rate of change for metabolites is then defined by the difference of the reactions producing and depleting them

$$\frac{\mathrm{d}}{\mathrm{d}t}X(t) = \dot{X}(t) = V_{\text{production}} - V_{\text{consumption}} \tag{1.5}$$

Combining this with the known stoichiometry of the reactions we can simplify the dynamics for metabolites to

$$\dot{X}(t) = S_{\mathcal{X},\cdot}V(t). \tag{1.6}$$

For all models of metabolic networks the first and foremost problem is how we can determine the reaction rates $V(t)$. In the following section we look at some possible answers for this question, before we present the solution we are using in Chapter 2.

1.2 Modeling metabolism

1.2.1 Dynamic models

Focus in this work is on kinetic and constrained based modeling, other approaches and their relation have been reviewed recently in the book of Ramkrishna [88]. The most widespread model for the reaction rates $V(t)$ is using Michaelis-Menten kinetics [72]. These are given for an enzymatic reaction as

$$\mathrm{E} + \mathrm{S} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathrm{ES} \xrightarrow{k_{\text{cat}}} \mathrm{E} + \mathrm{P}, \tag{1.7}$$

with the enzyme E, substrate S, product P, complex ES and the positive rate constants k_f (forward rate), k_b (reverse or backward rate) and k_{cat} (catalytic rate). This model assumes that the binding between substrate and enzyme is a reversible process with the respective rates, while the formation of the product is irreversible. The enzyme is not depleted during the formation of P and is simply released again.

Using the *law of mass action* [31] we get four ODEs to define the dynamics of the involved species

$$\frac{\mathrm{d}\mathbf{E}}{\mathrm{d}t} = -k_f\mathbf{E}\cdot\mathbf{S} + k_b\mathbf{ES} + k_{\text{cat}}\mathbf{ES} \tag{1.8}$$

$$\frac{\mathrm{d}\mathbf{S}}{\mathrm{d}t} = -k_f\mathbf{E}\cdot\mathbf{S} + k_b\mathbf{ES} \tag{1.9}$$

$$\frac{\mathrm{d}\mathbf{ES}}{\mathrm{d}t} = k_f\mathbf{E}\cdot\mathbf{S} - k_b\mathbf{ES} - k_{\text{cat}}\mathbf{ES} \tag{1.10}$$

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = k_{\text{cat}}\mathbf{ES}, \tag{1.11}$$

with the forward rate constant k_f and the backward constant k_b. As the enzyme is neither depleted nor produced in this example the total concentration $\mathbf{E}$ + $\mathbf{ES}$

stays constant $\mathbf{E}(\mathrm{t}) = \mathbf{E}(\mathrm{t}{=}0) = \mathbf{E}_0$ and we can express the relation between free and bounded enzyme as

$$\mathbf{E} + \mathbf{ES} = \mathbf{E}_0. \tag{1.12}$$

Michaelis and Menten assumed in their original analysis [72] that substrate and complex are in chemical equilibrium $k_f\mathbf{E} \cdot \mathbf{S} = k_b\mathbf{ES}$. This leads directly to

$$k_f(\mathbf{E}_0 - \mathbf{ES}) \cdot \mathbf{S} = k_b\mathbf{ES} \Leftrightarrow \mathbf{ES} = \frac{\mathbf{E}_0\mathbf{S}}{K_d + \mathbf{S}}, \tag{1.13}$$

with the *dissociation constant* $K_d = k_b/k_f$. The rate of product formation can be described as

$$\mathbf{V}(t) = \frac{\mathrm{d}\mathbf{P}(\mathrm{t})(t)}{\mathrm{d}t} = V_{\max}\frac{\mathbf{S}(t)}{K_d + \mathbf{S}(t)} = k_{\mathrm{cat}}\mathbf{E}_0\frac{\mathbf{S}(t)}{K_d + \mathbf{S}(t)}, \tag{1.14}$$

with the maximum rate achievable $V_{\max} = k_{\mathrm{cat}}\mathbf{E}_0$.

An alternative formulation was derived by Briggs [14] using the quasi stady-state assumption instead of the chemical equilibrium

$$k_f\mathrm{E} \cdot \mathrm{S} = k_b\mathrm{ES} + k_{\mathrm{cat}}\mathrm{ES}. \tag{1.15}$$

Combining this with (1.12) yields a quite similar result

$$\mathbf{V}(t) = \frac{\mathrm{d}\mathbf{P}(\mathrm{t})(t)}{\mathrm{d}t} = V_{\max}\frac{\mathbf{S}(t)}{K_M + \mathbf{S}(t)} = k_{\mathrm{cat}}\mathbf{E}_0\frac{\mathbf{S}(t)}{K_M + \mathbf{S}(t)}, \tag{1.16}$$

but uses the *Michaelis constant* $K_M = (k_b + k_{\mathrm{cat}})/k_f$ instead of the dissociation constant. For the rest of this work we will focus on the Michaelis constant.

Depending on the concentration of the substrate, equation (1.16) can be further simplified. At very low concentrations $\mathbf{S} \ll K_M$, we can assume first-order kinetics are sufficiently accurate and write

$$\mathbf{V}(t) = V_{\max}\frac{\mathbf{S}(t)}{K_M}. \tag{1.17}$$

If $\mathbf{S}$ reaches saturation $\mathbf{S} >> K_M$ the reaction rate also converges to $V_{\max}$ and we can substitute (1.16) even with zero-order kinetics

$$\mathbf{V}(t) = V_{\max} = k_{\mathrm{cat}}\mathrm{E}_0. \tag{1.18}$$

Translating this equation means all enzyme is bound to the substrate and higher rates can only be achieved by an increase in available enzyme.

So far we only discussed single substrate reactions and have shown under which conditions their kinetics can be simplified. In genome-scale networks, we must deal with multi-substrate reaction, which rely on complex non-linear dynamics for modeling [45]. Using these non-linear dynamics in large scale metabolic networks has proven to be challenging in application. Combining this with the simple fact that the reaction and enzyme specific values for the Michaelis constant are usually not available, makes it in most cases not viable or even impossible to use Michaelis-Menten kinetics in genome-scale models. Nevertheless, there exist large-scale non-linear metabolic models, like k-ecoli457 [56], but their creation relies heavily on parameter fitting and can usually only be used in very specific scenarios. To simplify the model creation process and minimize the amount of experimental data needed a different approach called constraint-based modeling was invented.

1.2.2 Constraint-based methods

An alternative approach to the kinetic modeling are the constraint-based methods [11]. Their idea is to determine the unknown reaction rates by applying large amounts of biological and physical constraints to the networks and use a mathematical optimization to determine a flux distribution maximizing a biological objective like growth rate. Most of these methods assume metabolites are kept at *steady-state*, meaning production and depletion of all metabolites is balanced and their overall concentration levels stay constant. This assumption is motivated by the vast differences in the time-scales of cellular processes [105]. Nutrient uptake and metabolic reactions run very fast in comparison to gene expression and creation of biomass. A time-scale separation [110] then leads to a quasi steady-state constraint on the metabolic rates in the form

$$S\mathbf{V}(t) = 0 \;\Leftrightarrow\; \mathbf{V}(t) \in \ker(S), \tag{1.19}$$

with the $\ker(S)$ being the kernel of $S \in \mathbb{R}^{n,m}$. This constraint can be translated into the need for an effective metabolism as (1.19) prevents unnecessary accumulation of metabolites inside the cell. The cells basically implement a just-in-time production to minimize the space requirements for the metabolites. We will show a rigorous mathematical derivation of this explanation in Section 2.3.3.

But just constraining the fluxes to the kernel of the stoichiometric matrix will usually not yield a unique flux distribution as $\dim(\ker(S)) > 1$. The *Flux Balance Analysis* (FBA) [105] tries to identify the correct flux distribution by assuming the metabolism optimizes itself to maximize the rate of a biomass producing reaction. Because biomass components, such as enzymes, cell wall, etc., are not explicitly modeled so far, we introduce a single pseudo-reaction $\mathbf{V}_{\text{bio}} : \mathbb{R}^n \to \emptyset$ consuming the metabolites needed for growth like amino acids, ATP, fatty acids, etc. The flux through this reaction is to be maximized. We can infer additional constraints by enforcing the reaction flux through irreversible reactions to be positive $\mathbf{V}_{\text{irr}} \geq 0$ and present the mathematical description of the FBA as

$$\max_{\mathbf{V}\in\mathbb{R}^m} \mathbf{V}_{\text{bio}} \tag{1.20}$$

$$\text{s.t. } S\mathbf{V} = 0 \tag{1.21}$$

$$\mathbf{V}_{\text{irr}} \geq 0 \tag{1.22}$$

$$\mathbf{v}_{\text{min}} \leq \mathbf{V} \leq \mathbf{v}_{\text{max}}, \tag{1.23}$$

with the upper and lower reaction bounds $\mathbf{v}_{\text{min}}$ and $\mathbf{v}_{\text{max}}$. While it is not necessary to infer these bounds on all reactions in the system, key reactions must be constrained, e.g. the uptake of nutrients $\mathbf{V}_{\mathcal{Y}} < \infty$, to limit the rate of the biomass reaction to finite levels. On the other hand, one can include data on active reactions from experiments via the flux bounds.

This method can bring insight to some of the basic workings of the metabolism. For example it is possible to identify necessary nutrients to enable growth [30] or predict by-products during growth [80]. But at the same time this method is very limited as it can only determine a single flux distribution for the given reaction bounds. This means we can not directly model changes in e.g. the nutritional situation. Furthermore, the biomass reaction $\mathbf{V}_{\text{bio}}$ can only approximate the real need for nutrients as the biomass composition varies over time due to e.g. the cell cycle. This methods also completely neglects the necessity to translate the enzymes to even realize the reactions. This might lead to infeasible solutions if enzymes can not be translated in the given environment. An obvious advantage of the FBA is that we only have to solve a linear program (LP) for which fast and efficient solvers are available. Over the course of this work and especially in Chapter 2, we will present more advanced resource allocation methods which address the shortcomings of the FBA while sticking to the simple problem description via an LP.

Another point to discuss with the FBA is the assumption that the metabolism purely tries to maximize the biomass reaction or, more general, the growth rate. While this might be applicable to strains grown over a large number of generations under perfect laboratory conditions, we can not assume this to be true for wild types which are used to changes in their environment. Studies like [37] have investigated different objective functions for the FBA, e.g. maximization of ATP yield, minimization of reaction steps, minimization of byproduct yield or the regular maximization of biomass production. Unfortunately, the results strongly depend on the used model. While maximizing the biomass seems a reasonable approach for most cases for some models the results are very disappointing. Therefore, newer studies like [81] develop automated methods to determine model-specific objective functions. But these automatically generated objectives are calculated using experimental data and can loose their physical meaning up to a point that we can not understand what is even optimized.

Another approach is to find a formulation for the objective which reflects the overall goal of survival for the cell population. But it seems to be impossible to reduce this to a single objective as some aspects of survival might contradict each other. For example, a wild-type cell must be capable to grow as fast as possible in a given environment to be able to outgrow possible competitors for the nutrients. At the same time this wild-type must be able to react to changes in the environment, which means preparing some enzymes to enable pathways which are not used in the current environment. But this lowers the growth rate in the given environment. It usually does not make sense to compress these very different objectives into a single objective function with multiple weighted terms. While this combination might result in unique solutions, these will rely heavily on the chosen weights as these determine the relevance of the individual terms. Instead we can investigate contradicting objectives with tools from *multi-objective optimization* [27]. When using multiple objectives we do not investigate a single optimal solution anymore but look at the Pareto frontier (or Pareto set) of the problem. This is the set of Pareto optimal solutions, meaning no further improvements to any given objective can be made without harming another one. An application of this idea to a model of the bacterium *E.coli* can be found in [99]. The result of this study shows that the metabolism of *E.coli* works close to the Pareto frontier determined by three objectives: Minimizing the total flux, maximizing ATP yield, and maximizing the biomass yield. But the problem sustains that these objectives have to be chosen by hand and results are highly dependent on the ones chosen.

1.3 Goals and contributions

The basis for this work is one of the latest constrained-based modeling methods called the *dynamic enzyme-cost Flux Balance Analysis* (deFBA) [110], which we will present fully in Section 2.3. We strive to achieve two major objectives in this work: Extending the functionalities of the deFBA and making it easy for user to utilize the method.

A derivative of the deFBA, called the *constrained Flux Balance Analysis*, was already successfully applied to a genome-scaled model for cyanobacteria in the work [90]. The model in this work was created without underlying guidelines and is only available in a MATLAB specific data format. To enable users to easier create this type of model, we present systematic way to create deFBA models starting from gene-annotated FBA models, which was created in cooperation with A. Reimers and S. Waldherr [91]. To further enable the user to save and share the resulting models, we also devised an extension to the widely used *Systems Biology Markup Language* (SBML) called *Resource Allocation Modeling* (RAM). This new format can be used to save and load deFBA models, while ensuring a minimum of encoding errors to be present.

For the analysis of deFBA models we implemented a Python toolbox `deFBA-Python` which collects multiple simulation methods and supports import and export to RAM. A focus in the development of `deFBA-Python` was to keep the computational cost as low as possible to ensure that simulations can be performed on a regular personal computer. We implemented support for multiple linear solvers, namely `CPLEX` [102], `Gurobi` [43], `SoPlex` [114], and `cvxopt` [2], which all provide free academical licenses.

The need for low computational cost is also reflected in the introduction of a new deFBA method, called the *short-term deFBA*. This methods shifts the original approach of the deFBA with a single large optimization problem to a series of smaller optimization problems. Another benefit of this method that it decouples the optimization from a given end-time as this has proven to impact the quality of simulation results [109].

A downside of the optimization based approach used in deFBA methods is the need to operate these models in well-defined environmental conditions. The predicted enzyme-levels and growth rates can therefore only be compared to data collected under very strictly controlled conditions. In nature, microorganism can not rely on closely monitored growth conditions and have developed complex regulatory interactions to ensure that a sudden shift in growth conditions does not prove fatal. We are very interested in the study of the mechanisms making the cells robust against variations in the nutrient supply and developed an extension to the deFBA, which takes uncertain availability of nutrients into account. The resulting *robust deFBA* allows us to study how microorganisms adapt in unsteady conditions by shifting the stiff mathematical objective of the regular deFBA in the direction of general survivability.

1.4 Structure of this work

To formulate this new method, we introduce the reader in more detail to the history and the currently used resource allocation models. We do this in Chapter 2. This historical overview shows a direct path from the presented Flux Balance Analysis to the dynamic enzyme-cost Flux Balance Analysis (deFBA), which we will use and enhance for the rest of the work.

In the third chapter we present our work on the creation deFBA models and the introduction of new elements to the formalism. Furthermore, we discuss guidelines on how these models should be formulated to make it more easy to share them via an extension to the Systems Biology Markup Language we devised. We end the chapter with an introduction to the `deFBA-Python` package, which is a main result of our work. It contains tools to handle deFBA models and implementations of the new methods we present during this work. We will end each chapter to highlight the functions of `deFBA-Python` with respect to the presented methods.

Chapter 4 connects the deFBA formalism to possible applications in Model Predictive Control. This is done by the introduction of a receding prediction horizon to the resulting numerical problems. Because the results of deFBA models can vary with the considered time frame, we also device a systematic procedure to minimize the influence of this prediction horizon in the short-term deFBA. This is the first step in making the deFBA viable to be used in a robust optimization environment.

The final results of this work are explained in Chapter 5, which introduces the robust dynamic enzyme-cost FBA which enables us to consider sudden shifts in the nutritional situation in our models. This allows us to emulate possible future conditions and shift the objective in the mathematical model from simply optimizing in a given deterministic future to a more realistic setting in which survivability may be more important.

2 Resource allocation methods

This chapter gives an overview on the history of resource allocation models and its current state. To clarify the evolution and the differences of the individual methods, we follow the introductory chapter of [92] and use a single leading example and present it in the different methods.

2.1 Resource allocation in a self-replicator

In the introduction of constraint based methods the networks were presented as self-optimizing with the goal to maximize certain objectives by attaining an optimal steady state flux distribution. Another view on metabolic networks was established over the last decades postulating that metabolic networks are solving a resource allocation problem. This means the cell decides to funnel available nutrients into different cellular function, like producing biomass (ribosome, enzymes, etc.), storing energy in some form (starch, cellulose, etc.) or doing maintenance (repair of DNA damages, etc.). At the same time the cells must decide how to optimally utilize the available enzymes and how to adapt these to the given environmental situation.

These ideas were introduced to metabolic networks over a very limited time span. One of the first studies to include the limitation of the proteome are [15] and [58]. These works are connecting optimal flux distribution calculated via FBA with the proteome by assuming the cells maintain the minimal enzyme concentration possible to realize this flux distribution. But this constraint is only applied to the metabolic fluxes, while in reality the proteome constrains all cellular functions [75]. This means any growth strategy advised by the cells must be planned around the production of more enzymatic biomass and especially the ribosome as core of the reproductive machinery.

The study [75] is built around a minimal toy model for a self-replicating structure as given in Figure 2.1. This model consists of a single external species $Y = [S_{\text{out}}]$ representing the substrate and two internal metabolites $X = [S_{\text{in}}, G]$; the uptaken substrate S_{in} and processed precursors G. Cellular subsystems are represented via a new species type: the *macromolecules* P, which are addressed by the index set $\mathcal{P}$. We collect in P all species with any enzymatic function, which in this model are the transporters T, enzymes E needed for the processing of the substrate, the machinery L to synthesize membrane lipids M and ribosomes R producing the above and itself. For the synthesis of these components are only the precursors G needed, $P = [E, L, R, T]^T$. The membrane lipids M play a special role in this model as they

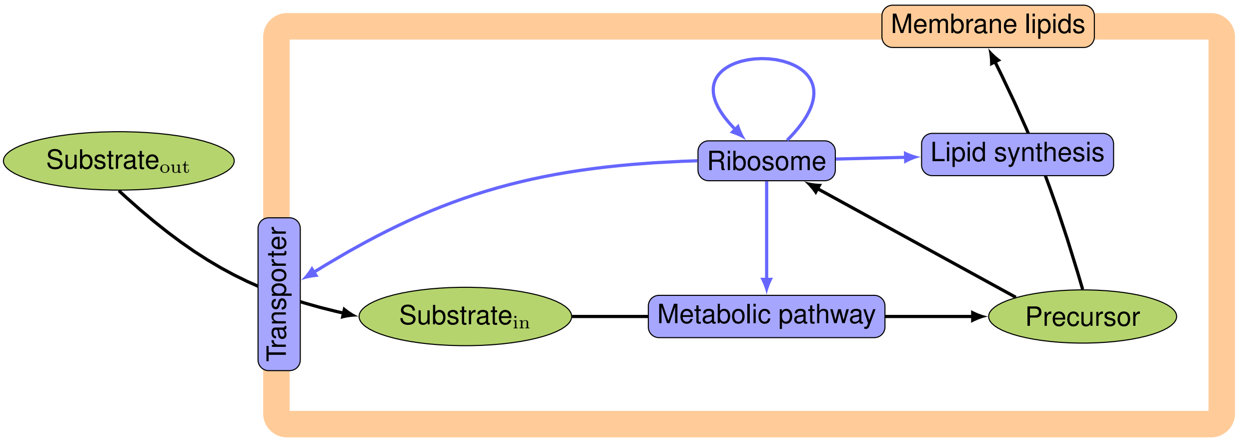


Figure 2.1: A minimal self-replicator as suggested by [75]. Biochemical species are shown in green circles, specialized enzymes are shown in blue boxes. Matter transport is depicted by a black arrow. A blue arrow means synthesis.

Table 2.1: Details on the reactions in the self-replicator model.

Reaction name	Reaction	Catalyzed by	Turnover rate
V_{in}	$S_{\text{out}} \to S_{\text{in}}$	T	7
V_{met}	$S_{\text{in}} \to G$	E	5
V_M	$G \to M$	L	5
V_R	$G \to R$	R	3
V_L	$G \to L$	R	3
V_T	$G \to T$	R	3
V_E	$G \to E$	R	3

do not serve any auto-catalytic purpose. Instead the cell must produce enough to keep its boundary membrane intact during growth. We call this type of macromolecules *quota components* and use the letter $Q = [M]$ to collect them and address them via the index set $\mathcal{Q}$. We collect all the internal species in the vector $Z = [X, Q, P]$.

As the model is formulated in concentrations we use bold fonts for the species to clarify their units. The self-replicator is assumed to operate in *balanced growth* [50]

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t} = \mu \mathbf{Z}., \tag{2.1}$$

This means all components are growing at the same growth rate $\mu \in \mathbb{R}_{\geq 0}$. Or in other words, the composition of the biomass remains constant.

While in the work itself, the growth rate is motivated and defined by the volume change over time, it is unnecessary to go into more detail at this point. For a more rigorous derivation please see the supplements of [75]. It is assumed that the main objective of the cell is to maximize the growth rate $\mu(\mathbf{S}_{\text{out}})$ depending on the concentration of the external substrate.

In the supplements of [75] the numerical analysis is described as a nonlinear optimization problem constructed in the following way. As shown in Figure 2.1 we must regard 7 reactions $V = [V_{\text{in}}, V_{\text{met}}, V_M, V_R, V_L, V_T, V_E]$ in the model. These are given in Table 2.1. While the concentration of the species will change by the difference of production and consumption as explained in Section 1

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{Z}(t) = \mathbf{V}_{\text{production}} - \mathbf{V}_{\text{consumption}}, \tag{2.2}$$

in this dynamic setting we must also consider the dilution due to increase in cell volume $\mathcal{V}$. Given the balanced growth constraint on the system we can explain the increase in volume as

$$\mathcal{V}(t) = \mathcal{V}_0 \exp(\mu t) = \mathcal{V}_0 e^{\mu t}, \tag{2.3}$$

with the initial volume $\mathcal{V}_0$. The dilution due to volume increase is added to the dynamics (2.2) with another consumption term as

$$\begin{align} \frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t} &= \mathbf{V}_{\text{production}} - \mathbf{V}_{\text{consumption}} - \mu\mathbf{Z} \tag{2.4} \\ &= S\mathbf{V}(t) - \mu\mathbf{Z}, \tag{2.5} \end{align}$$

with the stoichiometric matrix as given by Table 2.1. This way we do not need to consider the volume explicitly in the model.

Furthermore, we can express the synthesis rates of E, L, R, and T via the catalytic rate of the total ribosome pool $\mathbf{V}_{\text{rib}}$ via the non-negative weights α_j as

$$\mathbf{V}_j = \alpha_j \mathbf{V}_{\text{rib}}, \;\; \sum_j \alpha_j = 1, \; j \in \{E, L, R, T\}. \tag{2.6}$$

These weights are the main decision factor in this resource allocation problem as they determine the concentrations of the catalytic machinery. To determine the rates in the model the authors use Michaelis-Menten kinetics (cf. Section 1.2.1) with the turnover numbers as given in Table 2.1 as

$$\begin{align} \mathbf{V}_{\text{in}} &= \frac{7 \cdot \mathbf{T} \cdot \mathbf{S}_{\text{out}}}{K + \mathbf{S}_{\text{out}}} \tag{2.7a} \\ \mathbf{V}_{\text{met}} &= \frac{5 \cdot \mathbf{M} \cdot \mathbf{S}_{\text{in}}}{K + \mathbf{S}_{\text{in}}} \tag{2.7b} \\ \mathbf{V}_M &= \frac{5 \cdot \mathbf{L} \cdot \mathbf{G}}{K + \mathbf{G}} \tag{2.7c} \\ \mathbf{V}_{\text{rib}} &= \frac{3 \cdot \mathbf{R} \cdot \mathbf{G}}{K + \mathbf{G}}, \tag{2.7d} \end{align}$$

and the Michaelis constant K chosen as 1 for all reactions.

The membrane lipids M have no direct benefit in the model as producing them can not increase the growth rate μ. To ensure stability of the membrane, we enforce the production of M via a constraint in the form

$$\mathbf{M} \geq \mathbf{T}. \tag{2.8}$$

As we will later see this must be done for all structural macromolecules without catalytic functions in optimization methods.

Finally, we need to limit the maximal concentration of enzymes in the cell as space is limited. In [75] this is done via

$$\mathbf{E} + \mathbf{L} + \mathbf{R} + \mathbf{T} \leq 1. \tag{2.9}$$

Combining the steady-state constraint with these dynamics and the constraints we end with this nonlinear optimization problem

$$\begin{aligned}
&\max_{\mu,\mathbf{Z},\alpha} \mu && && \text{(2.10a)}\\
&\text{s.t. } \mathbf{V}_j = \alpha_j \mathbf{V}_{\text{rib}} && \forall j \in \{E, L, R, T\} && \text{(2.10b)}\\
&\quad S\mathbf{V} - \mu\mathbf{Z} = 0 && && \text{(2.10c)}\\
&\quad \sum_{j\in\{E,L,R,T\}} \alpha_j = 1 && j \in \{E, L, R, T\} && \text{(2.10d)}\\
&\quad \mathbf{V}_{\text{in}} = \frac{7 \cdot \mathbf{T} \cdot \mathbf{S}_{\text{out}}}{K + \mathbf{S}_{\text{out}}} && && \text{(2.10e)}\\
&\quad \mathbf{V}_{\text{met}} = \frac{5 \cdot \mathbf{M} \cdot \mathbf{S}_{\text{in}}}{K + \mathbf{S}_{\text{in}}} && && \text{(2.10f)}\\
&\quad \mathbf{V}_M = \frac{5 \cdot \mathbf{L} \cdot \mathbf{G}}{K + \mathbf{G}} && && \text{(2.10g)}\\
&\quad \mathbf{V}_{\text{rib}} = \frac{3 \cdot \mathbf{R} \cdot \mathbf{G}}{K + \mathbf{G}} && && \text{(2.10h)}\\
&\quad \mathbf{M} \geq \mathbf{T} && && \text{(2.10i)}\\
&\quad \mathbf{E} + \mathbf{L} + \mathbf{R} + \mathbf{T} \leq 1 && && \text{(2.10j)}\\
&\quad \mathbf{Z} \geq 0 && , && \text{(2.10k)}
\end{aligned}$$

with given and fixed external substrate concentration $\mathbf{S}_{\text{out}}$. In the study this is solved using GAMS [24] with the KNITRO solver [17], which guarantees at least locally optimal solutions for the problem. Afterwards a check for global optimality is performed using LINDOGlobal [63]. While using a solver for Non-Linear Problems (NLP) (and especially a global one) for such a small example can be feasible, the size of genome scale models with hundreds of reactions, enzymes and metabolites makes it impossible to use this approach.

Even if the model is small it is very interesting to see that it can replicate some behaviors observed in experiments which are not replicable with FBA. In [75] the model shows that the size of the ribosome pool increases with increasing growth rate while the uptake capacity for nutrients decreases. It is also shown that with small extensions the model is also able to predict overflow metabolism at high glucose concentrations, a growth strategy in which substrates are incompletely oxidized and therefore partly wasted [78]. Especially the reproduction of these kind of growth strategies on multiple substrates as trade-off between cost and benefit of proteome allocation gave rise to a variety of resource allocation model approaches. We will go into detail on the *Resource Balance Analysis* (RBA) [39], *Metabolism and gene Expression models* (ME-Models) [82] and the *dynamic enzyme-cost Flux Balance Analysis* (deFBA).

2.2 Resource Balance Analysis

The work of Moleenar *et al.* [75] sparked an increased interest in resource allocation models and in turn a variety of new modeling methods were published. Following these chronologically the next impactful publication is [39] in which the Resource Balance Analysis was firstly presented. At its core, it kept the idea of the self-replicator as presented in the last section, but reduces the numerical complexity of the problem to be applicable to genome-scale problems.

This was mostly done by three major changes in the approach:

1. Relaxation of steady-state constraint.

   The steady-state constraint as given by (2.10c) is mainly relaxed by neglecting the dilution terms for all non-macromolecules and allowing accumulation (overproduction) of macromolecules and metabolic precursors. This translates to the new form of (2.10c) as

   $$S_{S_{\mathrm{in}},\cdot}\mathbf{V} = 0 \tag{2.11a}$$
   $$S_{P,\cdot}\mathbf{V} \geq 0 \tag{2.11b}$$
   $$S_{\mathcal{Q},\cdot}\mathbf{V} - \mu\mathbf{Z}_{\mathcal{Q}} \geq 0 \tag{2.11c}$$
   $$S_{\mathcal{P},\cdot}\mathbf{V} - \mu\mathbf{Z}_{\mathcal{P}} \geq 0, \tag{2.11d}$$

   with the index sets as explained in Section 1.1. We assume the error due to the relaxation can be neglected as the optimization will keep any accumulation to a minimum. Please note, that μ is an optimization variable and (2.11c), (2.11d) still present a quadratic constraint.

2. Replacing nonlinear kinetics with linear ones.

   Assuming all enzymes are near their respective saturation levels in the cell we can simplify the non-linear kinetics (2.10f-g) with a simple linear approximation

as

$$|\mathbf{V}_{\text{met}}| = k_{\text{cat},E}\mathbf{E},\ |\mathbf{V}_M| = k_{\text{cat},L}\mathbf{L}. \tag{2.12}$$

The absolute value is necessary to handle reversibility of reactions. This equality claim is supported for steady-state growth [39] and other experimental work show that saturation of the enzyme is in most cases given. The study [7] measured absolute metabolite and enzyme concentrations in *Escherichia coli* and compared those to the respective K_M values. The results show that 83% of the measured enzymes are at least 50% saturated with substrate. Additionally, 59% of the substrates have a concentration more than 10-fold higher than the K_M value. Therefore, one can even speak of a trend towards over saturation.

The equality claim is quite strong and may force fluxes to be larger than observed. This problem is addressed in combination with the absolute value in (2.12) via another simplification. We substitute the constraint with two linear constraints handling both possible directions of the reaction with

$$\begin{aligned} \mathbf{V}_{\text{met}} &\leq k_{\text{cat},E}\ \mathbf{E}, \quad -\mathbf{V}_{\text{met}} \leq k_{\text{cat},E}\ \mathbf{E}, \\ \mathbf{V}_M &\leq k_{\text{cat},L}\ \mathbf{L}, \quad -\mathbf{V}_M \leq k_{\text{cat},L}\ \mathbf{L}. \end{aligned} \tag{2.13}$$

This allows for reaction fluxes to be zero even in the presence of the enzyme.

This simplification of the reaction kinetics has an even larger impact on the form of the optimization problem. In the form of [75] all reaction fluxes were given by the kinetics (2.10e-h). Instead the RBA handles them as free variables with enzyme dependent bounds (2.13). The only exception are uptake reactions from the environment to the cells inside as substrate concentrations are usually not given at saturation levels. Hence, for the self-replicator example we keep

$$\mathbf{V}_{\text{in}} = \frac{7 \cdot \mathbf{T} \cdot \mathbf{S}_{\text{out}}}{K + \mathbf{S}_{\text{out}}}. \tag{2.14}$$

With the fluxes as free variables we can also omit the complex allocation of the ribosome capacity via the α_i weights. Instead we can simplify the constraint to

$$\mathbf{V}_L + \mathbf{V}_E + \mathbf{V}_R + \mathbf{V}_T \leq k_{\text{cat,rib}}\mathbf{R} = 3\mathbf{R}. \tag{2.15}$$

This means to handle the ribosome as you would any enzyme capable of catalyzing multiple distinct reactions.

3. Using density constraints instead of volume constraints.

   The work [75] does not model the volume explicitly as it uses concentration variables, the supplements of the paper contain an elaborate approximation of cell volume based on membrane surface. This is then used to derive the membrane-to-transporter ratio, which is 1 for the example (2.10i), and a total concentration limit, which is also 1 (2.10j). The RBA utilizes instead *density constraints* to limit the total concentration of macromolecules to the mean density $\tilde{D}$

$$\rho^T \begin{pmatrix} \mathbf{Z}_{\mathcal{Q}} \\ \mathbf{Z}_{\mathcal{P}} \end{pmatrix} \leq \tilde{D}, \tag{2.16}$$

with the density vector ρ, $[\rho] = 1/\mathrm{M}$ containing the density of the macromolecules. While using this formulation instead of the volume based derivation in [75] does not yield a numeric benefit, it is simpler to generate as the density of each type of molecules can be experimentally measured.

Please note, that this only limits the proteome concentration as the metabolites are still modeled in steady state.

The mathematical formulation of the new quadratic optimization problem is then given as

$$\begin{aligned}
\max_{\mu,\mathbf{Z},\mathbf{V}} \quad & \mu && (2.17\text{a})\\
\text{s.t.} \quad & S_{N_{\text{in}},\cdot}\mathbf{V} = 0 && (2.17\text{b})\\
& S_{P,\cdot}\mathbf{V} \geq 0 && (2.17\text{c})\\
& \begin{pmatrix} S_{\mathcal{Q},\cdot} \\ S_{\mathcal{P},\cdot} \end{pmatrix} - \mu \begin{pmatrix} \mathbf{Z}_{\mathcal{Q}} \\ \mathbf{Z}_{\mathcal{P}} \end{pmatrix} \geq 0 && (2.17\text{d})\\
& \mathbf{V}_{\text{met}} \leq 5\mathbf{E} && (2.17\text{e})\\
& \mathbf{V}_{\text{M}} \leq 5\mathbf{L} && (2.17\text{f})\\
& \mathbf{V}_{\text{in}} = \frac{7 \cdot \mathbf{T} \cdot \mathbf{S}_{\text{out}}}{K + \mathbf{S}_{\text{out}}} && (2.17\text{g})\\
& \mathbf{V}_L + \mathbf{V}_E + \mathbf{V}_R + \mathbf{V}_T \leq 3\mathbf{R} && (2.17\text{h})\\
& \mathbf{M} \geq \mathbf{T} && (2.17\text{i})\\
& \rho^T \begin{pmatrix} \mathbf{Z}_{\mathcal{Q}} \\ \mathbf{Z}_{\mathcal{P}} \end{pmatrix} \leq \tilde{D} && (2.17\text{j})\\
& \mathbf{Z} \geq 0, \ \mathbf{V} \geq 0 && (2.17\text{k})
\end{aligned}$$

where we assumed all reactions can only operate in positive direction and use a fixed substrate concentration $\mathbf{S}_{\text{out}}$.

This problem formulation is only quadratic in the growth rate μ and [39] suggests using a bisection algorithms utilizing half-interval search over μ in combination with a simple feasibility test to determine the solution of (2.17a). This means we test for a fixed value of μ if (2.17a) has a solution. If this is the case we double μ as long as the problem stays feasible. Once the problem becomes infeasible a typical binary search is applied to find the maximal feasible value for μ between the last feasible value and its infeasible double. Without going into detail, in [39] it is proven that RBA problems have a unique maximal solution $\mu^* \geq 0$ and we can find flux distributions $\mathbf{V}$ leading to all values $\mu \in [0, \mu^*]$. Therefore, we do not have to worry about local maxima while using the binary search and can identify μ^* for any initial guess of the growth rate.

Even if we have to check very often for feasibility, this is quite easy for a purely linear problem and is numerically very efficient. Hence, we can use the RBA even for very large problems and especially for genome-scale problems. In [39] the RBA is successfully applied to a model of *Bacillus subtilis* consisting of 358 metabolic reactions, 342 genes, 277 enzymes and 54 transporters.

The results of this analysis were able to reproduce behaviors observed in experiments. For example, the RBA predicts an increase in the growth rate if constraint (2.17i) is relaxed and the cell needs to produce less non-catalytic proteins. This was experimentally proven in [33] by the creation of a knock-out mutant strain of *Bacillus subtilis* deleting part of the chemotaxis machinery not needed in laboratory conditions. The mutant was growing faster than the original wild-type.

2.2.1 ME models

Depending on the objective of the model it may be necessary to add more detail to some processes in the cell, for example, compartmentalization and transport inside the cell, repair processes of DNA, cell division, etc. One prominent model type called *Metabolism and gene Expression models* (ME-Model) [61, 82] focuses on the transcription and translation of genes and cell division. This means the ME-Models explicitly incorporate RNA as mRNA, rRNA, and tRNA and add some coupling-constraints representing production, usage, degradation and dilution of enzymes and RNA. While the resulting model are quite similar to the RBA model type from a perspective of model development they showcase how different objectives shape the modeling methods.

We do not want to discuss the different functions of RNA types here in detail and refer to well known books on this topic like [1]. But for the model of the self-replicator this means we have to include species representing the messenger RNA (mRNA) for all types of enzymes mRNA_L, mRNA_E, mRNA_R, and mRNA_T. Furthermore, we need for each mRNA a set of reactions representing the synthesis of the mRNA $\mathbf{V}_{\text{mRNA}_j,\text{synth}}$, dilution $\mathbf{V}_{\text{mRNA}_j,\text{dil}}$, utilization of mRNA for enzyme translation $\mathbf{V}_{\text{mRNA}_j,\text{trans}}$, and a degradation process $\mathbf{V}_{\text{mRNA}_j,\text{degrad}}$, $j \in \{E, L, R, T\}$ in which the mRNA is deconstructed into its base components which can be reused. The concentration of mRNA limits the possible reaction rate of the synthesis of the enzyme in the same fashion as the ribosome

$$\mathbf{V}_{\text{mRNA}_j,\text{trans}} \leq k_{\text{cat,max,j}} \text{mRNA}_j, \; j \in \{E, L, R, T\}. \tag{2.18}$$

Because the translation reactions for the enzymes are already included in the model, we must add another constraint

$$\mathbf{V}_j = \mathbf{V}_{\mathrm{mRNA}_j,\mathrm{trans}},\ j \in \{E, L, R, T\}. \tag{2.19}$$

For easier reading we omit the dependency on the enzymes for the rest of this section.

More interesting are the coupling constraints to connect these new reactions to one another. The dilution reaction is mostly motivated by a maximum of lifecycles (synthesis - degradation - resynthesis) for each mRNA $a_{\max}$. It is approximated as

$$a_{\max} = \frac{t_{\mathrm{d}}}{\tau_{\mathrm{mRNA}}}, \tag{2.20}$$

with the mean lifetime τ_{mRNA} and the doubling time of the cell t_{d}. The doubling time of the cell can be easily converted into the growth rate via $\mu = \log(2)/t_{\mathrm{d}}$. Following the supplemental material of [61] the coupling constraint

$$\mathbf{V}_{\mathrm{mRNA,dil}} \geq a_{\max} \mathbf{V}_{\mathrm{mRNA,degrad}} \tag{2.21}$$

can then be understood as ”Remove one mRNA from the cell for every $a_{\max}$ times it is degraded”.

Synthesis of the mRNA and the degradation reaction are coupled via another constraint given as

$$\mathbf{V}_{\mathrm{mRNA,degrad}} \geq b_{\max} \mathbf{V}_{\mathrm{mRNA,trans}}, \tag{2.22}$$

with the coupling constant $b_{\max}$. This constant is derived from the maximum translation rate $k_{\mathrm{cat,max}}$, which in term can be derived from the length of the mRNA and the enzyme it is encoding. Here is an example calculation for an approximation of $k_{\mathrm{cat,max}}$ taken from [61]. Assuming the mRNA is 1000 nucleotides long and a ribosomes footprint is 20 nucleotides, we can fit 50 ribosomes on the mRNA. Using the translation rate of 20 amino acids per second [115] for the ribosomes and assume the protein length to be 333 amino acids, we can calculate

$$k_{\mathrm{cat,max}} = \frac{50\ \text{ribosomes}}{\text{mRNA}} \frac{20\ \text{amino acids}}{\text{ribosome sec}} \frac{1\ \text{protein}}{333\ \text{amino acids}} = \frac{3\ \text{proteins}}{\text{sec mRNA}}. \tag{2.23}$$

The authors state that this is just an upper bound and that translation is expected to be much slower. The translation constant is then derived as $b_{\max} = (k_{\mathrm{cat,max}} \tau_{\mathrm{mRNA}})^{-1}$. Thus, the meaning of this constraint translates to ”Degrade one mRNA every $b_{\max}$ times it is translated”.

As the mRNAs are assumed to operate in steady-state, we must introduce one more constraint

$$\mathbf{V}_{\mathrm{mRNA,synth}} = \mathbf{V}_{\mathrm{mRNA}_j,\mathrm{degrad}} + \mathbf{V}_{\mathrm{mRNA}_j,\mathrm{dil}}. \tag{2.24}$$

For the enzymes in the model, we also add a dilution reaction $\mathbf{V}_{\text{dil}}$ which is coupled to the reaction $\mathbf{V}_{\text{usage}}$ the enzyme catalyzes as

$$\mathbf{V}_{\text{dil}} \geq c_{\max} \mathbf{V}_{\text{usage}}. \tag{2.25}$$

The coupling parameter in this case is chosen as

$$c_{\max} = \frac{1}{k_{\text{cat,usage}} t_{\text{d}}}, \tag{2.26}$$

with the turnover number $k_{\text{cat,usage}}$ for the reaction $\mathbf{V}_{\text{usage}}$ and the doubling time t_{d}. The meaning of this constraint can be interpreted as "One enzyme must be removed for every $k_{\text{cat,usage}} t_{\text{d}}$ it is used to catalyze a reaction".

As the enzymes are assumed to operate in steady-state we also have to couple the production of the enzyme to the dilution

$$\mathbf{V}_j = \mathbf{V}_{j,\text{dil}}, \quad j \in \{E, L, R, T\} \tag{2.27}$$

to keep the concentration of enzyme fixed.

Overall, comparing the ME-Models to RBA models, we have to add for each enzyme at least one mRNA species, two reactions, four constraints and two new parameters utilizing the equality constraints to eliminate as many new reactions as possible. While this increases the complexity of the problem quite a lot, the predictive power of the model stays roughly the same. The authors of [82] state ”ME-Models produce experimentally testable predictions for: (1) the cell’s maximum growth rate (μ^*) in the specified environment, (2) substrate uptake/by-product secretion rates at μ^*, (3) metabolic fluxes at μ^*, and (4) gene product expression levels at μ^*.” This is identical to the assertions that can be made with an RBA model. Currently, there exist no direct comparison between the results of an RBA model and an ME-Model. Hence, it is unclear whether the inclusion of mRNA influences the results in terms of growth rate and flux distribution.

2.3 deFBA

The previously presented methods all have in common that they model the network at steady-state and under fixed environmental conditions. While this allows to make strong predictions on metabolic pathway usage and gene expression levels, cells rarely live under fixed laboratory conditions. Instead they are faced with the need to constantly adapt and prepare to changes in the nutritional conditions. To study these adaptation processes we model the environment and the proteome dynamically.

The first methods to model dynamic environments were published in the early 2000s, starting with the *dynamic FBA* [70]. In the so-called *static optimization-based dynamic FBA* an FBA problem for a given environment is used to predict the current metabolic fluxes, uptake rates, etc. These rates are then used to update the bounds and the environment composition by assuming these rates stay constant over small time steps. This process is repeated until a given end-time t_{end} is reached. The results of these methods are metabolic fluxes and uptake rates over the whole time course. But this simple approach suffers from the limitations of the FBA. For example, it has been shown that the large flux variability in FBA calls for the need of lexicographic optimization [93] to guarantee unique solutions. The choice and the order of these objectives can have a very large impact on the solutions [41]. But still the method was successful utilized to model a wide variety of species, e.g. the green microalgae *Dunaliella salina* [34].

But the biggest downside is the missing inclusion of genetic information and the detail level as provided by the RBA. So we opted to base our work on the *dynamic enzyme-cost Flux Balance Analysis* (deFBA), which we present in this section as introduced in [110]. As the rest of this work is based on the deFBA we will go into more detail in comparison to the previous methods. We will not only showcase the notation for the deFBA in terms of the self-replicator example but also present the universal notation. In this section, we will restrain ourselves to reviewing the content of [110], to distinguish our changes in the upcoming chapters.

The first thing comparing the deFBA to its predecessors is that does not model species and reaction rates in concentration based units but instead measures them in molar amounts. We label this by using regular font for these variables instead of the bold font (cf. Supplement B). All variables are now to be understood as mappings from the time to their respective ranges, e.g. metabolic species $t \mapsto X(t) \in \mathbb{R}^{n_x}$ or reaction rates $t \mapsto V(t) \in \mathbb{R}^m$. To differentiate, we address the value at a given time point t always via the mapping $X(t)$ and time slices via $X([t_0, t_1]) = \{X(t) \mid t \in [t_0, t_1]\}$. Usually, the deFBA is defined on a finite time interval $[t_0, t_{\text{end}}]$ with initial time t_0 and the final time $t_{\text{end}} < \infty$. Hence, we write shortly $X(\cdot) = X([t_0, t_{\text{end}}])$.

2.3.1 Species

In the formulation of the deFBA as given by [110], the n species are categorized as either: External species $Y \in \mathbb{R}^{n_y}_{\geq 0}$, metabolic species $X \in \mathbb{R}^{n_x}_{\geq 0}$, or macromolecules $P \in \mathbb{R}^{n_p}_{\geq 0}$. Quota elements as M in the self-replicator model are included in the macromolecule category.

The deFBA allows us to track the growth by observing the molecular mass or simply *biomass* present in the system. To measure this we assign each macromolecule species P_i their *molecular weight* w_i, with a unit of g/mol. The *total biomass* B at time t in the system is then computed as

$$B(t) = w^T P(t), \tag{2.28}$$

with $w \in \mathbb{R}^{n_p}_{\geq 0}$. The biomass corresponds to the dry weight of the cell.

The biomass allows us also to introduce the instantaneous growth rate μ to the deFBA as

$$\mu(t) = \frac{1}{B(t)} \frac{\mathrm{d}B(t)}{\mathrm{d}t}. \tag{2.29}$$

With regards to the self-replicator model the species used in the model are $Y = [S_{\text{out}}]$, $X = [S_{\text{in}}, G]^T$, and $P = [E, M, L, R, T]^T$. Because all macromolecules in this are constructed from the same amount of precursor (cf. Table 2.1), we choose the molecular weight for all species as $w_i = 1\text{g/mol},\ i \in \{E, M, L, R, T\}$.

2.3.2 Reactions

The reactions in the model are classified into three categories as well: *Exchange reactions* $V_y \in \mathbb{R}^{m_y}$ exchanging matter with the outside, *metabolic reactions* $V_y \in \mathbb{R}^{m_y}$ converting metabolic species into one another, reactions $V_p \in \mathbb{R}^{m_p}$ producing biomass from metabolites or deconstructing biomass into them. The reaction vector is constructed as $V = [V_y, V_x, V_p]^T \in \mathbb{R}^m$. A priori, we assume the reactions to be independent variables whose trajectories will be defined by the solution of an optimization problem.

Additionally, direction for the fluxes is defined with biomass independent flux constraints

$$v_{\min} \leq V \leq v_{\max}. \tag{2.30}$$

with $v_{\min} \in \{0, -\infty\}$, $v_{\max} \in \{0, \infty\}$. Information on the direction of the fluxes can be gathered from e.g. thermodynamic constraints [47]. Of course, the constraint (2.30) can also be used to enforce arbitrary independent flux constraints, but in deFBA we usually do not encounter these as the formulation in molar amounts allows for arbitrary large reaction fluxes given enough reactants and enzymes are present. The direction of the fluxes is chosen in a way such that irreversible reactions always transpire into the positive direction with $v_{\min} = 0$.

In the self-replicator model , we use the same reactions as in the RBA version (2.17a) and can categorize them as $V_y = [V_{\text{in}}]$, $V_x = [V_{\text{met}}]$, and $V_p = [V_E, V_M, V_L, V_R, V_T]^T$. All reactions are assumed to operate in positive direction only $V \geq 0$.

2.3.3 Time-scale separation

As we are handling a dynamic problem we must deal with different time-scale on which the processes happen. Following [110] we can distinguish two major time-scales; the metabolite amounts are assumed to change very fast in comparison to the nutrient concentration outside the cell and the changes in macromolecule amounts. This is reflected in three major assumptions.

1. The macromolecules are composed of large amounts of small metabolites. Genome-scale models contain enzymes, ribosome, etc. as macromolecules which consist of hundreds up to thousands of amino acids. This means we can express biomass reactions in a form $V_p : \alpha X \to P$ with a large stoichiometric coefficient $\alpha >> 0$.

2. Production reactions V_p become proportionally slower with increasing α. This assumption holds also true for genome-scale models as the speed of enzyme translation is fixed to roughly 20 amino acids per second. Hence, the production reactions become slower, if the size of the enzymes increases. This property is included by scaling the biomass reaction fluxes with a small factor ε.

3. A very large extracellular volume in comparison to the volume of the cell population $\mathcal{V}_{\text{ext}} >> \mathcal{V}_{\text{int}}$.

The dynamics of the system can then be described as

$$\dot{Y}(t) = S_{\mathcal{Y},y} V_y(t) \tag{2.31a}$$

$$\dot{X}(t) = S_{\mathcal{X},y} V_y(t) + S_{\mathcal{X},x} V_x(t) + \alpha\varepsilon S_{\mathcal{X},p}(t) \tag{2.31b}$$

$$\dot{P}(t) = \varepsilon S_{\mathcal{P},p} V_p(t), \tag{2.31c}$$

with the submatrices of the stoichiometric matrix $S \in \mathbb{R}^{n,m}$ given as described in Section 1.1.

Without going into detail, this model can be transferred to a long time scale $T = \varepsilon t$ resulting into a standard form to apply a singular perturbation method by Tikhonov's theorem [55]. If the theorems' conditions are met, one can approximate the fast dynamics in the metabolites with the quasi-steady assumption $\dot{X} = 0$. It also follows, that the quasi steady-state is locally unique and exponentially stable. Unfortunately, is quite complex to check whether the conditions are met. While the existence of the steady-state can be determined via fixed conditions (cf. [38], [85]), a stability analysis using a kinetic model is also needed, which is usually not available. Hence, we are limited to arguing with biological insight that the quasi steady-state assumption is applicable.

Nevertheless, we will use the *boundary layer condition* in all following models as

$$0 = S_{\mathcal{X},y} V_y(t) + S_{\mathcal{X},x} V_x(t) + \alpha\varepsilon S_{\mathcal{X},p} V_p(t), \ \forall t \geq t_0. \tag{2.32}$$

For the self-replicator model we set $\alpha = \varepsilon = 1$ the dynamics can then be described

as

$$\dot{Z}(t) = \begin{pmatrix} \dot{Y}(t) \\ \dot{P}(t) \end{pmatrix} = \begin{pmatrix} \dot{S}_{\text{in}}(t) \\ \dot{E}(t) \\ \dot{M}(t) \\ \dot{L}(t) \\ \dot{R}(t) \\ \dot{T}(t) \end{pmatrix} = \begin{pmatrix} -V_{\text{in}}(t) \\ V_E(t) \\ V_M(t) \\ V_L(t) \\ V_R(t) \\ V_T(t) \end{pmatrix}, \ t \geq t_0 \tag{2.33}$$

with initial conditions $Z(t_0) = Z_0$. The boundary layer conditions read

$$V_{\text{in}}(t) = V_{\text{met}}(t) \tag{2.34}$$

$$V_{\text{met}}(t) = V_E(t) + V_M(t) + V_L(t) + V_R(t) + V_T(t) + V_M(t), \ \forall t \geq t_0. \tag{2.35}$$

which allows us to effectively delete V_{met} from the model.

2.3.4 Enzyme capacity constraint

The deFBA assumes all enzymes operate at saturation levels meaning the linear approximation used in the RBA (2.13) can be applied as well. But let us formalize this idea in a bit more detail. There exist enzymes capable of catalyzing different reactions. This fact and the inclusion of translational machinery like the ribosome in the deFBA means we can not assume the existance of one-to-one mappings from reactions to enzymes. Instead we use the set

$$\text{cat}(i) = \{j \mid P_i \text{ catalyzes } V_j\} \tag{2.36}$$

to collect all reaction indices for the reactions catalyzed by a specific macromolecule. Please note that this set can be empty for some elements of P if the macromolecule has no catalytic effect in the model, e.g. the quota element Q in the self-replicator model.

The constraint for the reaction rates translates to

$$\sum_{j \in \text{cat}(i)} \left| \frac{V_j(t)}{k_{\text{cat},\pm j}} \right| \leq P_i(t), \forall t \geq t_0. \tag{2.37}$$

Here we distinguish between the forward constant $k_{\text{cat},+j}$ and the reverse one $k_{\text{cat},-j}$.

We assume here that a reaction is only catalyzed by a single macromolecule. Thus, we can write (2.37) in vector form

$$h_{c,i}^T V(t) \leq \mathbb{1}_i^T P_i(t), \ \forall t \geq t_0, \tag{2.38}$$

with the canonical unit vector $\mathbb{1}_i$ having a 1 only at the i-th position and zeros elsewhere. The components of $h_{c,i}$ are given by the inverse turnover numbers

$$(h_{c,i})_j = \begin{cases} \pm k_{\text{cat},\pm j}^{-1} & \text{if } j \in \text{cat}(i) \\ 0 & \text{otherwise.} \end{cases} \tag{2.39}$$

Depending on the reversibility of the reactions catalyzed by P_i we need to account for all possible sign combinations in the k_{cat} values. We collect all combinations in the rows of the matrix $H_{c,i}$ and extend the canonical unit vector to a unit matrix E_i whose rows equal $\mathbb{1}_i^T$.

To clarify here is a minimal example. The enzyme P_1 catalyzes V_1, V_2 reversible with the respective constants $k_{\text{cat},+1}$, $k_{\text{cat},-1}$, $k_{\text{cat},+2}$, $k_{\text{cat},-2}$, while P_2 has no catalytic function. The constraint regulating the enzymatic capacity of P_1 is then written as

$$H_{c,1}V = \begin{pmatrix} \frac{1}{k_{\text{cat},+1}} & \frac{1}{k_{\text{cat},+2}} \\ -\frac{1}{k_{\text{cat},-1}} & \frac{1}{k_{\text{cat},+2}} \\ \frac{1}{k_{\text{cat},+1}} & -\frac{1}{k_{\text{cat},-2}} \\ -\frac{1}{k_{\text{cat},-1}} & -\frac{1}{k_{\text{cat},-2}} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = F_1 P. \tag{2.40}$$

We call this constraint on the network level the *enzyme capacity constraint*

$$H_c V \leq H_f P, \tag{2.41}$$

where H_c and H_f are the vertical concatenations of the matrices $H_{c,i}$ and F_i, $i \in \{1, \dots, n_p\}$.

The enzyme capacity constraint plays a central role in limiting the growth rate of the network as we usually rarely expect to encounter biomass independent flux constraints (2.30).

In the example of the self-replicator we have four macromolecules with catalytic function L, M, T, R as shown in Table 2.1. Using the turnover rates from the table and assuming all reactions can only happen in positive direction the enzyme capacity constraint reads

$$\begin{pmatrix} \frac{1}{7} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{5} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} V_{\text{in}}(t) \\ V_{\text{met}}(t) \\ V_E(t) \\ V_M(t) \\ V_L(t) \\ V_R(t) \\ V_T(t) \end{pmatrix} \leq \begin{pmatrix} T(t) \\ E(t) \\ L(t) \\ R(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} E(t) \\ M(t) \\ L(t) \\ R(t) \\ T(t) \end{pmatrix}. \tag{2.42}$$

2.3.5 Biomass composition constraints

Another important fact to model is the inclusion of known constraints on the composition of the biomass. A natural example would be enforcing the production of non-catalytic macromolecules, which do not increase the auto-catalytic capacity, like cell walls, DNA, etc. We have already seen this in the self-replicator example in form of the membrane lipids.

While the dry weight of the cells typically fluctuate during environmental changes, we quite often find measurements to enforce at least some minimal quota for most macromolecule types. The study [59] for example analyzed the dry composition of the yeast strain *Saccharomyces cerevisiae* and found out that at least 34.5 % of the biomass are made up from proteins. We include these kind of constraints via fractions $\psi_{\text{s}} \in [0, 1)$ of the total biomass B (2.28), e.g. for a structural component P_{s}

$$\psi_{\text{s}} B(t) \leq w_s P_{\text{s}}(t) \tag{2.43}$$

$$\Leftrightarrow \left(w^T - \frac{w_s}{\psi_s} \mathbb{1}_s^T \right) P(t) \leq 0, \qquad \forall t \geq t_0, \tag{2.44}$$

with the molecular weight w_s of P_s and the canonical unit vector $\mathbb{1}_s$ having a 1 at the s-th position and zeros elsewhere.

As with the enzyme capacity we lift this constraint on the network level as

$$H_b P(t) \leq 0, \ \forall t \geq t_0, \tag{2.45}$$

where the rows of H_b correspond to $\psi_s w^T - w_s \mathbb{1}_s^T$. We refer to constraint (2.45) as the *biomass composition constraint.*

In the self-replicator example we encounter only the quota constraint

$$M \geq T, \tag{2.46}$$

which might be useful for a static problem but is not reasonable in the dynamic deFBA setting. As the amount of membrane lipids M directly corresponds to the surface of the cells it is necessary to keep on producing M even if the amount of transporter enzyme T stays constant. Otherwise, we would allow solution curves which are not physically feasible. The need to formulate biomass composition constraints as given by (2.45) is based on the lack of volume or density constraints in the deFBA.

For the self-replicator model we change (2.46) to a need that 20% of the total biomass must consist of membrane lipids

$$0.2B(t) \leq M(t) \Leftrightarrow E(t) - 4M(t) + L(t) + R(t) + T(t) \leq 0, \tag{2.47}$$

with $H_b = (1, -4, 1, 1, 1)$. Of course, it is possible to keep constraints like (2.46) additionally, if such close connections between individual biomass components are known.

2.3.6 Objective functional

To define a complete linear optimization problem we are still missing an objective functional in the form

$$\max_{V(t),P(t)} J(Y(t), P(t), V(t)), \ \forall t \in [t_0, t_{\text{end}}]. \tag{2.48}$$

The original work [110] compares three functionals and its effects on the flux variability and the biophysical meaning of the solution via a minimal example taken from [76]:

1. Maximize biomass amount at end-time t_{end}

$$J_1 = w^T P(t_{\text{end}}) = B(t_{\text{end}}). \tag{2.49}$$

2. Discounted maximization of biomass accumulation

$$J_2 = \int_{t_0}^{t_{\text{end}}} B(t) \exp(-\varphi t) \ \mathrm{d}t, \tag{2.50}$$

with the discount parameter $\varphi \in \mathbb{R}_{\geq 0}$.

3. Minimize time to metabolize all nutrients

$$J_3 = -\int_{t_0}^{t_{\text{end}}} \mathrm{d}t = -t_{\text{end}}, \ Y(t_{\text{end}}) = 0. \tag{2.51}$$

The first functional J_1 is commonly used in the dynamic FBA with an dynamic optimization approach. Unfortunately, the results show that using this objective results in non-unique solutions and therefore a quite high flux variability as proven by a *Flux*

Variability Analysis [16]. Because we want a method which can reliable be used to reproduce results, we refrain from using this type of objective for the deFBA.

The second objective J_2 in comparison has zero flux variability for the presented Monod example and delivers results identical to predictions from a kinetic model taken from literature. The interpretation of this objective is also nearly identical to the growth rate maximization we have seen in the RBA and ME-models as the discounted integral favors solutions growing as fast as possible; especially in the beginning of the predicted time. At the same time the dynamic formulation allows for solutions which do not optimize the growth rate at any given time. For example, the deFBA allows solution which start slowly, e.g. an adaption phase due to changed environmental conditions, but pay out after some time. The newly introduced discount factor φ plays a large role in the shaping the optimal solution. Choosing larger values for φ favors the early stage of the optimization further and maybe detrimental to long-term effects. Hence, we set without loss of generality the value to zero as we miss a reliable method to choose it.

Lastly, the solutions using J_3 are identical in quality to ones from J_2. We can still not use this objective in general. For example, it is quite obvious that the end-time condition $Y(t_{\text{end}}) = 0$ is not applicable to any problem as we can not restrict ourselves to batch processes in which the nutrients run out. Furthermore, it might even be hard to identify which external species are even supposed to be nutrients. E.g. yeast produces alcohol during fermentation, thus in most cases we would see alcohol as a side-product and not a carbon source for the yeast. But experiments have shown that ethanol can and will be metabolized in the yeast under the right conditions (temperature, pH-value, ethanol concentration, etc.) [111]. This would make it necessary to invest a lot of knowledge *a priori* into the objective functional and we want to keep the method as accessible as possible.

Hence, we will only use objective $J = J_2$ without the discount for the rest of this work.

2.3.7 Optimization problem

We can now formulate the full dynamic enzyme-cost Flux Balance Analysis as a dynamic linear optimization problem

$$\max_{V(t),P(t)} \int_{t_0}^{t_{\text{end}}} B(t)\ \mathrm{d}t \tag{2.52a}$$

$$\text{s.t. } Z(t_0) = Z_0 \tag{2.52b}$$

$$\dot{Y}(t) = S_{\mathcal{Y},y} V_y(t) \tag{2.52c}$$

$$\dot{P}(t) = \varepsilon S_{\mathcal{P},p} V_p(t) \tag{2.52d}$$

$$S_{\mathcal{X},y} V_y(t) + S_{\mathcal{X},x} V_x(t) + \alpha\varepsilon S_{\mathcal{X},p} V_p(t) = 0 \tag{2.52e}$$

$$H_b P(t) \leq 0 \tag{2.52f}$$

$$H_c V(t) \leq H_f P(t) \tag{2.52g}$$

$$v_{\min} \leq V(t) \leq v_{\max} \tag{2.52h}$$

$$Z(t) \geq 0,\ \forall t \in [t_0, t_{\text{end}}], \tag{2.52i}$$

with short-notation $Z = (Y^T, P^T)^T$ and the given initial values Z_0.

The optimization problem (2.52a) can be solved by a multitude of methods [107], but we stick for this work to a collation method based on a time discretization of the dynamic variables $Y(t), P(t)$, and $V(t)$ [10], [108], [89]. In Section 4.5 we will present the discretization in a bit more detail and showcase a collocation based on the implicit midpoint rule [92] for quadrature in Section 5.5.

2.3.8 Conditional FBA

Like the RBA with ME models, there also methods around which are very closely related to the deFBA. A parallel development to the deFBA is the *conditional FBA* (cFBA) [95], which was developed with specifically phototropic organisms in mind, e.g., cyanobacteria capable of photosynthesis and transforming sunlight and CO_2 into organic carbon. Their metabolism follows hence a strong diurnal lifestyle in which energy storage, usually starch, is produced during daylight and consumed during the night.

While being mostly identical in formulation to the deFBA in formulation and constraints, the cFBA adds a periodic constraint to reflect on the diurnal strategies of the cyanobacteria

$$P(t_{\text{end}}) = \beta P(t_0), \tag{2.53}$$

with $\beta > 1$. This can be understood as balanced growth on the interval $t_{\text{end}} - t_0$. For an interval length of 24 hours this constraint reflects that the bacteria may not maximize their growth rate at all times but instead maximize their growth over a complete day-night cycle. Mathematically this translates to an objective functional maximizing β, meaning gain as much possible biomass during the given time. Of course, it is necessary to make the initial biomass composition $P(t_0)$ a free variable when formulating the optimization problem.

The work [90] uses the cFBA to evaluate a genome-scale model of the cyanobacteria *Synechococcus elongatus* PCC 7942 and adds a new form of constraints to the cFBA formulation: Maintenance. While deFBA and cFBA contain the possibility to enforce the production of non-catalytic biomass in form of the biomass composition constraint, this does not contain the necessary prolonged energy expenditures to recycle these components constantly. In this work, the maintenance is modeled by adding a flux hydrolyzing ATP at a rate of 0.13 mmol per gram dry weight.

Lastly, we have to state that due to the constraint (2.53) the cFBA becomes a quadratically constrained program, but is linear for any fixed β. The most efficient way to solve this kind of problem is using a binary search in β comparable to RBA. As a cFBA problem is multiple times the size of a corresponding RBA problem, the computation time is also hugely increased in comparison.

2.4 Conclusion

In this chapter we presented the most common resource allocation models to introduce the user into this model class and give an overview on their development. The static methods like RBA and ME-models are very interesting for the analysis of cells growing under fixed environmental conditions. They can predict the necessary gene expressions to survive under the given conditions and the achievable growth rate. From an engineering perspective this can also be inverted and used e.g. to determine optimal feeding strategies to maximize certain byproducts [34].

Nevertheless, these methods are incapable to generate data under a dynamic environment. For example a lot bioproduction processes are run as batch-processes as the biomass must be harvested in the end. It can be quite cumbersome to predict the results or choose an ideal runtime with the static methods. The deFBA on the other hand fits ideally into these kinds of situations as the dwindling resource availability due to the uptake of the cells is a core feature.

The deFBA has been successfully used to predict an optimal switching time in a two-step process and improved on the yield and quality of the desired product, see [51].

From a more research motivated perspective the deFBA can substitute expensive experiments or be used for experimental design applications. For example, multiple groups are currently working on methods to generate regulatory networks from snapshot data, e.g. [83], [20]. But looking at these studies they always assume the data originating in error-prone measurements from either single-cells or whole populations.

We believe it would be very beneficial to generate deFBA models for the different organisms instead and utilize these to speed up the process and make it more efficient.

The deFBA as presented in [110] can be further improved on to make it viable in more applications. We will also see that the deFBA can produce biological infeasible predictions, due to inherent issues with the optimization approach. We will address these over the course of this work and do our best to improve the deFBA.

As a first step we implement a complete deFBA workflow starting with generating the model itself, formulating a way to share the model, and finally introducing a closed software environment for the evaluation and simulation.

3 Building, exchanging and evaluating deFBA models

This chapter is based on the joined publication [91] with Alexandra Reimers within the European project ROBUSTYEAST. The approach on building deFBA models from FBA models was refined by A. Reimers. H. Lindhorst was in charge for the SBML extension *Resource Allocation Modeling* (RAM) which was also published individually as an SOP [65]. The `deFBA-Python` package was developed solely by H. Lindhorst, based on the collocation class `LinOpt` by Steffen Waldherr.

This chapter presents the reader with a full pipeline from the creation to the utilization of deFBA models. We start with some extensions to the deFBA standard presented in Section 2.3. Using deFBA becomes simpler for modelers if we add components which are often used in other model setting. A good example is adding maintenance reactions as used in the cFBA (cf. Section 2.3.8).

Afterwards, we present a systematic way to encode a deFBA model in the *Systems Biology Markup Language*. The Section 3.3 discusses how we can extend existing FBA models to deFBA models. We close this chapter with an introduction to the `deFBA-Python` toolbox and its core functions.

3.1 Extending the deFBA formalism

The deFBA formalism as presented in (2.52a) is very general and can be applied in almost every situation. But the open presentation might make it difficult for modelers to utilize it to its full extent. We decided therefore, to present in this section slight modification to the formalism to make it easier to understand the functions of some components and enable new constraint types. Furthermore, these changes make it easier to prepare models for the export to SBML as we will discuss in Section 3.2.

3.1.1 Refining the macromolecules

We want to further classify the types of macromolecules we can handle with the deFBA to make it easier for modelers to map desired functionality to a species in the model.

We already mentioned storage species during the introduction to the cFBA. While storage plays a secondary role for laboratory species living under constant and adequate nutrient supply, we want to model species in highly dynamic environments, which might also include starvation periods. We enable the cells to prepare for these occasions by creating an energy storage, e.g., in the form of starch. We identify storage species in the model as $C \in \mathbb{R}^{n_c}_{\geq 0}$, $n_c \leq n_p$ and consider them as part of the macromolecules P. We assign the storage a new variable as they are clearly different to the other types of macromolecules we have introduced so far. Because these storage components can accumulate in the model, they must be included into the total biomass B (cf. (2.28)). Hence, we assign each storage species their molecular weights $w_{\mathcal{C}}$ as with the other macromolecules.

In connection to the storage species we also introduce a new class of reaction used to synthesize the energy storage or to deplete them. These *storage reactions* are addressed as $V_{\mathcal{C}} \in \mathbb{R}^{m_c}$ and are regarded as a subclass of the macromolecules producing reactions $V_{\mathcal{P}}$.

Furthermore, we have already seen *quota components* with no catalytic value, c.f. membrane lipids in Section 2.1. Because their fundamental difference to the enzymatic part of the biomass and the storage, we address them with the variable $Q \in \mathbb{R}^{n_q}_{\geq 0}, n_q \leq n_p$. As with storage they are a subclass of all macromolecules and we assign them their individual molecular weights via $w_{\mathcal{Q}} \geq 0$ to include them in the total biomass.

The reactions producing quota components are called *quota reactions* $V_q \in \mathbb{R}^{m_c}$ and are a subclass of the macromolecule producing reactions V_p. We will look at the effects on the dynamics of the model once we introduced the other extensions to the deFBA.

All macromolecules which have a enzymatic function are collected under the *enzymatic molecules* identifier $E \in \mathbb{R}^{n_e}$ with molecular weights w_e and the respective production reactions are identified as $V_e \in \mathbb{R}^{m_e}$.

Therefore, we can express the respective identifiers as

$$P = \begin{pmatrix} C \\ Q \\ E \end{pmatrix}, \; V_p = \begin{pmatrix} V_c \\ V_q \\ V_e \end{pmatrix}, \text{ and } w = \begin{pmatrix} w_c \\ w_q \\ w_e \end{pmatrix}. \tag{3.1}$$

It is quite obvious that this increases the complexity of the notation a lot. Hence, we stick to using P for all macromolecules and only differentiate the subclasses if necessary.

Using these subclasses can become an issue as the attributes of each subclass are not mutually exclusive. So it is possible to enforce a quota constraint on an enzyme or a storage species. The hierarchy of the attributes will be clarified in Section 3.2.

3.1.2 Maintenance reactions

Maintenance reactions are necessary energy expenditures from the cell, which are not modeled directly in enzyme or quota production reactions. A typical example for maintenance is the upkeep of ion pumps to create the necessary membrane potential. We address maintenance reactions as $V_a \in \mathbb{R}^{m_a}$ and assume that their rates scale with the total biomass. Because we want to keep the complexity of these reactions simple, we assume they are in the simple form

$$V_a : \text{ ATP } \rightharpoonup \text{ADP} + \text{Pi}. \tag{3.2}$$

Depending on the resolution of the model it is also possible to include other metabolites, e.g. amino acids, in the maintenance reaction, but this might violate mass balance constraints. Maintenance reactions should operate only on the metabolites level. Therefore, we count them to the metabolic reactions $V_a \subset V_x$. Unless important we will include the maintenance reactions in V_x without explicitly mentioning them.

To enforce certain rates on the maintenance reactions scaling with biomass we introduce a new constraint

$$\phi_i B(t) \leq V_{a,i}(t), \ \forall t \geq t_0, \tag{3.3}$$

with the *maintenance coefficient* $\phi_i \in \mathbb{R}_{\geq 0}$. On the network level we use the matrix form

$$H_a P(t) \leq H_g V(t), \tag{3.4}$$

with the rows of H_a corresponding to $\phi_i w^T$ and the rows of H_g are $\mathbb{1}_i$. Please note, that we do not limit the constraint to the maintenance reactions V_a to enable the modeler to enforce biomass related flux constraints on *any* flux. At the same time we warn the reader that this might quickly lead to infeasible optimization problems and should only be used with a certain awareness.

3.1.3 Extended dynamics and lumped reactions

With three new subclasses of the macromolecules and three additional reaction types the system dynamics become quite complex. Additionally, we have to consider the scaling factor α representing the mean macromolecules length. The purpose of scaling the biomass producing reactions was done in [110] to be able to proof the meaningfulness of the time-scale separation. From a numerical perspective it is also very useful because it equalizes the magnitude of fluxes in the model. But α was introduced as a single scalar value and it would be more useful to scale each biomass reaction independently as the size of the biomass components P can vary a lot. We do not go into more detail into optimal scaling for the stoichiometric matrix and biomass production fluxes as we usually leave the scaling to our numerical tools. Hence, from here on we set the scaling to $\alpha = 1$.

This way the extended dynamics for the deFBA read

$$\dot{Y}(t) = S_{\mathcal{Y},y}V_y(t) \tag{3.5a}$$

$$\dot{P}(t) = \begin{pmatrix} \dot{C}(t) \\ \dot{Q}(t) \\ \dot{P}(t) \end{pmatrix} = \begin{pmatrix} S_{\mathcal{C},c} & S_{\mathcal{C},q} & S_{\mathcal{C},e} \\ S_{\mathcal{Q},c} & S_{\mathcal{Q},q} & S_{\mathcal{Q},e} \\ S_{\mathcal{E},c} & S_{\mathcal{E},q} & S_{\mathcal{E},e} \end{pmatrix} \begin{pmatrix} V_c(t) \\ V_q(t) \\ V_e(t) \end{pmatrix} = S_{\mathcal{P},p}V_p(t),\ \forall t \geq t_0 \tag{3.5b}$$

and the boundary layer condition reads

$$S_{\mathcal{X},y}V_y(t) + S_{\mathcal{X},x}V_x(t) + S_{\mathcal{X},c}V_c(t) + S_{\mathcal{X},q}V_q(t) + S_{\mathcal{E},e}V_c(t) = 0. \tag{3.6}$$

In an ideal model all of the new submatrices of $S_{\mathcal{P},p}$ except the ones on the main diagonal are equal to zero as we assume no direct conversion from one biomass type to another

$$S_{\mathcal{P},p} = \begin{pmatrix} S_{\mathcal{C},c} & 0 & 0 \\ 0 & S_{\mathcal{Q},q} & 0 \\ 0 & 0 & S_{\mathcal{E},e} \end{pmatrix}. \tag{3.7}$$

But this shows a crucial problem with the stoichiometry as shown so far. The work [110] assumes implicitly that every in-between step of a reaction is modeled. For example we can look at a non-branching pathway from nutrient to biomass in the form

$$\mathrm{Y} \overset{E_1}{\to} \mathrm{Y}_{\mathrm{in}} \overset{E_2}{\to} \mathrm{X} \overset{E_3}{\to} \mathrm{P}\ , \tag{3.8}$$

with nutrients Y, internal nutrient Y_{in}, precursor X and biomass P. The reactions are catalyzed by the respective enzymes E_i and the catalytic constants $k_{\mathrm{cat},i}$, $i \in \{1, 2, 3\}$.

We can lump this reaction to a single reaction

$$\mathrm{Y} \overset{E_s}{\to} \mathrm{P}\ , \tag{3.9}$$

catalyzed by the ”super”-enzyme E_s consisting of all three previous enzymes.

$$E_s = \sum_{i\in\{1,2,3\}} k_{\mathrm{cat},i}E_i, \tag{3.10}$$

with the new catalytic constant $k_{\mathrm{cat},s} = 1$. This kind of lumping can only be used to non-branching pathways in which all in-between states are metabolites. Another example would be the already mentioned direct conversion of biomass types in which we do not model the dismantling process. To include this reaction in the very specific dynamics given (3.5a) we need to add another matrix $S_{\mathcal{Y},p}$ and $V_{\mathcal{P}}$.

Instead of using column specific slices of the stoichiometric matrix in all places we simplify (3.5) and (3.6) to

$$\dot{Y}(t) = S_{\mathcal{Y}} V(t) \tag{3.11a}$$

$$\dot{P}(t) = \begin{pmatrix} \dot{C}(t) \\ \dot{Q}(t) \\ \dot{P}(t) \end{pmatrix} = \begin{pmatrix} S_{\mathcal{C}} \\ S_{\mathcal{Q}} \\ S_{\mathcal{E}} \end{pmatrix} V(t) = S_{\mathcal{P}} V(t) \tag{3.11b}$$

$$0 = S_{\mathcal{X}} V(t), \qquad \forall t \geq t_0, \tag{3.11c}$$

with the short-notation $S_i = S_{i,\cdot}$ for all submatrices.

At the moment it is still unclear how to assign the lumped reactions to a reaction class as they might produce external species and biomass components at the same time. But using the row-based notation as presented in (3.11) makes this quite irrelevant. As we will see later in Section 3.2 reaction types are even unnecessary to encode deFBA models in SBML.

3.1.4 Objective biomass

So far we have identified the objective functional with the integral of the total biomass B over time. But studies as [110], [109] show that including biomass species without catalytic capabilities into the objective functional can lead to artificial solutions in which the whole network focuses on the production of quota/storage components. This can often be explained by a higher biomass yield of the quota components and especially storage species in comparison to costly enzymes. From a biological perspective an increased production of these components is usually useless; unless storage is created with a starvation period in mind.

Hence, we avoid these problems preemptively by introducing a new concept: *Objective biomass*

$$B_{\mathrm{o}}(t) = w_{\mathrm{o}}^T P(t) = w_{\mathcal{C},\mathrm{o}}^T C(t) + w_{\mathcal{Q},\mathrm{o}}^T Q(t) + w_{\mathcal{E},\mathrm{o}}^T E(t), \tag{3.12}$$

with the objective weights w_o, $[w_o] = \mathrm{g/mol}$. The objective weights are identical to the molecular weights for the catalytic part $w_{\mathcal{E},o} = w_{\mathcal{E}}$, but we recommend to set these to zero for storage $w_{\mathcal{C}} = 0$ and quota $w_{\mathcal{Q}} = 0$. This ensures that these two components are handled as just means to enable growth but slowing it if too much of these components are produced. The new objective functional for the deFBA is then given as

$$J = \int_{t_0}^{t_{\mathrm{end}}} B_{\mathrm{o}}(t)\ \mathrm{d}t. \tag{3.13}$$

Please note, that we only change the weights for the objective biomass. The total biomass B is still used for constraints scaling with biomass, such as maintenance reactions (3.4) or the biomass composition constraint (2.45).

3.1.5 Optimization problem

With the extended notation, we can redefine the formulation of a deFBA problem as

$$\max_{V(t),P(t)} \int_{t_0}^{t_\text{end}} B_\text{o}(t)\ \text{d}t \tag{3.14a}$$

$$\text{s.t. } Z(t_0) = Z_0 \tag{3.14b}$$

$$\dot{Z}(t) = \begin{pmatrix} S_\mathcal{Y} \\ S_\mathcal{P} \end{pmatrix} V(t) \tag{3.14c}$$

$$S_\mathcal{X} V(t) = 0 \tag{3.14d}$$

$$H_a Z(t) \leq H_g V(t) \tag{3.14e}$$

$$H_b Z(t) \leq 0 \tag{3.14f}$$

$$H_c V(t) \leq H_f Z(t) \tag{3.14g}$$

$$v_\text{min} \leq V(t) \leq v_\text{max} \tag{3.14h}$$

$$Z(t) \geq 0,\ \forall t \in [t_0, t_\text{end}], \tag{3.14i}$$

where we also adapted the constraint matrices H_a, H_b, H_c to the new state vector

$$Z = (Y^T, P^T)^T = (Y^T, C^T, Q^T, E^T)^T. \tag{3.15}$$

Any further reference to the continuous deFBA problem refers to (3.14a).

3.2 SBML extension - Resource Allocation Modeling

Finding a suitable way to digitally save a deFBA model proofed to be quite a problem as we are faced with a multitude of specialized data formats, software and interfaces between those. We chose to utilize the *Systems Biology Markup Language* (SBML) [48] as basis for the exchange. It is a representation format, based on *XML* [13], for storing computational models of biological processes.

We chose SBML for its easy accessibility and extendability. There exist a software library called `libSBML` [12], which is available for the most prominent programming languages, e.g. `C++`, `Python`, `MATLAB`. This makes it quite easy to design import/export interfaces from and to SBML in the respective language. Furthermore, a multitude of extensions are already available for SBML. The most relevant one is the *Flux Balance Constraint* (FBC) extension [84], which enables SBML to include information on which enzyme is catalyzing which reactions.

Because the community using SBML focuses mostly on classical FBA models, the language is currently lacking possibilities to encode all information necessary for the construction of deFBA models. While some information might also be encoded using existing extensions like the `groups`-package [49], these were often designed with another usage in mind. So to keep our solution focused and compatible with existing SBML formats, we choose to instead design our own extension called *Resource Allocation Modeling* (RAM) [91], [65].

In this section we explain RAM in every detail and highlight how the chosen format supports the creation and debugging of deFBA models. To be inline with documentation documents for SBML we use bold font with a capital letter for XML nodes (resp. elements) and bold font with small letters for attributes of the elements, e.g. **Species** is a node representing a species in the network and one of its attributes is **name** containing additional information on the species' chemical name.

The additional data encoded via RAM are placed inside **Annotation** nodes. Each and every element can be annotated so we can add on any component. To make sure the annotations can still be used for other information we create a new namespace `RAM` as

```
<ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
```

with the URL of the specification documents included.

For a full example of all functions of RAM, please see the *resalloc* model in Section A.1.

3.2.1 SBML header

The header is very important for SBML files as it declares which SBML level and its respective version is used. The current one is SBML Level 3 Version 2 Release 1 (L3V2) and its documentation is available online at [1]. The RAM extension was designed to work with L3V2, but due to the minimal changes for the components we are using should be working with Level 3 Version 1 as well. Furthermore, the header also clarifies which extensions are used in the file.

The typical header for RAM looks like this:

```
<sbml xmlns="http://www.sbml.org/sbml/level3/version2/core"
  level="3" version="2" fbc:required="false"
  xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version2">
```

Please note, that the FBC package was not yet novelized for Level 3 Version 2.

3.2.2 The model

An SBML file contains one or more **Model** elements containing all data to construct the respective model. Any data outside of the **Model**-declaration can not be included

---

[1] `http://sbml.org/Documents/Specifications`

while reading it in `libSBML`. This ensures that models can not cross define to one another.

We recommend putting only a single model in each SBML file. The only attributes that must be added to the **Model** is a unique **id** and the **fbc:strict** flag, e.g.

```
<model id="example" fbc:strict="false">.
```

3.2.3 Compartmentalization

Usually, the first entry in a **Model** is the **listOfCompartments**, including all **Compartments** of the model. Compartments represent a bounded space in which the species are located. In SBML they do not actually have to correspond to actual structures in or outside the cells, but it is recommended to use them this way. We have to stress at this point that compartments are only to be used for physical location and not to map model related functions to the species.

Because the deFBA does not contain explicit modeling of compartments in SBML, we do not enforce the usage of compartments. Later we will see that due to a required attribute in the species, we must include at least one ”default” compartment. But we highly recommend to use compartments as they can help in verification and debugging of models. Take for example ATP, which is produced in the mitochondria but used throughout the cell. Without modeling the compartmentalization the model would only include a single ATP species and thus neglecting the need for transport processes. It is surely possible to create an ATP species for each compartment without using the **Compartments** provided by SBML but these would only differ in their **id**, which might be misleading. Later on, we will also suggest some conventions for choosing ids based on compartments. This will make it easier to check for correctness, e.g., if all species involved in a reaction are in the same compartment.

For examples using **Compartments** see Supplements A. For more detail on **Compartments** please see the SBML documentation [49].

3.2.4 Parameters

The **listOfParameters** contains the **Parameters**, which are used to define a symbol associated with a value. They can be used at any point in the model in place of a numerical value. Additionally, they can be used in mathematical formulas in the model, e.g., to define kinetic rate laws.

Parameters need a unique[2] **id** and must specify whether the **Parameter** is **constant**. If the **constant** attribute is set to ’false’ nothing in the SBML file except **InitialAssignment** can change the **value** attribute. Interestingly, the **value** of a

---

[2] When talking about unique **ids** we always mean unique in their respective type. For example there may exist a **Parameter** and a **Species** with identical **ids**. SBML also makes use of **metaids**, which must be unique among all model components, but we do not use these in RAM.

Parameter needs not to be set, but we strongly suggest to define all parameters in the SBML file. In SBML L3V2 parameters can now also be used for logic statements and boolean **values** can be used. Additionally, one can include a **unit** attribute here pointing to a **Unit** element. It might also be helpful to add an **sbo** term [25] to clarify the meaning of the parameter inside the model.

The deFBA uses four major parameter types: Forward catalytic constants $k_{\text{cat},+}$, reverse catalytic constants $k_{\text{cat},-}$, molecular weights w and objective weights w_o. While it is possible to give the numerical value for each of these constants at their respective position, we decided that in RAM all these parameters must be given via a **Parameter** element using the **value** attribute. This way, we can distinguish parameters even if they share the same numerical value and we can easily change their **values**. We also introduce a `zero` Parameter, which we will use to ensure value correctness in certain places and to avoid division-by-zero errors. Inside `zero`, we set the **value** to zero and use this more as another chance to check whether the user forgot to set something or put it to zero on purpose.

3.2.5 Species

Another list inside the model is the **listOfSpecies**. Every **Species** element in it contains at least these five required attributes.

- **id** (string) Unique identifier for the species. We suggest to add the **id** of the **Compartment** to avoid duplicate **ids** We elaborate on this problem later in this section.
- **compartment** (string) An inconsistency of SBML, while **Compartments** themselves must not be included in the model, this is a strictly required attribute for each species and the value must be a valid **id** of a **Compartment** in the **listOfCompartments**.
- **constant** (boolean) Specifies whether the species is regarded as a ”fixed” species. If this is set to ’true’ the amount of the species can not be changed after initial assignment. Limiting external species and biomass components must have this set to ’false’.
- **boundaryCondition** (boolean) Specifies, whether a species is at the systems boundary. The amounts of a boundary species can not be changed by any reaction, but by rules, e.g. **AssignmentRule**, **RateRule**. Non-limiting external species, marked with **boundaryCondition**=’true’ and **constant**=’true’ are considered to be available in unlimited amounts. Hence, they can be deleted from the optimization. As example consider oxygen as non-limiting external species. Then we substitute the corresponding uptake reaction $V_O : O_2 \to O_{2,\text{internal}}$ with $V_O : \emptyset \to O_{2,\text{internal}}$.

- **hasOnlySubstanceUnits** (boolean) This flag indicate whether the species is modeled in molar amounts or concentrations.

Additionally, a **Species** element can have the relevant optional attributes.

- **name** (string) Additional information about the species; usually the scientific name.
- **initialAmount** (double) If the model provides initial values, these can be set here. Fixed species should not be given a value. To ensure correctness of the values, this attribute must be set for all external species and biomass species. Otherwise, none of the given values are imported into the deFBA model.

While these attributes, and other optional ones we did not mention, suffice for the construction of an FBA model, we need to construct some new attributes. We save these in a RAM node as explained earlier. An example is provided at the end of this section.

Inside the RAM node, we place a **ram:species** element, with the following attributes

- **ram:speciesType** (string) This attribute does not point to **Parameter** but can only be chosen from a list of species types. In the current version 1.0.1, we distinguish between 'extracellular', 'metabolite', 'enzyme', 'storage', and 'quota'. They represent the classes of species as explained in Section 2.3.1 and Section 3.1.1. This attribute is used to determine whether a species is modeled dynamically or in quasi steady-state. Furthermore, we use this attribute to check if all necessary attributes are given for the species type.
- **ram:molecularWeight** (string) This entry points to the **id** of a **Parameter** containing the molecular weight w of the species. This attribute must only be given for macromolecules P, meaning species with **ram:speciesType** 'enzyme', 'storage', or 'quota'. While we allow this element to be zero, a warning is issued to the user in this case.
- **ram:objectiveWeight** (string) Containing the **id** of a **Parameter**. The respective values determines how the species enters the objective biomass as w_o, see Section 3.1.4. For enzymatic species this should be identical to the molecular weight w defined in the attribute **ram:molecularWeight**. If a macromolecule species is not to be included in the objective set this to the `zero` parameter.
- **ram:biomassPercentage** (string) Points to the **id** of a **Parameter** corresponding to the biomass percentage $\psi \in (0,]$ used for the construction of the biomass composition constraint (2.45). A biomass composition constraints must be active for quota species Q and can be for other macromolecules. To ensure correctness of the entry, we recommend setting this to the `zero` parameter if no constraint should be active for this species.

As the handling of the RAM attributes can be a bit confusing, we clarify the optionality of attributes based on the **ram:speciesType** in Table 3.1.

Table 3.1: Defining possible values ram attributes for different species types. The symbol for empty set ∅ means the value can left out, a plus + means only positive values are allowed, and the slash / represents an or relation.

	extracellular	metabolite	storage	quota	enzyme
molecularWeight	∅/`zero`	∅/`zero`	+	+	+
objectiveWeight	∅/`zero`	∅/`zero`	+/`zero`	+/`zero`	+
biomassPercentage	∅/`zero`	∅/`zero`	+/`zero`	+	+/`zero`

An example for a species is taken from the *resalloc* model, which can be found in the appendix A.1.

```
<species id="Emetab1" name="Generic metabolic enzyme" compartment="bio"
 initialAmount="1.1" constant="false" hasOnlySubstanceUnits="true"
 boundaryCondition="false">
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:species ram:speciesType="enzyme ram:biomassPercentage="zero"
     ram:objectiveWeight="weight_12" ram:molecularWeight="weight_2"/>
   </ram:RAM>
  </annotation>
</species>
```

3.2.6 Encoding genetic information

Before we go into detail on how to handle reactions, we explain how the FBC extension is used to encode genetic information. Genetic information contains of two parts; composition of macromolecules and the relation between enzymes and the reactions they are catalyzing.

In terms of complexity, we look at two types of macromolecules: These are either complexes made from several genes or they are the direct result of a single gene, e.g., the Gal1 promoter is the product of the Open Reading Frame YBR020W. Complexes can also be the combination of multiple gene products from the same type. These are then called dimers, trimers, etc. An example would be the enzyme *isocitrate dehydrogenase* in yeast, composed of the gene products YOR136W and YNL037C, where both gene products participate as dimers. For both complexes and gene products it is unnecessary to create a new species for each gene as we are only interested in the final product. Hence, we use the genetic information given for the creation of the biomass producing reactions but do not model each gene product individually.

If the composition of a macromolecule is available we add the information to the SBML file via the **fbc:GeneProduct** elements.

Each of the macromolecules is also present as a **species** and additionally we create an **fbc:GeneProduct** element for each one of them. An **fbc:GeneProduct** has the following attributes:

- **fbc:id** (string) Because the **id** of the macromolecule species is unique, we can reuse it as it is also unique among the gene products. To reflect the meaning of the gene product a bit more, we also suggest using the gene identifier as id for monomers and using 'complex_counter' for complexes and multimers.
- **fbc:associatedSpecies** (string) The **id** of the macromolecule **Species** associated to the gene product is placed here.
- **fbc:label** (string) The label is currently not well defined. The documentation of FBC simply lists this as a field for "additional information on the gene product". Hence, we do not violate existing rules, when we enter the full gene composition of the macromolecule at this position. We encode them as "number*gene id AND number*gene id AND ..." to an easily parsable format. Please see the *resalloc* model for examples.

People who have already worked with the FBC package might find the use of **fbc:label** unnecessary as FBC provides **fbc:and** inside **fbc:geneProductAssociation** to state which genes are necessary to catalyze a reaction, e.g.

```
<reaction id="reaction_2" reversible="false">
 <fbc:geneProductAssociation fbc:id="reaction2">
  <fbc:and>
   <fbc:geneProductRef fbc:geneProduct="YVR173" />
   <fbc:geneProductRef fbc:geneProduct="YKT009W" />
  </fbc:and>
 </fbc:geneProductAssociation>
</reaction>
```

While this states which genes are involed, we are forced to repeat the **fbc:geneProductRef** inside the **fbc:and** to reflect if a gene product enters a macromolecule multiple times. As this can easily lead to misunderstandings, we have chosen to save the gene composition for the enzyme in **fbc:label**. Please note, that this information is necessary for the construction of the protein producing reactions. Once these are defined it is possible, yet inadvisable, to forget the genetic information of the protein.

So far we have not talked about isoenzymes in this context. Isoenzymes are enzymes differing in amino acid sequence but catalyzing the same reaction. Consider the reaction V_1 to be catalyzed by either $E_{1,1}$ or $E_{1,2}$ with the respective turnover numbers $k_{\text{cat},1,1}$ and $k_{\text{cat},1,2}$. While for most cases it would be possible to change the format of the capacity constraint to

$$|V_1| \leq k_{\text{cat},\pm 1,1} E_{1,1} + k_{\text{cat},\pm 1,2} E_{1,2}, \tag{3.16}$$

we must consider situations in which at least one of the isoenzymes catalyzes more than one reaction, e.g., V_2 being catalyzed by $E_{1,2}$. This can only be clearly expressed if each reaction is maximally catalyzed by a single enzyme. We realize this by copying the reaction in the amount of present isoenzymes. This means for the example we transform (3.16) to

$$\left|\frac{V_{1,1}}{k_{\text{cat},\pm 1,1}}\right| \leq E_{1,1} \tag{3.17}$$

$$\left|\frac{V_{1,2}}{k_{\text{cat},\pm 1,2}}\right| + \left|\frac{V_2}{k_{\text{cat},\pm 2}}\right| \leq E_{1,2}, \tag{3.18}$$

with $V_{1,1}$ and $V_{1,2}$ sharing their stoichiometry. This also makes it easier to assign the k_{cat} values as each reaction can maximally have a single pair of forward and reverse constants.

3.2.7 Reactions

The **Reaction** elements are collected in the **listOfReactions**. As with species we want to present the relevant attributes of a **Reaction** starting with the required ones.

- **id** (string) The **id** must be unique among the reactions. Reactions are solely addressed via their **id**.
- **reversible** (boolean) This flag decides whether a reaction is reversible or not. In a lot of FBA models we encountered this was used as slack flag and allowing for the reaction flux to assume negative values. SBML L3V2 clearly states, that **reversible**='false' means that the flux must be larger or equal zero during the simulation. We use this to define the flux reversibility via box constraints, see Equation (2.30).
- **fast** (boolean) Removed in SBML L3V2. We keep it for compatibility to older versions in the code, but do not use it to convey any information.
- **listOfReactants** This list of **SpeciesReferences** contains the **id** and **stoichiometry** for species consumed during the reaction. The list can be empty.

- **listOfProducts** This list contains **SpeciesReferences** with the **id** and **stoichiometry** for species created during the reaction.

As with the **Species**, **Reactions** have optional arguments as well. The **name** attribute can be used for the scientific name of the reactions. If there is an enzyme catalyzing the reaction, we add a **fbc:geneProductAssociation** as shown in the last section to encode this.

We add three more attributes to each reaction by using another **ram:RAM** node in the **Annotation**. The specialized **ram:reaction** elements provide the attributes for that

- **ram:maintenanceScaling** (string) Points to a **Parameter** containing the value for the construction of the maintenance H_a, see (3.4). If the **Reaction** has no maintenance constraint attached, this must be set to `zero`.
- **ram:kcatForward** (string). Points to a **Parameter** containing the forward turnover number $k_{\text{cat},+}$. **Reaction** elements without a **fbc:geneProductAssociation** can leave this attribute empty. For all other reactions, the parameters value must be positive.
- **ram:kcatBackward** (string) Analog to the forward value but instead containing the respective reverse constant. Reversible reactions with **fbc:geneProductAssociation** must have a non-zero reverse constant. Irreversible reactions, on the other hand, must have this parameter set to `zero`.

By enforcing values for catalytic constants depending on the existence of an associated gene product, we add a simple way to check for errors in the SBML file. We suggest issuing warnings to the user if turnover numbers are assigned without an associated enzyme.

The example for a typical biomass producing reaction is taken from the *resalloc* model, see Appendix A.1.

```
<reaction id="PMetab1" reversible="false" fast="false">
 <annotation>
  <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
   <ram:reaction ram:kcatForward="kcatR3" ram:kcatBackward="zero"
    ram:maintenanceScaling="zero"/>
  </ram:RAM>
 </annotation>
 <fbc:geneProductAssociation fbc:id="Ribosome">
  <fbc:geneProductRef fbc:geneProduct="R" />
 </fbc:geneProductAssociation>
 <listOfReactants>
```

```
  <speciesReference species="AA" stoichiometry="200" constant="true"/>
  <speciesReference species="ATP" stoichiometry="800" constant="true"/>
 </listOfReactants>
 <listOfProducts>
  <speciesReference species="Emeta1" stoichiometry="1" constant="true"/>
 </listOfProducts>
</reaction>
```

3.2.8 Naming conventions

We already hinted at a possible connection between compartments and ids for species and reactions. From our experience, it can be very challenging to find errors based on misinterpretation of metabolite or reaction locations in deFBA models as deFBA does not model compartments explicitly. Instead, we should utilize existing knowledge from the FBA model at base as these often include compartments. To transfer this knowledge over to the deFBA model, we suggest adding a compartment code to each species and reaction in the model. This means, if the species originally has the id 'atp', we add the id from the compartment it is located in as 'atp_cytoplasm'. As we will use the ids from the SBML as the identifiers in the deFBA model, we can see the location of the species at first glance.

Basically, the same goes for reactions, but we have to distinguish between metabolic, transport, and production reactions. Metabolic reactions only acting in a single compartment can be clarified by adding the compartment code. Transporters always act between two compartments. Hence, we simply suggest adding both codes in the form 'transport_external_cytosol'. Production locations can be handled like metabolic reactions, but we suggest even adding an 'synth_' at the beginning of the id. This way, the user can immediately see, that this reaction is not part of the original FBA model.

3.2.9 Future of RAM

The RAM extension was tailored to save and manipulate deFBA models. But while designing the different components we always had other resource allocation problems like the RBA or ME models in mind. This is why we called it Resource Allocation Modeling instead of DEFBA modeling. Talks to the developers of RBA and SBML have proven that the functions we have integrated are asked for by several groups. We are hoping to gather these people and use RAM as foundation to find a shared standard. This would make it possible to create single model files which include all data necessary to evaluate a model with the different methods. We strongly believe that the community and the end-user would highly benefit from that.

3.3 Generating deFBA models

Equipped with the means to save and exchange deFBA models via SBML, we want to explain how we can generate models starting with gene annotated FBA models. It is not a focus of this work to explain this protocol in every detail, hence we advise the reader to study [91], before applying the protocol to create a model on their own.

3.3.1 Prerequisites

The process of metabolic reconstruction is a scientific field its own and we will not venture into it here. For the interested reader we recommend [35] for a detailed discussion of the problem. Instead we want to utilize existing metabolic reconstructions and show how we can use these to our advantage. To this day, roughly 2600 draft reconstructions for a multitude of organisms are freely available on the internet via databases like BioModels [21], [62], [60] and BiGG Models [57]. If no metabolic reconstruction but a full genome-sequence of the organism is available, the protocol of Thiele and Palsson [104] can be applied to generate a metabolic reconstruction.

But even if a metabolic reconstruction is available, not any model is suitable to be extended to a deFBA model. First off, it is necessary that all genes used in the model are given, e.g., as **fbc:geneProductAssociation** (cf. Section 3.2.6). Additionally, amino acid compositions for these genes are needed. Otherwise, it becomes nearly impossible to construct the enzyme producing reactions. Of course, this makes it also necessary for the model to include amino acids at all. Model with lower detail on the metabolite level can not be extended to deFBA models.

Secondly, full genome scale models might be simply too large to be simulated with deFBA as the resulting linear programs grow increasingly complex to solve. This is especially true if the time scale on which the model is to be evaluated is very large. In Chapter 4, we will present a new approach for handling deFBA models which effectively lowers the computational cost. Yet, the limit for deFBA models is roughly at 500 metabolic reactions at the moment. Networks of this size have been successfully simulated as shown in [90]. If the size of the starting network is already too large, we suggest using tools like the *minimal network finder* [94], *redGEM* [4], or *NetworkReducer* [32]. While these methods differ vastly, they share the concept to preserve some functions of the model, like growing under specific conditions or retaining some pathways, and minimize the network size from there.

For the rest of this section, we assume to have a reconstruction on the FBA level available and explain how to amend the additional deFBA components.

3.3.2 Building enzyme production

As explained in Section 3.2.6 we have to make some changes in the model if isoenzymes are present. In these cases we copy the respective reactions until each reaction is exactly catalyzed by a single enzyme. At this point, we want to include all isoenzymes in the model, as we can not decide which of these will be expressed during simulation.

For the construction of enzyme producing reactions, we need the amino acid composition for them. These can usually be obtained in the FASTA format [87] from online databases such as Genbank [8] or UniProt [3]. We must highlight that UniProt currently has a superior API, which does allow the user not only to gather data on a the gene sequence, but at the same time gives access to the Enzyme Commission (E.C.) number [112] and more importantly the subunit stoichiometry for enzyme complexes.

If these more general databases do not have the data available, we suggest searching for organism specific databases. In our own experience we could obtain a complete FASTA file on *S. cerevisiae* from the Saccharomyces Genome Database [22].

Constructing the production reaction for a simple monomer enzyme is done by summing up the amino acids cost from the FASTA file and setting these as reactants for the reaction. For the initiation of the peptide chain a single 10-formyltetrahydrofolate (fTHF) is converted into tetrahydrofolate (THF) [79]. Additionally, the energy cost for the elongation of chain must be paid: Per amino acid $\mathrm{AA_i}$ added to the chain, a single ATP is hydrolyzed into AMP and $\mathrm{PP_i}$, and two GTP are hydrolyzed into two GDP and two $\mathrm{P_i}$ [79].

When handling an enzyme complex (or multimer), we calculate the cost for each subunit and add those up with respect to the subunit stoichiometry. While the stoichiometric information for multimers can often be found on UniProt, data on enzyme complexes is usually not easily available and it is necessary to do a literature search by hand. We must stress the relevance of this step. Wrong enzyme compositions can vastly change the trade-off of single enzymes immensely.

The prototype for all biomass reactions can be constructed as

$$\begin{aligned} V_P : &\sum_{i=1}^{20} S_{i,p}\mathrm{AA}_i + n_1\mathrm{ATP} + 2n_1\mathrm{GTP} + n_2\mathrm{fTHF} \\ &\longrightarrow \mathrm{E} + n_1\mathrm{AMP} + n_1\mathrm{PP_i} + 2n_1\mathrm{GDP} + 2n_1\mathrm{P_i} + n_2\mathrm{THF}, \end{aligned} \tag{3.19}$$

with $S_{i,p}$ being the amount of amino acid $\mathrm{AA_i}$ needed, the count of chain elongations $n_1 = \sum_{i=1}^{20} S_{i,p}$ and n_2 being the number of subunits.

Depending on the data available to the user, it might be useful to also add the cost of mRNA to the synthesis of the enzymes directly, as we have seen with the ME-models (cf. Section 2.2.1). In such a case, we will add RNA cost via quota constraints as explained in Section 3.3.3.

We must also point out that translational machinery, such as the ribosome, is handled identically to very large enzyme complexes. This means a single production reaction describes the full translation and assembly process. The composition for the ribosomes are usually easy to find in comparison to single enzymes. We suggest looking for these in the Kyoto Encyclopedia of Genes and Genomes (KEGG) [53], or the

Ribosomal Protein Gene Database [77]. We are aware, that using a single reaction for the creation of the ribosome is very restrictive. Using this formulation, all genes associated to the ribosome must be expressed to construct ribosome. In reality, the ribosome is very robust and can retain full or limited functionality even if multiple subunits are missing [101]. But with the current sparse availability of data on ribosome subunit functionality in different organisms, we will usually have to rely on a single ribosome producing reaction.

3.3.3 Setting up quota

We are mostly interested in the quota species as we expect that non-catalytic species will not be produced in an optimal solution unless enforced via constraints. Typical examples are DNA, membranes, cell wall and RNA, if not directly attached to the production rates of macromolecules. These serve no catalytic function in the deFBA model. Hence, we must enforce their production via the biomass composition constraint (3.14f) as we would otherwise neglect a significant part of the resource cost while growing. As explained in Section 2.3.5, we assume the amount of any quota species is connected to the amount of total biomass via fixed scaling factor, e.g., 10% of total biomass must consist of DNA at any given time.

While in some cases the ratio of a biomass product will be directly available in the literature, we can not expect to find exact values for all types of quota components. Instead, we can have a look at the objective from the original FBA model (1.20). This objective is a single biomass creating flux, which represents the typical composition of the organisms' biomass, from which we can derive the desired constraints. Unfortunately, this process is a bit abstract, so we use an example taken from the Yeast 6.06 model to explain the process. The biomass reaction from this model is shown in Table 3.2. Looking at the reactants of the reaction we can group these as: proteins (charged transfer RNAs), cell wall (mannan and β-D-glucan), storage components (glycogen and trehalose), DNA (dAMP, dCMP, dGMP, dTMP), RNA (AMP, CMP, GMP, UMP), membrane (lumped lipid), small molecules, and the ATP requirements for polymerization. We condense the reaction to

$$\sum_{i \in \text{Reac}} S_i Z_i \to 1\text{biomass} + \sum_{i \in \text{Bipr}} S_i Z_i, \tag{3.20}$$

with the index sets Reac representing the reactants and Bipr being the byproducts. The stoichiometries in the biomass reaction are chosen in a way that weighting the stoichiometries with the respective molecular weights will add up to one to make the construction of the quota elements easier, i.e.,

$$\sum_{i \in \text{Reac}} S_i w_i - \sum_{i \in \text{Bipr}} S_i w_i = 1, \tag{3.21}$$

Table 3.2: Biomass reaction of the Yeast 6.06 model.

Reactants	Stoichiometry	Products	Stoichiometry
Ala-tRNA(Ala)	0.4588	tRNA(Ala)	0.4588
Arg-tRNA(Arg)	0.1607	tRNA(Arg)	0.1607
Asn-tRNA(Asn)	0.1017	tRNA(Asn)	0.1017
Asp-tRNA(Asp)	0.2975	tRNA(Asp)	0.2975
Cys-tRNA(Cys)	0.0066	tRNA(Cys)	0.0066
Gln-tRNA(Gln)	0.1054	tRNA(Gln)	0.1054
Glu-tRNA(Glu)	0.3018	tRNA(Glu)	0.3018
Gly-tRNA(Gly)	0.2904	tRNA(Gly)	0.2904
His-tRNA(His)	0.0663	tRNA(His)	0.0663
Ile-tRNA(Ile)	0.1927	tRNA(Ile)	0.1927
Leu-tRNA(Leu)	0.2964	tRNA(Leu)	0.2964
Lys-tRNA(Lys)	0.2862	tRNA(Lys)	0.2862
Met-tRNA(Met)	0.0507	tRNA(Met)	0.0507
Phe-tRNA(Phe)	0.1339	tRNA(Phe)	0.1339
Pro-tRNA(Pro)	0.1647	tRNA(Pro)	0.1647
Ser-tRNA(Ser)	0.1854	tRNA(Ser)	0.1854
Thr-tRNA(Thr)	0.1914	tRNA(Thr)	0.1914
Trp-tRNA(Trp)	0.0284	tRNA(Trp)	0.0284
Tyr-tRNA(Tyr)	0.1020	tRNA(Tyr)	0.1020
Val-tRNA(Val)	0.2646	tRNA(Val)	0.2646
ATP	59.2760	ADP	59.2760
H2O	59.2760	phosphate	58.70001
(1→3)-β-D-glucan	1.1348	H^+	59.3050
(1→6)-β-D-glucan	1.1348	biomass	1
mannan	0.8079		
glycogen	0.5185		
trehalose	0.0234		
riboflavin	0.00099		
lipid	1		
sulphate	0.0200		
dAMP	0.0036		
dCMP	0.0024		
dGMP	0.0024		
dTMP	0.0036		
AMP	0.0460		
CMP	0.0447		
GMP	0.0460		
UMP	0.0599		

with the molecular weight w_i. The percentage of the biomass consisting of certain components can easily be calculated from here. For example we can determine the percentage for protein components

$$\phi_{\text{prot}} = 1 + \sum_{i\in\text{Bipr}} S_i w_i - \sum_{i\in\text{Reac},i\notin\text{Prot}} S_i w_i = \sum_{i\notin\text{Prot}} S_i w_i, \tag{3.22}$$

with Prot being the index set containing all charged transfer RNAs. The production reaction for the protein species can be determined as

$$\sum_{i\in\text{Prot}} \frac{S_i}{\phi_{\text{prot}}} Z_i + \frac{S_{\text{ATP}}}{\phi_{\text{prot}}}\text{ATP} + \frac{S_{H_2O}}{\phi_{\text{prot}}}\text{H}_2\text{O} \tag{3.23}$$

$$\rightarrow 1\ \text{prot} + \sum_{i\in\text{tRNA}} \frac{S_i}{\phi_{\text{prot}}} Z_i + \frac{S_{ADP}}{\phi_{\text{prot}}}\text{ADP} + \frac{S_{\text{phosphate}}}{\phi_{\text{prot}}}\text{phosphate} + \frac{S_{H^+}}{\phi_{\text{prot}}}\text{H}^+, \tag{3.24}$$

with the index set tRNA containing all protein products.

3.3.4 Determining turnover numbers

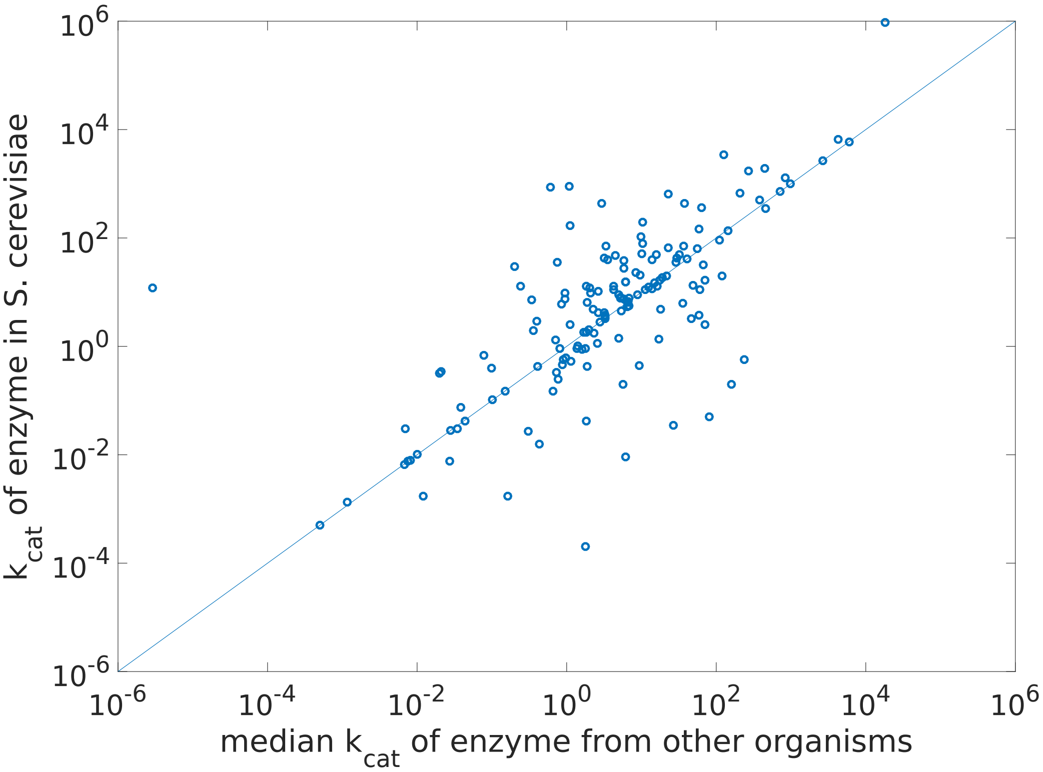


Figure 3.1: Turnover rates in yeast versus the median kcat values from other organisms. The Figure is taken from [91].

One of the biggest challenges in the construction of a deFBA model is to acquire the catalytic constants k_{cat} needed for the enzyme capacity constraint (3.14g). While it is possible to derive these from experiments as shown in [40], we are relying on the turnover numbers obtained from online databases as these have proven to be sufficient approximations to *in vivo* turnover numbers [26]. The main sources for the k_{cat} values are BRENDA [97] and SABIO-RK [113]. SABIO-RK is special in comparison to other databases as it is completely hand-curated. We believe that this is superior to automatic data gathering as these can easily lead to misinterpretation of the measurements. As the enzyme capacity is very important to whether a pathway is used in the simulation, we recommend to check automatically acquired values for feasibility.

Please note, that databases often enable the user to filter for different organisms, mutant strain, pH-value, etc. From our experience looking for wild-type data taken at a physiological pH should be the way to use those filters. If no data from the modeled organism is available, one should use data from other organisms. In fact, using the median of all values available is the way to go as shown in [91]. To visualize this result, we have taken Figure 3.1 from this work, which compares the turnover numbers in *S. cerevisiae* with the mean values obtained from other organisms found in BRENDA. If possible, the modeler should make a sensitivity analysis of the final model, to check the impact of the turnover numbers taken from other organisms.

A final warning for gathering values for isoenzymes: In many cases the user will use the E.C. number as search string, but E.C. numbers are **reaction** and not **enzyme specific**. This might lead to shared turnover numbers for isoenzymes, which in reality is rarely the case.

3.3.5 Molecular weights & objective weights

The molecular weights $w_{\mathcal{P}}$ for macromolecules and storage species can be looked up in databases like BioNumbers [73]. If these are not available, we can approximate the molecular weight for biomass components via the amino acid composition. This is done by summing the weights of the individual amino acids. Due to the different bonding mechanisms between the amino acids in the molecules, we can not calculate the exact weights without having knowledge on each individual bond. Therefore, the sum of amino acid weights gives an upper approximation to the correct weight.

As stated in the first section of this chapter, we extended the deFBA formalism to operate with special objective to measure the objective value of a species instead of just using their molecular weight. As we will later see in Chapter 4, the deFBA can produce artificial results if some macromolecules have a very good objective yield in comparison to their resource cost. To eliminate this kind of behavior, species can be "deleted" from their objective by setting their objective weight to zero. While these objective values are in fact a tuning parameters to counter unusual behavior of the model, we suggest to chose the objective weights identical to the molecular weights for all enzymatic species E and set these to zero for all quota Q and storage species S.

3.3.6 Model validation

While it is impossible to validate the model with absolute certainty without numerous experiments to measure the gene expression levels, one can often find growth rates in different environments. The growth rate μ in the deFBA model can be determined as

$$\mu(t) = \frac{1}{B(t)} \frac{\mathrm{d}B(t)}{\mathrm{d}t}, \tag{3.25}$$

with the total biomass $B(t)$. Given a non-limiting environment the growth-rate will plateau at some point, when the model reaches balanced growth with constant biomass composition. We can use this plateau value to compare against growth rates from the literature. If the simulated value is too small $\mu \leq \mu_{\text{literature}}$, one should check the k_{cat} values again, as these are likely the cause for a bottleneck in a core pathway.

If the simulated values are too large, this usually hints at the lack of maintenance or quota requirements. The easiest solution is to enforce a maintenance flux, e.g. hydrolysis of ATP into ADP, to lower the growth rate, but the construction of any biomass composition constraints should also be checked.

3.4 deFBA-Python package

The final step in the deFBA pipeline is the simulation of the model. We implemented a software toolbox in `Python 2.7` using the `scipy` toolbox [52] and the `LinOpt` class created by Steffen Waldherr as basis. We call the result `deFBA-Python` package, which is freely available at `https://bitbucket.org/hlindhor/defba-python-package`. Our goal with this software is comparable to the ones we have with RAM: Lay down a foundation for a unified framework for different kinds of resource allocation problems. In this section we want to give a basic overview on the core functions of `deFBA-Python` and explain difficulties during the development.

Over the course of this work, we will further extend and adapt the deFBA formalism. All these extensions are also included `deFBA-Python` and we will talk these in the respective chapters.

3.4.1 Linopt class

We started with the existing `LinOpt` class, which provides methods to input *linear time-invariant optimization* problems (LTI) in the standard form

$$\min_u \int_{t_0}^{t_{\text{end}}} (Cx(t) + Du(t) + E)\ \mathrm{d}t + Fx(t_{\text{end}}) \tag{3.26a}$$

$$\text{s.t. } x(t_0) = x_0 \tag{3.26b}$$

$$\dot{x} = Ax + Bu + u_0 \tag{3.26c}$$

$$G_x x(t) + G_u u(t) + G_k \leq 0 \tag{3.26d}$$

$$K_u u(t) + K_k = 0 \tag{3.26e}$$

$$H_x x(t_{\text{end}}) + H_k \leq 0. \tag{3.26f}$$

Providing also a class to collocate the trajectories it is possible to transform this problem to a linear program, which can be solved by the linear solvers `Gurobi` [43] and `cvxopt` [2]. In the collocation, Lagrange polynomials are used as basis functions and the user can choose from different schemes for setting the collocation points; Lobatto-, Radau-, and reverse Radau-collocation.

We extended the existing code such that the matrices are stored in a more efficient sparse matrix format called 'list-of-lists' [103]. Furthermore, we added interfaces to two more linear solvers `CPLEX` [102] and `SoPlex` [114], which are both freely available for academic use. The specialties of SoPlex are an iterative refinement procedure, which allows to compute high-precision solution, and a compile option to allow for 80bit extended precision, which is very beneficial for numerically difficult problems.

3.4.2 DefbaModel class

The package is build around the `DefbaModel` class, which contains the full deFBA model and provides all methods to work with the model. We store the model in its matrix form as (3.14a) via `numpy` matrices. Additional information on the model needed for the export to SBML, like compartments and mapping from reactions to their catalyzing enzymes, are kept in dictionaries, which are not used for the simulation. To make the class as accessible as possible we added multiple ways for the initialization of a model. At its core the model must only consist of the stoichiometric matrix, the molecular weights for macromolecules and a list containing the numbers for the different model elements, e.g., 10 external species, 12 metabolites, 5 macromolecules, 10 uptake reactions, 3 metabolic reactions, and 5 biomass producing reactions.

While the software accepts the model by giving the matrices directly, see the *carbon core* model from the examples folder, the software is centered around the import from SBML and RAM. Nevertheless, the extended format (3.14a) allows for additional system dynamics which are not part of the original SBML model

$$\dot{Z}(t) = RZ(t) + \begin{pmatrix} S_{\mathcal{Y}} \\ S_{\mathcal{P}} \end{pmatrix} V(t) + q, \tag{3.27}$$

with the *dilution matrix* $R \in \mathbb{R}^{n,n}$, $[R] = 1/\mathrm{h}$ and the constant input vector $q \in \mathbb{R}^n$, $[q] = \mathrm{mol/h}$. We added a method to add these dynamics via R and q directly or via dictionaries. The software currently only supports constant inputs q, but we plan to include time-dependent inputs in the future.

The two most important functions related to the regular deFBA are the two methods `DefbaModel.deFBA` and `DefbaModel.RBA`. The latter method is an adaption of the static RBA approach (cf. Section 2.2) to identify an optimal enzyme distribution for balanced growth using deFBA constraints. This problem is defined as

$$\begin{aligned}
\max_{V,P,\mu} \quad & \mu(v_{\text{intake}}, B_0, Y_0) && \text{(3.28a)}\\
\text{s.t.} \quad & S_{\mathcal{P}}V = \mu P && \text{(3.28b)}\\
& S_{\mathcal{X}}V = 0 && \text{(3.28c)}\\
& H_{a,\mathcal{P}}P \leq H_g V && \text{(3.28d)}\\
& H_{b,\mathcal{P}}P \leq 0 && \text{(3.28e)}\\
& H_c V \leq H_{f,\mathcal{P}}P && \text{(3.28f)}\\
& v_{\min} \leq V \leq v_{\max} && \text{(3.28g)}\\
& P \geq 0 && \text{(3.28h)}\\
& w_{\mathcal{P}}^T P = B_0 && \text{(3.28i)}\\
& V_{\mathcal{Y}} \leq v_{\text{intake}}(Y_0). && \text{(3.28j)}
\end{aligned}$$

This might seem like a regular RBA, but we keep the formulation in molar amounts for this problem, which changes the handling a bit. First off, the result depends on the uptake limits $v_{\text{intake}}(Y_0) \in \mathbb{R}^{m_y}$. Due to the formulation in molar amounts, we use this constraint to a check whether a nutrient type is available with $v_{\text{intake}} \in \{0, \infty\}^{m_y}$. The value is chosen as ∞ for available nutrients and zero otherwise. This definition of the constraint is only correct, if all exchange reactions are pointing to the inside. Otherwise, the constraint must be adapted such that only uptake of present species is allowed. At the same time, the secretion of all species must be allowed. The software checks automatically in which direction the transport reaction are defined and adapts the constraint accordingly. We discuss this in more detail in Section 4.3.

The balanced growth constraint is enforced by substitution of the dynamics with (3.28b). Finally, we must add a constraint for the biomass amount in the system (3.28i) to get a well defined optimization problem.

Because (3.28) is quadratic in μ as discussed in Section 2.2, we apply a binary search algorithm to determine the maximal achievable growth rate. Depending on the problem size, this is much quicker than solving the quadratic problem itself. The user has full control over the algorithm as it is possible to insert an initial guess for μ and determine the accuracy of the calculation by setting a fixed tolerance for the binary search.

Solving problem (3.28) is simple in comparison to a full deFBA and we use it for two main applications in `deFBA-Python`: Firstly, we calculate initial values for deFBA simulations with it. Secondly, if we are only interested in the maximal achievable growth rate for, e.g., model validation we can use `DefbaModel.RBA` instead of `DefbaModel.deFBA`.

The `DefbaModel.deFBA` method on the other hand presents the regular deFBA as described in (3.14a). Parameters are the linear solver, step size h, the end-time t_{end}, and initial values Y_0, P_0. If no initial values are given, the method automatically calls `DefbaModel.RBA` for the given environment and uses the results to initialize the simulation.

After the simulations the calculated values for all species and reactions can be exported to comma-seperated-values files (CSV) and/or be plotted in Python using the `matplotlib` library.

3.4.3 RAM interface

The interface between `DefbaModel` and SBML files in RAM format is a core feature of `deFBA-Python`. During the implementation of the `sbmlimport` module, we focused on an automatic check to ensure the model is encoded correctly. For example, SBML and RAM allow that a metabolite is equipped with a non-zero **ram:objectiveWeight** attribute. In a deFBA model this does not make any sense and hints at an error. These warning are either issued directly on the terminal to the user or collected in an error log.

Factual errors, like a biomass species P missing the **ram:molecularWeight** attribute, will immediately stop the parsing of the model and inform the user about the error via the terminal.

Difficult parts in the implementation of the import were the correct identification of the reaction type and construction of the enzyme capacity constraint matrix H_c. The problem with the reactions arises mostly from lumped reactions. Strong lumping might lead to reactions using external species to directly produce biomass components. Hence, we can not simply look at the reactants of a reaction but the products as well. We came up with the following hierarchy for the reactions

- Macromolecule reaction: Must contain at least one macromolecule species as either product or reactant; does not matter if this is a storage, enzyme or quota component. May contain any other species types as well. Because most biomass species are assembled by the ribosome or complementary biological machinery, these reactions must usually be irreversible with the macromolecule being a product. We log reactions for which this is not the case and issue a warning; unless the macromolecule is storage.

- Exchange reaction: Must contain an external species as reactant and may not contain any biomass species. If a reversible reaction contains external species as products, but no external species as reactants, it is classified as exchange reaction and its direction is automatically reversed. This way we guarantee the reaction is in agreement with the convention that exchange reactions are pointing inwards. Problematic are currently reversible reactions with external species as reactants and products at the same time and their directions are set arbitrarily.

- Metabolic reactions: We simply collect all reactions not fitting into the other groups.

The construction of the enzyme capacity constraint is quite complex as we need to cover all possible sign combinations for reversible reactions. In the construction of H_c (2.41), we must divide by the k_{kat} values. Therefore, we ensured that no k_{kat} value used in the model can be zero, cf. Section 3.2.7.

With further extensions of the package in mind, we added a management system for all parameters used in the model. If the user followed our guideline for using RAM, all parameters have an individual identifier and we store all their occurrences in the model. This makes it easy to manipulate model parameters and implement parameter estimation methods in the future.

The export via `sbmlexport` module is in comparison very simple. If the above mentioned parameter structure is present, it will be used to generate the **listOfParameters** and link all parameters respectively. Otherwise, the parameters are directly taken from the system matrices and are just given a running number code. At this time the export is only very rudimentary in its function as we do not expect the user to construct a genome-scale model by inserting the system matrices but instead encode them directly in SBML.

Finally, we added a `sbmltricks` module aimed at the modification of SBML files. Currently, it provides a method to easily create knockout mutants from the current model in which certain gene codes and all according reactions and macromolecules are deleted. In the next release of `deFBA-Python` we plan to include the code, we used for automatic data gathering here. Unfortunately, these methods are very dependent on the format of the original FBA file, the organism and the according databanks. This makes it hard to write code, which can be applied to as many FBA models as possible.

3.5 Conclusion

In this chapter, we presented the complete pipeline to create and analyze deFBA models. While this pipeline in itself is already very useful in predicting how gene expressions and the respective enzyme levels change in a dynamic environment, the method of the deFBA itself can be further improved. This will lead us in the next chapters to two new methods, based on the presented results.

4 Short-term deFBA

The short-term deFBA was published in the joined publication [66] with Alexandra Reimers within the European project ROBUSTYEAST. Idea, concept and manuscript were developed by H.L. while A.R. supported in the proofs of Theorem 1 and Theorem 2. In addition to the published results, we refined the calculation of time horizons by including the current environment and present a new mixed-integer problem to make these calculations more accurate.

This chapter presents the first functional extension to the regular deFBA. As the problem size gets very large when using a small stepsize in combination with a large end-time t_{end} it can become beneficial to transform the arising single optimization problem with a series of smaller ones. This is done by combining the deFBA with the idea of a receding prediction horizon as is used in model predictive control (MPC) [18]. This will allow us to counter the huge computational cost of genome scale models and prepare to utilize the deFBA in control applications.

4.1 Why using a receding prediction horizon?

Most readers will wonder why we introduce a receding prediction horizon to the deFBA. Our first reason can be seen as a bit philosophical because using a prediction horizon mimics the limited knowledge on the future in real biological systems. Single-cell organisms have developed a wide array of mechanisms to react and predict changes in the nutrient situation, like catabolite repression in the presence of better suited energy sources [96] or sensing of metabolic intermediates to control the metabolism [23]. While most of these structures must still be further investigated they implicitly result in a time frame for which the cells plans ahead. Of course, this is the result of complex interactions in the internal regulation of the cell. Nevertheless, we believe that the implementation of a limited prediction horizon will improve the accuracy of deFBA predictions.

The second reason for using the prediction horizon is a possible decrease in computational cost as deFBA problems become increasingly harder to solve with increasing end-time. This is not only due to the size of the problem but also to the exponential growth on larger time frames, which makes solving them even more numerically demanding due to changes in the size of the states in multiple orders of magnitude. By segmenting the problem into multiple sub-problems, we can lower the computational cost and solve the problem faster.

Another problem with the fixed end-time might occur if starvation takes place. Assuming the model includes maintenance reactions this might lead to an overall infeasible problem. But if we use a receding horizon, we can derive a solution until nutrients and/or storage species deplete.

Additionally, we can eliminate an inherent problem with the deFBA as the results of this method may depend on the chosen end-time t_{end}. To illustrate this, we introduce a simple academical example and show how the end-time effects results.

4.1.1 Enzymatic-growth model

We call the minimal example the *enzymatic-growth* model, which is available in the `deFBA-Python` package as an example. It consists of only three reactions, one nutrient, one metabolite and two biomass products. The irreversible reactions are given as

$$V_A: \quad 1\,N \quad \rightarrow 1\,A \tag{4.1a}$$

$$V_E: \quad 1\,N + 1\,A \quad \rightarrow 1\,E \tag{4.1b}$$

$$V_C: \quad 1\,N + 1\,A \quad \rightarrow 1\,C. \tag{4.1c}$$

The external nutrient N represents a collection of components necessary for growth, such as carbon, nitrogen, etc. Further processed components made from these nutrients are collected as the internal metabolite A. We differentiate the macromolecules into the group of enzymes E, collecting the whole enzymatic machinery needed for growth, and non-enzymatic macromolecules C. These can be interpreted as storage components such as lipids, starch, or glycogen.

We analyzed the model very detailed in [109] in terms of the influence of end-times, initial conditions and parameters of the system. As we are only interested in the end-time, we fix the parameters as shown in Table 4.1. The molecular weights w_C and w_E are chosen in a way to make its biomass yield for C better than for the production of E, meaning the same amount of N can be transformed into more biomass if the cell focuses on the production of storage. This is also reflected in the turnover numbers as the production of storage is also faster $k_C > k_E$.

Table 4.1: Parameter values used in the enzymatic-growth model.

w_C [$\frac{\text{g}}{\text{mol}}$]	w_E [$\frac{\text{g}}{\text{mol}}$]	k_A [h^{-1}]	k_C [h^{-1}]	k_E [h^{-1}]
15	10	1.45	2	1

Therefore, the deFBA model is given as

$$\max_{V} \int_{t_0}^{t_{\text{end}}} w^T Z(t)\ \mathrm{d}t = \int_{t_0}^{t_{\text{end}}} w_C Z_C(t) + w_E Z_E(t)\ \mathrm{d}t \tag{4.2a}$$

$$\text{s.t. } Z_N(t_0) = N_0,\ Z_E(t_0) = E_0,\ Z_C(t_0) = C_0 \tag{4.2b}$$

$$V_E(t) + V_C(t) = V_A(t) \tag{4.2c}$$

$$\dot{Y}_N(t) = -V_A(t) \tag{4.2d}$$

$$\dot{Z}_E(t) = V_E(t) \tag{4.2e}$$

$$\dot{Z}_C(t) = V_C(t) \tag{4.2f}$$

$$\sum_{i\in\{A,C,E\}} \frac{1}{k_i} V_i(t) = \frac{1}{k_A} V_A(t) + \frac{1}{k_C} V_C(t) + \frac{1}{k_E} V_E(t) \le Z_E(t) \tag{4.2g}$$

$$Y_N(t) \ge 0, Z_E(t) \ge 0, Z_M(t) \ge 0,\ \ \forall t \in [t_0, t_{\text{end}}]. \tag{4.2h}$$

While minimal in its size, the model highlights the core of all resource allocation models as the network has to decide to invest the nutrients N either in self-replication by producing more E or in storage C, which has no direct application but is easy to produce ($k_C > k_E$) and has a better objective yield ($w_C \ge w_E$). This decision process is also impacted by nutritional stress, meaning N running out before the end-time t_{end} is reached. To eliminate this influence in the simulation as well, we assume unlimited access to nutrients with $N_0 = \infty$ in (4.2b).

From a biological perspective, we expect a solution of the problem to predict either only investment in the enzymatic part E or a mixed strategy favoring E while keeping a small percentage of the biomass as C. Interestingly, the results are neither. We observe two different types of solution curves depending on the chosen end-time as shown in Figure 4.1. For both plots we used identical initial values $Z_E(0)$=$Z_C(0)$=0.1 mol (4.2b) and used a discretization step size h = 3 min=0.05 h for the numerical solution (cf. Section 4.5). On the left side using an end-time of 10 hours we see a mixed solution in which for the first 8 hours the system only invests in enzymatic machinery and afterwards focuses on producing the storage component with the better yield. The solution depicted in the right plot by contrast shows that only producing C is indeed optimal for an end-time of $t_{\text{end}} = 1.4$ h.

This can be explained by looking at the objective values for the different solutions at t=1.4 h. While production of E yields an objective value of roughly 3.07 g at 1.45 h, the linear increase in C yields a value of 3.65 g at the same time. Therefore, we can deduce that there exists a *switching time* t_{s}, after which all solutions are a mix as seen in Figure 4.1 (Left) and all solutions for an end-time $t_{\text{end}} \le t_{\text{s}}$ are of the linear form. As calculated in [109] this switching time is for the chosen parameter values given as $t_{\text{s}} \approx 1.45$ h.

This example clearly shows the dependency on the end-time. But at the same time

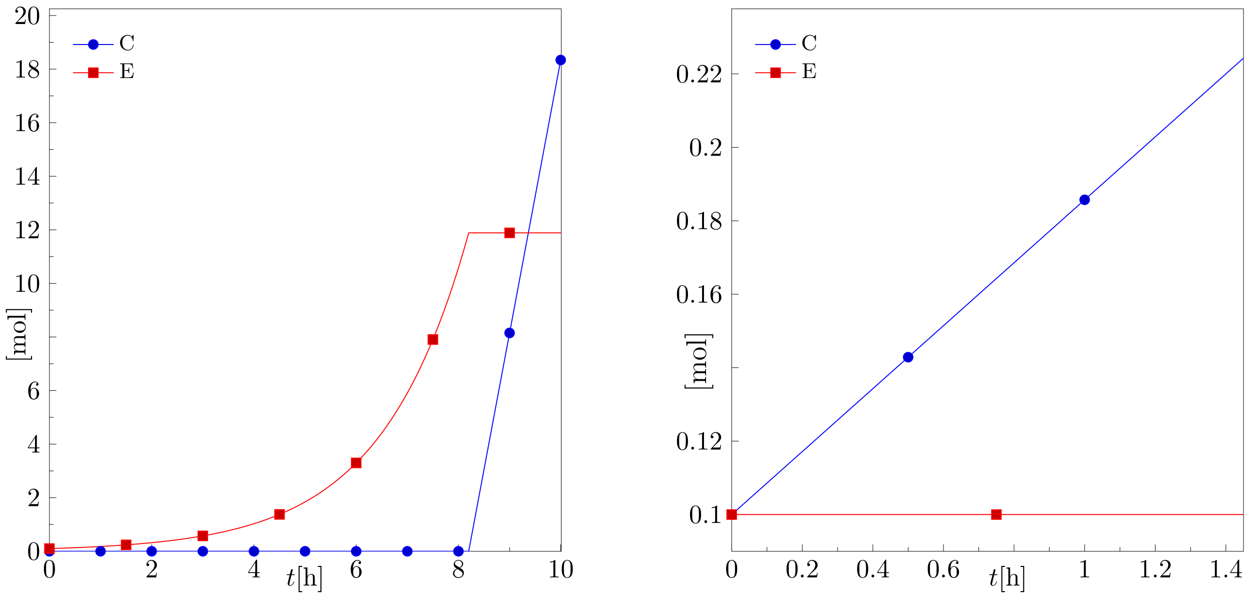


Figure 4.1: Results for the enzymatic growth model. For the simulation we choose initial values as E_0=C_0=0.1 mol (4.2b). Left: End-time was chosen as 10 h. Right: End-time was chosen as 1.45 h.

it highlights the fragments of the optimization approach as the hard-switch (Figure 4.1 (Left)) is usually not observed in biological systems. But with the receding time horizon we can eliminate this problem as well.

4.2 Implementing the receding time horizon

The implementation of the receding time horizon is straightforward. We just need to introduce two parameters to fully define the new approach: The *iteration time* t_i and the *prediction horizon* $t_\mathrm{p} \geq t_\mathrm{i}$ (cf. [18]). The time scale $t \geq t_0$ is split into intervals $[t_k, t_{k+1}]$ by using the time grid

$$\Delta_t(t_\mathrm{i}) = \{t_k = kt_\mathrm{i} + t_0 \mid k \in \mathbb{N}\}, \tag{4.3}$$

defined by the iteration time t_i on which the solutions are piece-wise defined. The goal is to solve a single deFBA starting at the nodes of Δ_t and determine the optimal solution over the prediction horizon t_p.

These individual problems are regular deFBA problem but definitions for initial

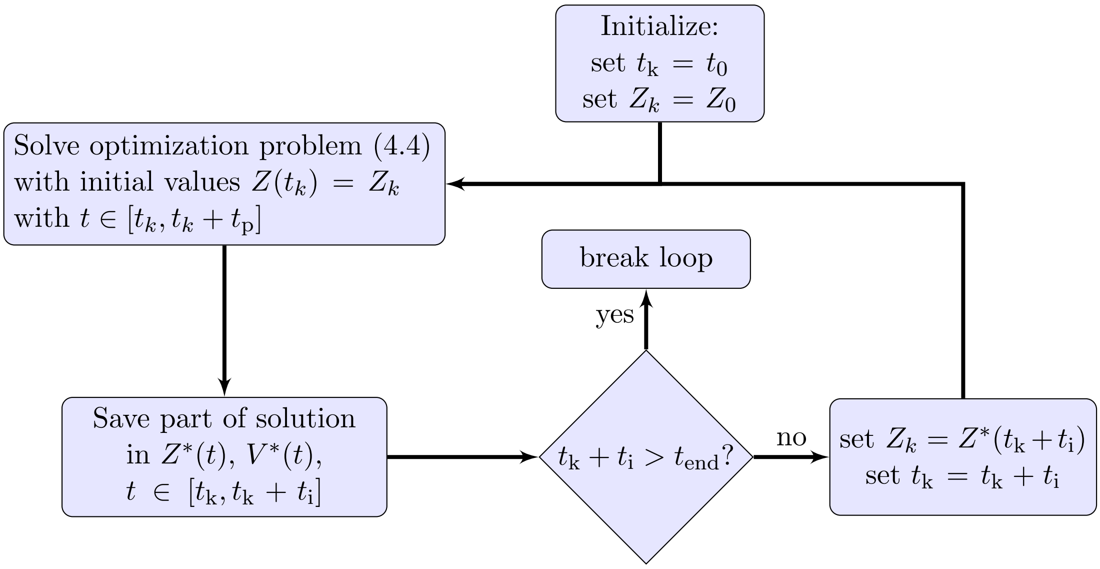

Initialize:
set $t_\mathrm{k} = t_0$
set $Z_k = Z_0$
Solve optimization problem (4.4)
with initial values $Z(t_k) = Z_k$
with $t \in [t_k, t_k + t_\mathrm{p}]$
break loop
yes
Save part of solution
in $Z^*(t)$, $V^*(t)$,
$t \in [t_\mathrm{k}, t_\mathrm{k} + t_\mathrm{i}]$
$t_\mathrm{k} + t_\mathrm{i} > t_\mathrm{end}$?
no
set $Z_k = Z^*(t_\mathrm{k} + t_\mathrm{i})$
set $t_\mathrm{k} = t_\mathrm{k} + t_\mathrm{i}$

Figure 4.2: Flowchart to explain the iterative scheme. After solving each individual optimization problem, the current time and the current state are updated. The overall solution is saved in Z^* and V^*.

values and end-times are adapted to the time grid Δ_t

$$\max_{V(t),P(t)} \int_{t_k}^{t_k+t_\mathrm{p}} B_\mathrm{o}(t)\ \mathrm{d}t \tag{4.4a}$$

$$\text{s.t. } Z(t_k) = Z_k \tag{4.4b}$$

$$\dot{Z}(t) = \begin{pmatrix} S_\mathcal{Y} \\ S_\mathcal{P} \end{pmatrix} V(t) \tag{4.4c}$$

$$S_\mathcal{X} V(t) = 0 \tag{4.4d}$$

$$H_a Z(t) \leq H_g V(t) \tag{4.4e}$$

$$H_b Z(t) \leq 0 \tag{4.4f}$$

$$H_c V(t) \leq H_f Z(t) \tag{4.4g}$$

$$v_\mathrm{min} \leq V(t) \leq v_\mathrm{max} \tag{4.4h}$$

$$Z(t) \geq 0,\ \forall t \in [t_k, t_k + t_\mathrm{p}]. \tag{4.4i}$$

The individual problems in the form (4.4) are connected as shown in the flowchart given by Figure 4.2. The problem is initialized via the initial values Z_0 for the states. After solving (4.4), we save the parts of the calculated trajectories between $[t_k, t_{k+1} = t_k + t_\mathrm{i}]$ to the *overall solution* trajectories $V^*([t_k, t_{k+1}]) = V([t_k, t_{k+1}])$, $Z^*([t_k, t_{k+1}]) = Z([t_k, t_{k+1}])$. As we choose the initial values Z_{k+1} for the next problem as the final ones $Z(t_{k+1})$ from the current one (4.4b), we can guarantee that the overall solution is continuous, but it might be not smooth. If smoothness is a concern, this can be enforced via additional constraints on the state derivatives.

The iteration stops once a given condition is reached. In Figure 4.2 a simple check for a given end-time is used, but is also possible to stop once the nutrients deplete or a certain amount of biomass is reached. We call this iterative process the *short-term deFBA* (sdeFBA). Some readers might find it easier to follow this explanation by looking at the discrete pseudo-code formulation of the sdeFBA presented in Section 4.5.

A huge benefit within the sdeFBA is, that we can manipulate the problem after each iteration, e.g. change the nutrient situation or the biomass to emulate external influences. It is also possible to change the optimization problem to emulate gene-knockouts or other discrete events.

The most natural application of the sdeFBA would be inside a model predictive controller. A possible application is depicted in [34]. Here the objective is to control the light and the nutrient feeds for a tank containing microalgae to maximize the production of beta-carotene. But we have yet to find an experimental partner for testing these kind of applications with our framework.

The problem in using sdeFBA is choosing suitable values for t_i and t_p. From a numerical view we want to minimize the prediction horizon to minimize the size of each individual problem (4.4). At the same time we must choose t_p large enough such that we do not experience problems with artificial behavior as seen with the enzymatic growth model. With the iteration time we face the inverse problem. We want to choose the iteration time maximally to decrease the number of necessary iterations, but we have to choose it small enough such that we keep the artificial solution parts near the prediction horizon (cf. Figure 4.1 (Left) hours 8-10) from entering the overall solution. We discuss a systematic approach in the next section.

4.3 Choosing the prediction horizon

Before we can fully discuss how to choose the prediction horizon, we must define some new concepts we use in the calculation. We start by differentiating between *active* and *passive biomass.* Active biomass P_act describes the enzymes that are actively catalyzing any reactions under the given environment. Additionally, we regard all quota components as active biomass as they are necessary for the overall function of the cell. The passive biomass P_pas collects all biomass components not part of the active biomass, meaning storage and enzymes not utilized in the current environment. We use these to define the possible growth phases in a deFBA solution, where we set $t_0 = 0$ for easier reading.

- **Balanced growth** While we already presented the concept of balanced growth (2.1), we repeat the definition here for sake of completeness. During this growth mode, the biomass composition stays constant, meaning the percentage share of the individual components stays the same. The biomass increases with a fixed exponential growth rate $\mu \in \mathbb{R}_{\geq 0}$. The mathematical definition is given as

$$\begin{aligned} \dot{P}(t) = \mu P(t), \quad \Rightarrow & P(t) = P(0)e^{\mu t} \\ & B(t) = B(0)e^{\mu t}. \end{aligned} \tag{4.5}$$

While this growth mode is suitable for static methods, like RBA, in a dynamic setting this growth mode is only achieved if the initial biomass composition $P(0)$ is suited for the current nutrient situation. Any changes in the composition of the environment will make it impossible to achieve this growth mode again.

- **Generalized balanced growth** While the idea is identical to the classical balanced growth, we differentiate between active and passive biomass components in this definition. The growth rate is termed μ_{bal} in this case and the dynamics under generalized balanced growth read

  $$\begin{aligned} \dot{P}_{\text{act}}(t) = \mu_{\text{bal}} P_{\text{act}}(t), \ \Rightarrow \ & P_{\text{act}}(t) = P_{\text{act}}(0)e^{\mu_{\text{bal}} t}, \\ \dot{P}_{\text{pas}}(t) = 0, \ \Rightarrow \ & P_{\text{pas}}(t) = P_{\text{pas}}(0). \end{aligned} \tag{4.6}$$

  The objective biomass still increases exponentially

  $$B_{\text{o}}(t) = B_{\text{o,act}}(t)e^{\mu_{\text{bal}} t} + B_{\text{o,pas}}(0). \tag{4.7}$$

  We will refer to this growth mode simply as an *exponential phase.*

- **Linear Growth** During a linear growth phase the cell invest only in a single or multiple biomass components which have a good short-term yield. As this growth mode usually does not increase all components necessary for exponential growth, these phases even harm the long-term goal of biomass increase. Mathematically, we define this via the objective biomass with the constant linear growth rate $\lambda > 0$

  $$\dot{B}_{\text{o}}(t) = \lambda, \ \Rightarrow \ B_{\text{o}}(t) = \lambda t + B_{\text{o}}(0). \tag{4.8}$$

- **Adaptation phases** In these phases the growth rate $\nu(t) \geq 0$ will be varying with time as the cell gradually shifts resources to the production of a new set of enzyme.

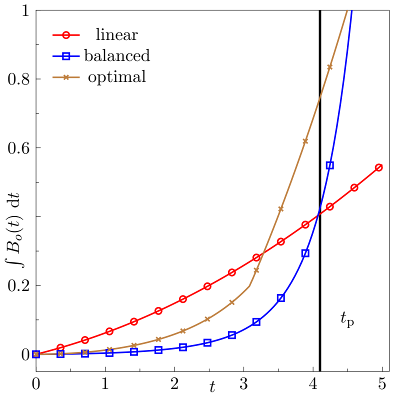
  $$\dot{B}_{\text{o}}(t) = \nu(t) B_{\text{o}}(t) \tag{4.9}$$

  These phases take place if a nutrient source runs out or a new one becomes available. An example is presented in Section 4.6.2.

- **Starvation or zero-growth** During starvation the biomass stays either constant or the system even experiences a loss thereof. As the presence of maintenance reactions, storage capabilities and the capabilities to deconstruct biomass components back into metabolites can be hugely varying for model types, we can not specify the existence of the phases further. In most cases prolonged starvation will lead to infeasible optimization problems in deFBA as maintenance constraints might not be satiable anymore, which coincides with cell death.

Experience has shown that solutions for regular deFBA problems are concatenations of these phases and we use this to determine the prediction horizon. The objective is to choose t_p sufficiently large to avoid having linear growth phases as we regard them merely as mathematical artifacts from the optimization. A simple way to ensure a suitable value is sketched in Figure 4.3. This plot shows the development of a linear solution in comparison to a balanced growth solution and an optimal one. While the linear solution with the linear rate λ clearly grows faster on the short run, any balanced solution in the discussed form overtakes the linear solution given enough time. We have shown that the form of an optimal solution depends on the chosen end-time. But due the optimality these must reach a higher - or at least identical - objective value compared to balanced or linear solutions. Following this idea, we choose the prediction horizon after the time where the linear and balanced curves meet. When using an upper bound on linear growth and a lower bound on balanced growth, the chosen prediction horizon must lead to a solution which grows at least partially exponentially (c.f. Figure 4.3).

The value for t_p depends on the availability of nutrients. In cases where a nutrient source depletes and the system is forced to enter an adaptation phase we are unable to apply this simplified approach as we can not predict values for $\nu(t)$ (4.9) a priori. To get rid of this problem during the determination of a suitable magnitude of the prediction horizon, we make the following assumption.



.

Figure 4.3: Illustration for choosing the prediction horizon. Upper bound on linear growth shown in red (○), balanced growth in blue (□), and optimal solution in brown (x).

Assumption 1. *For the determination of the prediction horizon we assume all external species $Y_{\text{init}} = Y(t_0)$ available at the initial time will stay available for all times.*

This assumption allows us to omit nutrient dynamics for the calculation of the prediction horizon. Instead, we enforce uptake reactions to zero if their nutrients are not present via

$$v_{\max,i}(Y_{\text{init}}) = \begin{cases} 0, & \text{if } Y_i(t_0) = 0 \ \wedge \ v_i \text{ is an uptake reaction} \\ \infty, & \text{else.} \end{cases}, \ \forall i \in \mathcal{Y}. \tag{4.10}$$

In the same way, we must enforce reversible secretion reactions to be positive

$$v_{\min,i}(Y_{\text{init}}) = \begin{cases} 0, & \text{if } Y_i(t_0) = 0 \ \wedge \ v_i \text{ is a seceretion reaction} \\ -\infty, & \text{else.} \end{cases}, \ \forall i \in \mathcal{Y}. \tag{4.11}$$

For a shorter notation we introduce the *initial biomass amount* as

$$B_{\text{init}} = w^T P(0) \tag{4.12}$$

and the *initial objective biomass* as

$$B_{\text{o,init}} = w_{\text{o}}^T P(0). \tag{4.13}$$

The linear growth rate $\lambda \in \mathbb{R}_{\geq 0}$ is calculated as

$$\lambda = w_{\text{o}}^T \frac{\mathrm{d}P_{\text{init}}}{\mathrm{d}t} = w_{\text{o}}^T S_{\mathcal{P}} V_{\text{lin}}, \tag{4.14}$$

with the constant fluxes $V_{\text{lin}} \in \mathbb{R}^m$. Please note, that λ is calculated with the objective weights, while B_{init} presents the total biomass.

We can now calculate an upper bound on the linear growth rate by solving the optimization problem

$$\begin{aligned}
\lambda_{\text{s}}(B_{\text{init}}, Y_{\text{init}}) = \max_{V_{\text{lin}}, P_{\text{lin}}} \quad & w_{\text{o}}^T S_{\mathcal{P}} V_{\text{lin}} && \text{(4.15a)} \\
\text{s.t.} \quad & S_{\mathcal{X}} V_{\text{lin}} = 0 && \text{(4.15b)} \\
& H_{a,\mathcal{P}} P_{\text{lin}} \geq H_g V_{\text{lin}} && \text{(4.15c)} \\
& H_{b,\mathcal{P}} P_{\text{lin}} \leq 0 && \text{(4.15d)} \\
& H_c V_{\text{lin}} - H_{f,\mathcal{P}} P_{\text{lin}} \leq 0 && \text{(4.15e)} \\
& w^T P_{\text{lin}} = B_{\text{init}} && \text{(4.15f)} \\
& v_{\min}(Y_{\text{init}}) \leq V_{\text{lin}} \leq v_{\max}(Y_{\text{init}}), && \text{(4.15g)}
\end{aligned}$$

with $P_{\text{lin}} \in \mathbb{R}^{n_p}_{\geq 0}$. Comparing this to a static deFBA, we have three major changes: The objective is different as we want to optimize the slope of the objective trajectory as shown in Figure 4.3. Furthermore, we omit the dynamics for external species as explained by Assumption 1 and use the new box constraints (4.15g). Finally, we only constrain the biomass via (4.15d) and (4.15f).

The growth phase determined by the *specific linear growth rate* λ_s is constructed as

$$P(t) = P_\mathrm{init} + S_\mathcal{P} V_\mathrm{lin} t. \tag{4.16}$$

The fluxes V_lin can usually not be used to construct feasible trajectories for deFBA problems. The resulting trajectories will violate the biomass composition constraints and/or the maintenance constraints as the fluxes do not scale with the increase in biomass. Still it is beneficial to include the constraint (4.4f) in the calculation as it lowers the value of $\lambda_\mathrm{s}(B_\mathrm{init}, Y_\mathrm{init})$. The specific linear growth rate λ_s depends on the initial biomass amount. Due to the way B_init enters the optimization problem (4.15f) it acts merely as a multiplier for the growth rate

$$\lambda_\mathrm{s}(B_\mathrm{init}, Y_\mathrm{init}) = |B_\mathrm{init}| \lambda_\mathrm{s}(1, Y_\mathrm{init}), \tag{4.17}$$

with the unitless multiplier $|B_\mathrm{init}| = B_\mathrm{init}\ \mathrm{g}^{-1}$.

Hence, we use the *regularized linear rate*

$$\lambda_\mathrm{r}(Y_\mathrm{init}) = \frac{\lambda_\mathrm{s}(B_\mathrm{init}, Y_\mathrm{init})}{B_\mathrm{init}}. \tag{4.18}$$

While the initial amount of biomass is enforced in the constraint (4.15f), the composition of P_init is part of the optimization. This emulates the possibility that the cell might adapt rather quickly to the linear growth mode and ensures the quality of λ_r as a rigorous upper bound.

We still miss a suitable balanced growth rate μ_bal to determine the prediction horizon. At this point we will use the regular balanced growth definition from (4.5), but refine this in Section 4.3.1. A simple approach for finding the growth rate is realized with an adapted form of the RBA [39]. As we are interested in any balanced solution, we fix the initial biomass composition and simplify the dynamics to

$$\frac{\mathrm{d}}{\mathrm{d}t} P_\mathrm{init} = \mu_\mathrm{bal} P_\mathrm{init}. \tag{4.19}$$

The balanced growth rate μ_{bal} is then determined via the optimization problem

$$\mu_{\text{bal}}(P_{\text{init}}, Y_{\text{init}}) = \max_{V_{\text{bal}},\mu} \mu \tag{4.20a}$$

$$\text{s.t.} \quad S_{\mathcal{P}} V_{\text{bal}} = \mu P_{\text{init}} \tag{4.20b}$$

$$S_{\mathcal{X}} V_{\text{bal}} = 0 \tag{4.20c}$$

$$H_a P_{\text{init}} \geq H_g V_{\text{bal}} \tag{4.20d}$$

$$H_c V_{\text{bal}} - H_{f,\mathcal{P}} P_{\text{init}} \leq 0 \tag{4.20e}$$

$$v_{\min}(Y_{\text{init}}) \leq V_{\text{bal}} \leq v_{\max}(Y_{\text{init}}). \tag{4.20f}$$

We assume the initial biomass composition P_{init} satisfies the biomass composition constraint (4.4f). As with the previous problem, we rely on Assumption 1 and substitute any nutrient dynamics with the fixed upper bound on the uptake fluxes via $v_{\max}(Y_{\text{init}})$. As the biomass is fixed, we can directly solve for μ_{bal} without the necessity for a binary search as this problem was designed with minimal computational cost in mind.

Another benefit from this simple approach is, the possibility to derive feasible, yet sub-optimal, trajectories from the results

$$P(t) = P_{\text{init}} e^{\mu_{\text{bal}} t}, \tag{4.21}$$

$$V(t) = V_{\text{bal}} e^{\mu_{\text{bal}} t}, \tag{4.22}$$

if P_{init} satisfies (4.4f) and Assumption 1 holds.

Unlike the linear rate $\lambda_{\text{r}}(Y_{\text{init}})$, the growth rate $\mu_{\text{bal}}(P_{\text{init}}, Y_{\text{init}})$ is highly dependent on the biomass composition P_{init}. Hence, if the biomass composition changes this rate needs to be recalculated. Depending on the amount of passive biomass this might be necessary after each iteration step in the sdeFBA.

With the exponential and the linear growth rate, we can calculate the values of the objective functional as shown in Figure 4.3. For the balanced growth solution the objective is given as

$$J_{\text{bal}}(t, \mu_{\text{bal}}(Y_{\text{init}}, P_{\text{init}})) = \int_0^t w_{\text{o}}^T P(\tau) \, \mathrm{d}\tau \tag{4.23}$$

$$= \frac{B_{\text{o,init}}}{\mu_{\text{bal}}} (e^{\mu_{\text{bal}} t} - 1). \tag{4.24}$$

and for the hypothetical value for the linear solution is

$$J_{\text{lin}}(t, \lambda_{\text{r}}(Y_{\text{init}}), B_{\text{init}}) = \int_0^t B_{\text{o,init}} + \lambda_{\text{s}}(B_{\text{init}}, Y_{\text{init}})\tau \, \mathrm{d}\tau \tag{4.25}$$

$$= \frac{\lambda_{\text{r}} B_{\text{o,init}}}{2} t^2 + B_{\text{o,init}} t. \tag{4.26}$$

We set the prediction horizon at the time where these objectives meet by solving

$$J_{\mathrm{bal}}(t_{\mathrm{p}}, \mu_{\mathrm{bal}}(Y_{\mathrm{init}}, P_{\mathrm{init}})) - J_{\mathrm{lin}}(t_{\mathrm{p}}, \lambda_{\mathrm{r}}(Y_{\mathrm{init}}), B_{\mathrm{init}}) = 0 \tag{4.27}$$

for $t_{\mathrm{p}} \geq 0$.

By looking at the slopes of the biomass curves at time zero, we can deduce that $t_{\mathrm{p}} > 0$ exists if, and only if, $\lambda_{\mathrm{r}} > \mu_{\mathrm{bal}}$. Otherwise, the model favors exponential solutions even on short time scales and t_{p} can not be determined via this method. For the rest of the chapter, we make the following assumption.

Assumption 2. *The linear growth rate is larger than the balanced growth rate* $\lambda_{\mathrm{r}} > \mu_{\mathrm{bal}}$.

An optimal solution of (4.4) on $[0, t_{\mathrm{p}}]$ can only produce an objective value equal or larger than $J_{\mathrm{bal}}(t_{\mathrm{p}})$, otherwise it would contradict the optimality principle. Hence, we conclude that any optimal solution must contain a super-linear (typically exponential) arc as shown in Figure 4.3. The length of this arc will be the topic of Section 4.4.

4.3.1 Improving on the boundaries for the growth rates

The presented method so far highly depends on the current environment Y_{init} and biomass composition P_{init}. The horizon should therefore be recalculated after every iteration step. This can become quite costly, especially if we are dealing with ill-posed problems in (4.20a) or (4.15). Therefore, the presented optimization problems are simplified as far as possible. A way to improve the quality of the prediction horizon is by finding an exponential growth rate instead of a balanced growth rate. The exponential rates are larger than the balanced one, especially if the system has just undergone a transition and a lot of passive biomass is present. Therefore, we must differentiate between active and passive biomass in the problem formulation to be able to predict a smaller prediction horizon. Unfortunately, identifying the passive biomass leads to a mixed-integer optimization problem, which is numerically more demanding.

To construct this problem we use the definition for generalized exponential growth as presented in the previous section and implement this into (4.20a). We introduce a integer decision vector $g \in \{0, 1\}^{n_p}$ and define a component wise vector multiplication as $c = a \odot b$, with $c_i = a_i b_i$. If the value of g_i is 1, the component P_i is regarded as an active biomass component. The nonlinear mixed-integer program then reads:

$$
\begin{aligned}
\mu_{\text{exp}}(P_{\text{init}}, Y_{\text{init}}) = \max_{V_{\text{bal}},\mu,g}\ & \mu && \text{(4.28a)}\\
\text{s.t.}\ & S_{\mathcal{P}} V_{\text{bal}} = \mu g \odot P_{\text{init}} && \text{(4.28b)}\\
& S_{\mathcal{X}} V_{\text{bal}} = 0 && \text{(4.28c)}\\
& H_a P_{\text{init}} \geq H_g V_{\text{bal}} && \text{(4.28d)}\\
& H_c V_{\text{bal}} - H_{f,\mathcal{P}} P_{\text{init}} \leq 0 && \text{(4.28e)}\\
& g_{\text{quota}} = 1 && \text{(4.28f)}\\
& v_{\min}(Y_{\text{init}}) \leq V_{\text{bal}} \leq v_{\max}(Y_{\text{init}}), && \text{(4.28g)}
\end{aligned}
$$

with the new constraint (4.28f) expressing that quota components are always regarded as active biomass. If P_i has a biomass composition constraint attached the value of g_i must be 1. The new formulation in (4.28b) proclaims that only the active biomass components are to be produced. This problem is nonlinear as (4.28b) contains a product of μ and g We can again use a binary search to eliminate this.

The resulting optimization is called *mixed integer linear program* (MILP) or in this special case a *mixed zero-one linear program*. We are not going into detail in solving these, but refer exemplary to a branch-and-bound method as described in [71]. As MILPs are already NP-hard and we must include the binary-search for μ. This means the new approach is computationally very costly, as we have to repeatedly solve the MILP after every sdeFBA iteration. Therefore, it is left to the user to decide whether the possible decrease in the calculated prediction horizon is worth the increased computational cost to determine μ_{exp}.

Some characteristics of the solution for (4.28a) can be stated without solving the problem. If P_{init} was calculated with an RBA, meaning P_{init} is optimal for balanced growth, the solutions of both methods are identical $\mu_{\text{exp}}(P_{\text{init}}, Y_{\text{init}}) = \mu_{\text{bal}}(P_{\text{init}}, Y_{\text{init}})$. Otherwise, $\mu_{\text{exp}}(P_{\text{init}}, Y_{\text{init}}) \geq \mu_{\text{bal}}(P_{\text{init}}, Y_{\text{init}})$ holds true.

Furthermore, we learn which components of P_{init} can be regarded as active P_{act} by looking at the results for the decision variable g^*. We can exploit this to improve on the estimate for the linear growth rate, by using only the active biomass to determine the specific growth rate $\lambda_{\text{act}} = \lambda_{\text{r}} B_{\text{act}}$, with $B_{\text{act}} = w^T g^* \odot P_{\text{init}} \leq B_{\text{init}}$. Therefore, the value for the prediction horizon calculated with μ_{exp} and λ_{act} must also be smaller. But the problem remains, whether the additional cost of solving the complex MILP after each iteration pays out.

4.4 Choosing the iteration time

So far, we have only determined a prediction horizon guaranteeing the existence of a superlinear arc in the biomass trajectory. We have not proven that the initial part of the trajectory is of this form. Therefore, it might be possible that an optimal solution starts with a linear arc and finishes with the exponential one. We will proof the correct order of the different arcs and determine the length of the exponential one, which we will use to determine the iteration time.

We start by showing that an optimal trajectory starting with a single linear arc stays in this growth mode. We construct a hypothetical solution starting in a linear growth phase and use balanced growth after the *switching time* $t_\mathrm{s} \in [0, t_\mathrm{p}]$. The objective biomass can then be described as

$$B_\mathrm{mix}(t) = \begin{cases} B_\mathrm{o,init}\lambda_\mathrm{r} t + B_\mathrm{o,init} & 0 \le t \le t_\mathrm{s} \\ B_\mathrm{o,init}(\lambda_\mathrm{r} t_\mathrm{s} + e^{\mu_\mathrm{bal}(t - t_\mathrm{s})}) & t_\mathrm{s} < t \le t_\mathrm{p}, \end{cases} \tag{4.29}$$

with the balanced growth rate μ_bal. In fact, the following calculations can be done with any rate μ as long as $\mu \le \lambda_\mathrm{r}$.

Theorem 1. *If Assumption 2 holds, any optimal solution curve* B_mix (4.29) *consists only of a single linear phase with* $t_\mathrm{s} = t_\mathrm{p}$.

Proof. We identify the optimal switching time by solving

$$\max_{t_\mathrm{s}} \int_0^{t_\mathrm{p}} B_\mathrm{mix}(t)\ \mathrm{d}t \tag{4.30}$$

analytically by finding local extrema via the first order derivative with respect to t_s

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}t_\mathrm{s}} \int_0^{t_\mathrm{p}} B_\mathrm{mix}(t)\ \mathrm{d}t \\ &= B_\mathrm{o,init}(\lambda_\mathrm{r}(t_\mathrm{p} - t_\mathrm{s}) + 1 - e^{\mu_\mathrm{bal}(t_\mathrm{p} - t_\mathrm{s})}), \end{aligned} \tag{4.31}$$

with the obvious zero $\bar{t}_\mathrm{s} = t_\mathrm{p}$. Evaluating the second derivative at this point gives

$$\frac{\mathrm{d}^2}{\mathrm{d}t_\mathrm{s}^2} \int_0^{t_\mathrm{p}} B_\mathrm{mix}(t)\ \mathrm{d}t \bigg|_{\bar{t}_\mathrm{s}} = B_\mathrm{o,init}(\mu_\mathrm{bal} - \lambda_\mathrm{r}) < 0, \tag{4.32}$$

with the last inequality following Assumption 2. Hence, $\bar{t}_\mathrm{s} = t_\mathrm{p}$ is a local maximum and any solution of the form B_mix does not include an exponential arc. For the sake of completeness, we must also mention that there exists another zero of (4.31) $\bar{t}_\mathrm{s,2} \in [0, t_\mathrm{p})$, which cannot be given in closed form. But, due to continuity and the intermediate value theorem, $\bar{t}_\mathrm{s,2}$ is a local minimum of (4.30). Hence, it presents the worst-case switching time. □

Due to the way we have chosen t_p, an optimal solution to (4.4) on $[0, t_\mathrm{p}]$ can not start with a linear phase as this would be suboptimal. Hence, the Theorem 1 implicitly states that an optimal solution must start with a super-linear phase, which grows at least as fast as the balanced growth curve. We formalize this by constructing a solution

as

$$B_{\text{opt}}(t) = \begin{cases} B_{\text{o,init}} e^{\mu_{\text{bal}} t}, & 0 \leq t \leq t_{\text{s}}, \\ B_{\text{o,init}} e^{\mu_{\text{bal}} t_{\text{s}}} (\lambda_{\text{r}}(t - t_{\text{s}}) + 1), & t_{\text{s}} < t \leq t_{\text{p}}. \end{cases} \tag{4.33}$$

Here, we allow a balanced growth phase to be followed up by a linear one. We have seen this behavior in Section 4.1.1. The question we want to answer with this is simple: At what time t_{s} does the solution switch to linear growth?

Theorem 2. *If Assumption 2 holds, an optimal solution* B_{opt} (4.33) *of the sdeFBA* (4.4) *is growing exponentially on the time frame* $[0, t_{\text{i}})$, *with*

$$0 < t_{\text{i}} < t_{\text{p}} - 2\left(\frac{1}{\mu_{\text{bal}}} - \frac{1}{\lambda_{\text{r}}}\right). \tag{4.34}$$

Proof. As in the previous proof, we identify the optimal switching time t_{s} by solving the optimization problem

$$\max_{t_{\text{s}}} \int_0^{t_{\text{p}}} B_{\text{opt}}(t) \ \mathrm{d}t. \tag{4.35}$$

The zeros of the first order derivative are given by

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t_{\text{s}}} \int_0^{t_{\text{p}}} B_{\text{opt}}(t) \ \mathrm{d}t = 0 \\ \Leftrightarrow \lambda_{\text{r}} \mu_{\text{bal}} t_{\text{s}}^2 + 2\left(\lambda_{\text{r}} - \mu_{\text{bal}} - \lambda_{\text{r}} \mu_{\text{bal}} t_{\text{end}}\right) t_{\text{s}} + \lambda_{\text{r}} \mu_{\text{bal}} t_{\text{end}}^2 + 2(\mu_{\text{bal}} - \lambda_{\text{r}}) t_{\text{end}} = 0 \end{aligned} \tag{4.36}$$

$$\Rightarrow \ \hat{t}_{\text{s},1} = t_{\text{p}} - 2\left(\frac{1}{\mu_{\text{bal}}} - \frac{1}{\lambda_{\text{r}}}\right), \ \hat{t}_{\text{s},2} = t_{\text{p}}. \tag{4.37}$$

Please note, that the extreme values only depend on the interplay between growth rates, switching time and end-time. To determine whether the extrema are maxima or minima we calculate the second-order derivatives as

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}t_{\text{s}}^2} \int_0^{t_{\text{p}}} B_{\text{opt}}(t) \ \mathrm{d}t =& e^{\mu_{\text{bal}} t_{\text{s}}} \left(\frac{\lambda_{\text{r}} \mu_{\text{bal}}^2}{2} t_{\text{s}}^2 + (2\lambda_{\text{r}} \mu_{\text{bal}} - -\mu_{\text{bal}}^2 B_{\text{o,init}} - \lambda_{\text{r}} \mu_{\text{bal}}^2 t_{\text{p}}) t_{\text{s}} \right. \\ & \left. + \frac{\mu_{\text{bal}}^2 \lambda_{\text{r}} t_{\text{p}}^2}{2} + \mu_{\text{bal}}^2 B_{\text{o,init}} t_{\text{p}} - 2\lambda_{\text{r}} \mu_{\text{bal}} t_{\text{p}} + \lambda_{\text{r}} - \mu_{\text{bal}} B_{\text{o,init}} \right). \end{aligned} \tag{4.38}$$

Evaluating these at the extrema gives

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}t_{\text{s}}^2} \int_0^{t_{\text{p}}} B_{\text{opt}}(t) \ \mathrm{d}t \bigg|_{\hat{t}_{\text{s},1}} &= (\mu_{\text{bal}} - \lambda_{\text{r}}) B_{\text{init}} e^{\mu_{\text{bal}} t_{\text{p}}} < 0, \\ \frac{\mathrm{d}^2}{\mathrm{d}t_{\text{s}}^2} \int_0^{t_{\text{p}}} B_{\text{opt}}(t) \ \mathrm{d}t \bigg|_{\hat{t}_{\text{s},2}} &= (\lambda_{\text{r}} - \mu_{\text{bal}}) B_{\text{init}} e^{\mu_{\text{bal}} t_{\text{p}}} > 0. \end{aligned} \tag{4.39}$$

Hence, $\hat{t}_{\mathrm{s},1}$ maximizes (4.35) and the solution is of exponential form until $\hat{t}_{\mathrm{s},1}$. □

We recommend choosing the iteration time t_{i} a bit smaller than given in (4.34) to compensate for possible numerical inaccuracies during the simulation. As (4.34) depends on the growth rates, the iteration time should also be recalculated together with the prediction horizon. The interplay between these values is very important and should always be checked if sdeFBA solution yield unexpected results.

4.5 Short-term deFBA in `deFBA-Python`

The sdeFBA is fully implemented in the `deFBA-Python` package [64] using the collocation method provided by the `LinOpt` package. Instead of presenting the full collocation, we explain the implementation of the sdeFBA by simple time discretization and substitution of the dynamics (4.4c) with a forward Euler scheme. We do not recommend using such simple approximations to the dynamics, but present these here for easier readability. A simple collocation scheme applicable to the deFBA problem class is presented in Section 5.5. The discretization uses an equidistant time grid

$$\Delta_t(h) = \{t_k = kh + t_0\}, \tag{4.40}$$

with step size $h \in \mathbb{R}_{>0}$. States and reactions are approximated on this time grid as $Z_k \approx Z(t_k)$, $V_k \approx V(t_k)$. To clarify the connections between initial values and the resulting predictions during an sdeFBA run we extend this notation to

$$V_{k|i} \approx V(t_k),\ Z_{k|i} \approx Z(t_k),\ k \geq i. \tag{4.41}$$

This means the approximation of derivatives and states at time t_k, with given Z_i as initial values for the current simulation. The *discrete prediction horizon* p and the *discrete iteration time* q are calculated as

$$p = \left\lceil \frac{t_{\mathrm{p}}}{h} \right\rceil,\ q = \left\lfloor \frac{t_{\mathrm{i}}}{h} \right\rfloor, \tag{4.42}$$

with the *ceiling function* $\lceil \cdot \rceil$ and the *floor function* $\lfloor \cdot \rfloor$. To give an idea how the sdeFBA is implemented, we present the algorithm as pseudo-code.

```
procedure DefbaModel.shortterm($Y0, P0, t_{\mathrm{end}}, h, r, p$)
    Initialize empty solution trajectories $Z^*$, $V^*$
    $i := 0$
    $Z^*_0 := [Y0, P0]$
    $t := 0$
    calculate $p, q$ via DefbaModel.calcPredictionHorizon($Z^*_0$)
    while $t < t_{\mathrm{end}}$ do
        $\mathcal{K} := \{i+1, \dots, i+p\}$
```

```
            solve:
```

$$
\begin{aligned}
& \max_{(V_{k|i}),\ k\in\mathcal{K}} \sum_{k\in\mathcal{K}} B_{k|i} && \text{(4.43a)}\\
\text{with given}\quad & Z_{i|i} = Z_i^*, && \text{(4.43b)}\\
\text{subject to:}\quad & Z_{k|i} = Z_{k-1|i} + hSV_{k-1|i} && \text{(4.43c)}\\
& S_{\mathcal{X}}V_{k|i} = 0 && \text{(4.43d)}\\
& H_aZ_{k|i} - H_fV_{k|i} \leq 0 && \text{(4.43e)}\\
& H_bZ_{k|i} \leq 0 && \text{(4.43f)}\\
& H_cV_{k|i} - H_eZ_{k|i} \leq 0 && \text{(4.43g)}\\
& H_dV_{k|i} \leq C_{k|i} && \text{(4.43h)}\\
& Z_{k|i} \geq 0 && \text{(4.43i)}\\
& V_{\min} \leq V_{k|i} \leq V_{\max}, \forall k \in \mathcal{K} && \text{(4.43j)}
\end{aligned}
$$

```
            for $k \in \{i, \dots, i+q-1\}$ do
                $Z^*_{k+1} := [Y_{k+1|i}, P_{i+1|i}]$
                $V^*_k := V_{k|i}$
                $i := i+1$
                $t := t+h$
                Additional break conditions can be placed here
            end for
            update $p, q$ via DefbaModel.calcPredictionHorizon($Z^*_i$)
        end while
        return $Z^*, V^*$
    end procedure
```

The pseudo-code only represents the basic functions of the method. Of course, we included the same comfort functions as we did with the regular deFBA. This includes automatic calculation of initial biomass via an RBA, detection of initial values from the SBML file containing the model, etc. Additionally, we allow the user to specify whether the horizons $t_\mathrm{p}, t_\mathrm{i}$ should be recalculated after each step or to simply use given values.

The method `DefbaModel.calcPredictionHorizon`() contains the presented way to calculate the time horizons. It needs the current biomass composition as parameter and can optionally include the current environment as explained before. If no environment is defined, the algorithm assumes all possible nutrients are available. To this point, the implementation does not support the more accurate version using the mixed integer program (cf. Section 4.3.1).

4.6 Numerical examples

The following numerical examples were solved with the `DefbaModel.shortterm()` method using a Radau collocation. They are also included in the web release [1] of `deFBA-Python` [64] as examples.

4.6.1 Enzymatic growth model revisited

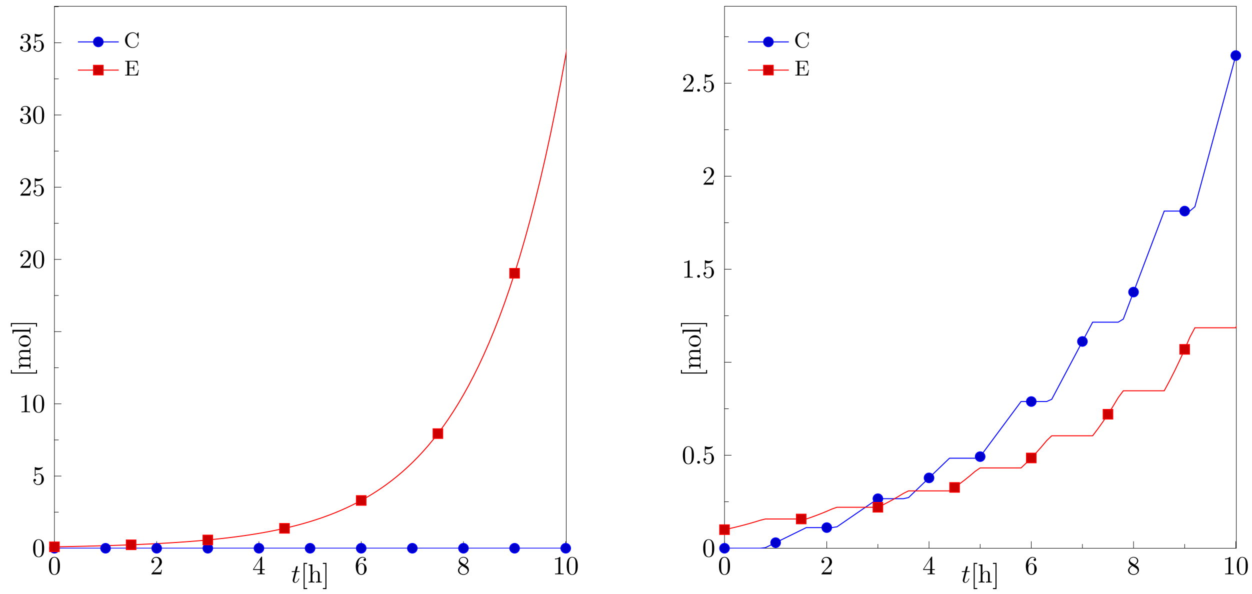


Figure 4.4: Results for the enzymatic growth model. For the simulation we choose initial values as E_0=C_0=1 mol. Left: Overall sdeFBA solution with $t_\mathrm{p} = 3.25$ h, $t_\mathrm{i} = 1.45$ h. Right: Faulty solution with $t_\mathrm{p} = 2.5$ h, $t_\mathrm{i} = 1.5$ h. With the chosen iteration time each part of the solution consists of an exponential phase, followed by a short linear one.

With the short-term deFBA fully explained, we want to revisit the *enzymatic growth* model from Section 4.1.1. We had seen in Figure 4.1 (left) that the regular deFBA produces a linear growth section near the end-time t_p. The sdeFBA on the other hand stays in a exponential growth phase all the time as shown in Figure 4.4 (left). In this simulation we started with the identical initial values E_0=C_0=1 mol and used the identical step size $d = 0.1$ h. For the prediction horizon and iteration time, we followed the described procedure and calculated $t_\mathrm{p} \approx 3.25$ h and $t_\mathrm{i} = 1.45$ h. This was translated to 3.3 h as the discrete prediction horizon and 1.4 h for the discrete iteration time.

To showcase the effects of using faulty horizon values, we did another simulation with $t_\mathrm{p} = 2.5$ h and $t_\mathrm{i} = 1.5$ h. The results are shown in Figure 4.4 (right). Here every sub-solution on the time grid $\Delta_t(t_\mathrm{i})$ is of the form we have seen with the regular

---

[1] https://bitbucket.org/hlindhor/defba-python-package

deFBA: Each exponential arc is followed by a linear part. This showcases that a smaller prediction horizon might still produce an exponential growth phase in the beginning, but combining this horizon with an over sized iteration time might prove fatal for the quality of the solution.

4.6.2 Carbon-core model

Table 4.2: Exchange and metabolic reactions in the carbon-core model. Catalytic constants k_{cat} are true for both directions for reversible reactions. Enzyme coloumn describes which biomass species is needed to catalyze the reaction.

Reaction stoichiometry	Enzyme	$k_{\text{cat}}/\text{min}^{-1}$
Exchange reactions		
$V_{y,\text{C1}}$: Carb1 $\rightarrow$ A	T_{C1}	3000
$V_{y,\text{C1}}$: Carb2 $\rightarrow$ A	T_{C2}	2000
$V_{y,\text{F}}$: $\text{F}_{\text{ext}} \rightarrow \text{F}$	T_{F}	3000
$V_{y,\text{H}}$: $\text{H}_{\text{ext}} \rightarrow \text{H}$	T_{H}	3000
$V_{y,\text{O2}}$: $\text{O2}_{\text{ext}} \rightarrow \text{O2}$	S	1000
$V_{y,\text{D}}$: $\text{D}_{\text{ext}} \leftrightarrow \text{D}$	S	1000
$V_{y,\text{E}}$: $\text{E}_{\text{ext}} \leftrightarrow \text{E}$	S	1000
Metabolic reactions		
$V_{x,1}$: A + ATP $\rightarrow$ B	E_{B}	1800
$V_{x,2}$: B $\rightarrow$ C + 2 ATP + 2 NADH	E_{C}	1800
$V_{x,3}$: C $\leftrightarrow$ 2ATP + 3D	E_{D}	1800
$V_{x,4}$: C + 4NADH $\leftrightarrow$ 3E	E_{E}	1800
$V_{x,5}$: B $\rightarrow$ F	E_{F}	1800
$V_{x,6}$: C $\rightarrow$ G	E_{G}	1800
$V_{x,7}$: G + ATP + 2NADH $\leftrightarrow$ H	E_{H}	1800
$V_{x,8}$: G $\rightarrow$ 0.8C + 2NADH	E_{N}	1800
$V_{x,9}$: O2 + NADH $\rightarrow$ ATP	E_{T}	1800

In this section we present another academic model, which was already used in [110] to present the features of the deFBA. We further verify the short-term deFBA by comparing their results for this model with the original ones using the standard deFBA. We call it the *carbon-core* model and its exchange and metabolic reactions are presented in Table 4.2. Additionally, this table shows which enzyme catalyzes the respective reaction and the used turnover numbers. The reversibility of each reaction is depicted

Table 4.3: Biomass producing reactions in carbon-core model. All reactions are catalyzed irreversible by the ribosome R. Molecular weights w relate to the products of the reactions. Initial biomass compositions are given by the rows P_0 and P_{switch}.

	Biomass reaction	$w/\frac{\text{g}}{\text{mol}}$	k/min^{-1}	P_0/mol	$P_{\text{switch}}/\text{mol}$
$V_{p,\text{T}_{\text{C1}}}$:	400H + 1600 ATP → T_{C1}	4	2.5	2.0474E-05	6.3039E-3
$V_{p,\text{T}_{\text{C2}}}$:	1500H + 6000ATP → T_{C2}	15	0.67	0.0	0.0
$V_{p,\text{T}_{\text{F}}}$:	400H + 1600 ATP → T_{F}	4	2.5	0.0	0.0
$V_{p,\text{T}_{\text{H}}}$:	400H + 1600ATP→ T_{H}	4	2.5	0.0	0.0
$V_{p,\text{E}_{\text{B}}}$:	500H + 2000ATP→ E_{B}	5	2	3.4123E-05	10.5065E-3
$V_{p,\text{E}_{\text{C}}}$:	500H + 2000ATP→ E_{C}	5	2	3.1457E-05	10.0354E-3
$V_{p,\text{E}_{\text{D}}}$:	1000H + 4000ATP→ E_{D}	10	1	1.1967E-05	2.8901E-3
$V_{p,\text{E}_{\text{E}}}$:	1000H + 4000ATP→ E_{E}	10	1	0.0	3.6913E-3
$V_{p,\text{E}_{\text{F}}}$:	2000H + 8000ATP→ E_{F}	20	0.5	2.6665E-06	4.7100E-4
$V_{p,\text{E}_{\text{G}}}$:	500H + 2000ATP→ E_{G}	5	2	1.4722E-05	2.6528E-3
$V_{p,\text{E}_{\text{H}}}$:	4000H + 16000ATP→ E_{H}	40	0.25	1.4723E-05	2.6528E-3
$V_{p,\text{E}_{\text{N}}}$:	500H + 2000ATP→ E_{N}	5	2	0.0	0.0
$V_{p,\text{E}_{\text{T}}}$:	500H + 2000ATP→ E_{T}	5	2	3.3467E-05	0.0
$V_{p,\text{R}}$:	4500H + 1500C + 21000ATP→ R	60	0.2	3.0621E-05	5.4538E-3
$V_{p,\text{S}}$:	250H + 250C + 250F + 1500ATP→ S	7.5	3	2.3333E-04	46.6976E-3

by the choice of arrows in the stoichiometric description. Please note, that we are following the convention that all reversible exchange reactions and irreversible uptake reactions are pointing inwards. The reaction $V_{y,\text{H}}$ is a pure secretion process and is thus chosen to point outwards.

The biomass producing reactions are shown in Table 4.3. All these reactions are catalyzed by the ribosome R with the given catalytic constants. Furthermore, the table depicts the molecular weights for all species.

The model itself is a very reduced take on the central carbon metabolism in single-cell organisms. The exchange with the outside is modeled for oxygen and the fermentation products D, E via the transporter S. The other external species are taken up or secreted via specialized transporters T_{C1}, T_{C1}, T_{F}, T_{H}. The amount of the transporter S is scaling with the biomass. Therefore, we must infer a biomass composition constraint on S, which is chosen as 35% of all biomass in this example.

The turnover numbers for the exchange processes and metabolic reactions are oriented on values typical observed in metabolism [6], [98], [74]. The k_{cat} values for the transcription of the biomass products on the other hand are derived from the measured translation rate of 17 amino acids per second in *E.coli* [115].

Firstly, the cell can decide which carbon source to utilize. The parameters are chosen in a way, that the active transport of Carb1 via the enzyme T_{C1} needs less investment and is more efficient than its counter-part T_{C2}. Furthermore, if the species F_{ext}, H_{ext} are available it is more efficient to invest into their respective transporters in comparison to enabling their biosynthesis via B → F and G → H. Very important for our second example is the utilization of oxygen as the model can grow both anaerobically and aerobically. Under aerobic conditions the cells can completely metabolize the carbon sources due to the cyclic reactions $V_{x,6}$ - $V_{x,8}$. If oxygen is present, the external versions of the fermentation products $\mathrm{D}_{\mathrm{ext}}$, $\mathrm{E}_{\mathrm{ext}}$ can even be taken up to enable further growth. Under anaerobic conditions this is not possible and D, E must be secreted. The model was designed this way to force multiple decisions in resource allocation at once.

We will use this model for two examples. First we will repeat one of the experiments from [110] to prove that the sdeFBA can reproduce results created with the original deFBA and further investigate the interplay between prediction horizon and iteration time during adaptive growth phases. Afterwards, we investigate how the model reacts to sudden shifts in the environment.

4.6.2.1 Carbon source switch

Table 4.4: Initial values used for the carbon source switch experiment given in molar amounts.

Carb1	Carb2	$\mathrm{O2}_{\mathrm{ext}}$	$\mathrm{D}_{\mathrm{ext}}$	$\mathrm{E}_{\mathrm{ext}}$	$\mathrm{F}_{\mathrm{ext}}$	$\mathrm{H}_{\mathrm{ext}}$
2	30	50	0	0	0	0

In this experiments we mimic a batch process in which the nutrients are limited and will deplete. We define the initial state of the environment as the values given in Table 4.4. The initial biomass state is calculated via an RBA using the given environment. They are printed in Table 4.3 column P_0. As Carb1 is easier to import, the RBA (3.28) predicts no investment into the transporter T_{C2} and a high investment into E_{T} to enable the utilization of the oxygen. The amount of the external oxygen is influenced by a fixed inflow V_{O} and a ventilation process leading to the full dynamics as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{O2}_{\mathrm{ext}} = -V_{y,\mathrm{O2}} + V_{\mathrm{O}} - \gamma_{\mathrm{O}}\mathrm{O2}_{\mathrm{ext}}, \tag{4.44}$$

with the ventilation rate γ_{O}. For this scenario we use the values $V_{\mathrm{O}} = 20$ mol min^{-1} and $\gamma_{\mathrm{O}} = 0.4$ min^{-1} We set the end-time for the regular deFBA simulation as $t_{\mathrm{end}} = 70$ min. We recreated the results from [110] with the `deFBA-Python` toolbox and plotted these in Figure 4.5. In the left plot the trajectories for the external dynamics are shown. We marked three major events: At $t = 19$ min the preferred carbon source Carb1 runs out, at t =56 min the second carbon source depletes, and at t =60 min all the fermentation waste D, which was secreted until $t = 45$ min, is used up and the solution enters a starvation phase. As this model does not include any maintenance, starvation means simple zero growth.

But the analysis gets more interesting, when we look at the development of the biomass shown in the right plot of Figure 4.5. The plot shows the biomass distribution, not the actual amount, and we can learn a lot from this depiction. We can see, that the cell stays in balanced growth until $t = 9$ min. Then it starts production of the transporter T_{C2} and starts metabolizing Carb2. This production spikes directly before Carb1 depletes, to keep the growth rate roughly constant. Very interestingly during this adaptation phase the cell refrains from keeping the production of R and reaches a new exponential growth phase starting at $t = 19$ min. This last until 30 min at which time more oxygen is needed to start the internal metabolization of D. We can see this in the increase in the enzyme E_{T} and the increase in oxygen uptake. At t =45 min we reach a linear phase in which all resources are used to produce the structural component S, which has the best biomass yield if carbon is available. Afterwards, when the carbon source depletes and only the fermentation product D is available, the cell focuses on the production of the ribosome as the production does not require the species C and F.

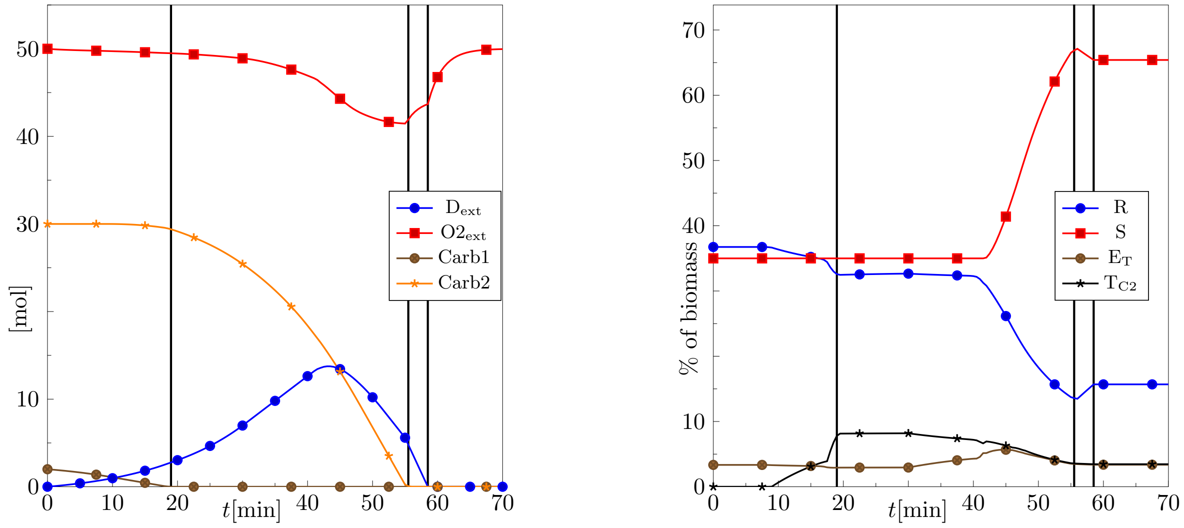


Figure 4.5: Results for regular deFBA model. (Left) These are the trajectories for the external species. Additionally are the times at which a nutrient source depletes marked. (Right) Biomass composition over time. We only picked the relevant macromolecules to clarify when different growth phases start.

If we instead use the short-term deFBA to simulate this model, we get nearly identical results, but the choices in prediction horizon and iteration time can have quite an impact. Firstly, we want to simulate with the horizons determined as described in Section 4.3 and Section 4.4. This results in the series of prediction horizons and iterations times as given in Table 4.5. Using these for the simulation with the sdeFBA yields the results presented in Figure 4.6. While the left side plot showing the external species is nearly identical, we can observe, that the fermentation product D_{ext} takes longer to be fully consumed in comparison to the previous results. Furthermore, the biomass percentages shown on the right are quite jittery as each iterative solution seems to include a phase in which unnecessarily more structure components S are produced. This means either the prediction horizon is chosen too small or the iteration time is too large. In this case, we believe the prediction horizon is chosen to small as Assumption 1 is violated due to the early depletion of Carb1. This means, the cells must adapt to another nutrient source in the considered time frame. Because these investments in T_{C2} in the beginning are expensive, the solution shows an early focus on the production of S. When the system has adapted, between $t = 20$ min and $t = 40$ min, we still get a little bit of false investment into S due to a slightly over-sized iteration time. Using the values shown in Table 4.6 in which we multiplied the calculated values for t_p with 1.5, we can perfectly reproduce the original deFBA results. We refrain from plotting these as the pictures look identical to the ones from Figure 4.5. For completeness, we must also state that using the method for the calculation of t_p in combination with a minimal iteration time of $t_i = 0.5$ min will also reproduce the original results. But in this case, the number of necessary iteration steps increases from roughly 9 to over 120, which makes this numerically too costly.

The possibility to reproduce the deFBA results teaches us two important things about the short-term deFBA. Firstly, the prediction horizon calculation as explained in this chapter gives a good idea on the size of the prediction horizon. But to be able to utilize this method better, we must find a better way to formulate the nutritional constraints $v_{\min}(Y), v_{\max}(Y)$ (cf. (4.10), (4.11)) in dependency on the current biomass, growth rate, and amount of nutrient available.

Secondly, under nutritional stress results from short-term and regular deFBA are identical as both solutions predict an increase in structural molecules S after $t = 45$ min

Table 4.5: Prediction horizons and iteration times used for the simulation shown in Figure 4.6. All times are given in minutes.

actual time	0	4.5	10	16.5	23.5	30.5	36.5	43.5	55
prediction horizon	14.5	16.5	18.5	20.0	17.5	19.5	16.5	29.5	77.0
iteration time	4.5	5.5	6.5	7.0	6.0	7.0	5.5	11.5	37.5

Table 4.6: Prediction horizons and iteration times capable of reproducing the original deFBA results shown in Figure 4.5. Values are calculated by the discussed methods, but prediction horizon is multiplied by a factor of 1.5.

actual time	0.0	4.5	9.0	14.0	19.0	24.0	29.0	34.0	39.0	45.0
prediction horizon	21.5	21.0	21.0	23.5	24.0	23.5	23.5	23.5	26.5	57
iteration time	4.5	4.5	4.5	5.0	5.0	5.0	5.0	5.0	6.0	16.0

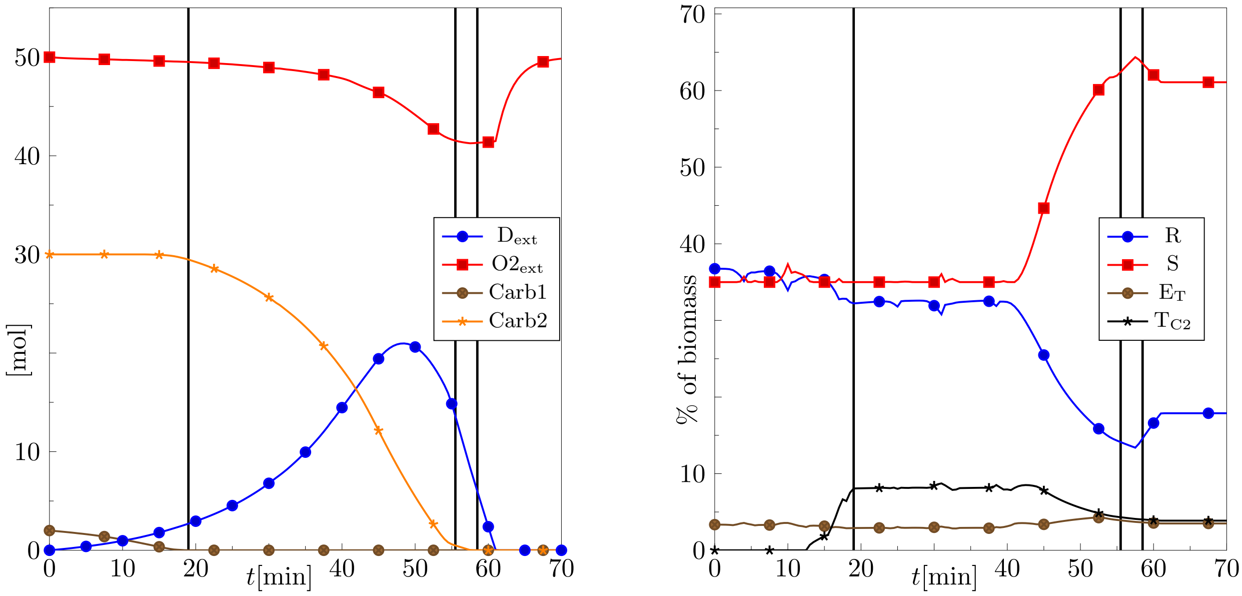


Figure 4.6: Results for short-term deFBA model. (Left) These are the trajectories for the external species. Additionally are the times at which a nutrient source depletes marked. (Right) Biomass composition over time. In comparison to the original deFBA results shown in Figure 4.5 we see mostly identical behavior; but most of the individual iteration times seems a bit long for the respective prediction horizon and we get unnecessary production of S at the end of each iteration step.

to maximize the objective value. Therefore, we suggest using the sdeFBA primarily to conserve computational cost for very large models and/or applications with very large end-times or in situation with plenty nutrients. Solving problems like the one we just presented are more efficiently solved with a single deFBA problem.

But in the next section we present another problem class better suited for the sdeFBA.

4.6.2.2 Aerobic - anaerobic switching

We use the *carbon-core model* for another experiment. Imagine a cell population living in a very dynamic environment in which the conditions can change very rapidly, like near a drainage system, a bioreactor for biogas production, etc. To test whether we can use the short-term deFBA to reliable predict how the population adapts to these sudden changes, we construct another *in silicio* experiment. The environment during this experiment switches from aerobic to anaerobic conditions every 30min. During the aerobic phase we assume oxygen is limitlessly available $\mathrm{O2}_{\mathrm{ext}} = \infty$. Additionally, we set the amount of Carb1 to infinity Carb1$= \infty$ to ensure the carbon dynamics do not interfere with the switching between aerobic and anaerobic growth phases.

The initial biomass composition is calculated via an RBA with $B_0 = w^T P_0 = 1$ g in an anaerobic environment containing only Carb1. The initial amount for the enzymes are depicted in Table 4.7. The initial phase of the experiment takes place in the

Table 4.7: Initial biomass values in molar amounts for the switching experiment. Unlisted species are set to zero.

Species	Eb	Ec	Ed	Ee	Ef	Eg
Amount	$1.05{\cdot}10^{-2}$	$1.00{\cdot}10^{-2}$	$2.89{\cdot}10^{-3}$	$3.69{\cdot}10^{-3}$	$4.71{\cdot}10^{-4}$	$2.65{\cdot}10^{-3}$
Species	Eh	Et	R	S	Tc1	Tc2
Amount	$2.65{\cdot}10^{-3}$	0	$5.45{\cdot}10^{-3}$	$4.67{\cdot}10^{-2}$	$6.30{\cdot}10^{-3}$	0

presence of oxygen as shown in the plots of Figure 4.7 with a blue background. Then we simulate the cells for half an hour using a prediction horizon $t_\mathrm{p} = 32$ min, which was calculated via the algorithm described in Section 4.3 using the initial values. As we are working in a highly dynamic environment we want to ensure maximal flexibility in the solution and use a minimal iteration time $t_\mathrm{i} = 2$ min as dictated by the step size of the discretization $h = 2$ min. We keep these values fixed for the whole experiment. At the end of this aerobic growth we use the values for the external species $Y(30 \text{ min})$ and the current biomass composition $P(30 \text{ min})$ as as initials for the next simulation part in anaerobic conditions. Every 30 min we switch the conditions and repeat the simulation until the gene expression rates stabilize and the biomass trajectories become periodic. For the example this already happens after a single cycle, meaning we can extend the plots (C) and (D) from Figure 4.7 over arbitrary time scales by repeating the trajectory of the interval [60 min, 120 min].

Evaluating the results from this experiments yields the following. While the overall growth rate stays nearly constant during the simulation, we observe a short drop after the switch from aerobic to anaerobic conditions. This can be explained by a lack of enzyme $\mathrm{E_E}$, which is necessary for anaerobic growth and not produced in an aerobic environment. Therefore, the capability for anaerobic growth dwindles during these phases. If the switch happens in the other direction, the model can sustain its anaerobic growth mode and retains full flexibility to restart the production of $\mathrm{E_T}$ to increase the metabolization of oxygen. Hence, the growth rate can only increase when oxygen becomes available again. Lastly, we must stress the fact, that the model always depletes the external reservoir of the fermentation product E, if oxygen becomes available. This is due to the fact that the model does not need a specific transporter for the uptake and can use it to boost growth.

Overall, we see very rapid responses from the model after the switches. Especially, the sudden increase in production of the secretion product $\mathrm{E_{ext}}$ after the anaerobic are unrealistic as the whole ribosome focuses on the production of a single enzyme $\mathrm{E_E}$ in these times. In a biological system we usually expect a longer lag-phases if the conditions change so rapidly. Additionally, we can not observe an adaptation process going on as the model only optimizes for its current environment and not for a possible switch in the future.

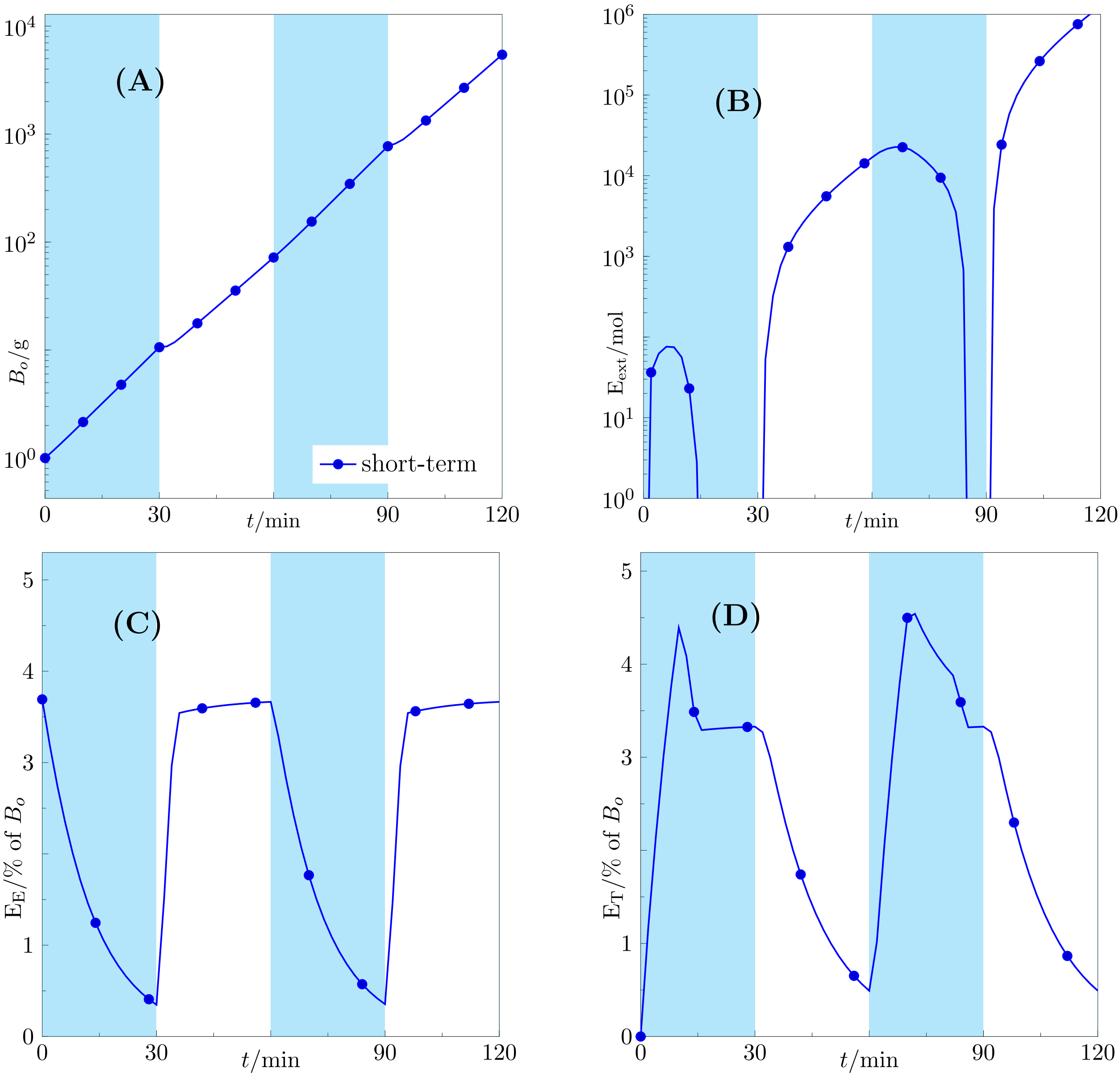


Figure 4.7: Results for short-term deFBA in the aerobic - anaerobic switching conditions. The solution stabilizes after a runtime of 120min. This means the biomass composition becomes periodic afterwards. (A) shows the objective value over time on a logarithmic scale. We can see small decreases of the growth rate when switching to anaerobic conditions as the system run into a bottleneck due to the absence of enough $\mathrm{E_E}$, but for most of the time the system experience exponential growth. (B) shows the development of the fermentation product $\mathrm{E_{ext}}$. While at the start of aerobic phases still a bit E is secreted the system quickly starts taking this up again until its external amount goes to zero. (C) present the biomass share of the enzyme $\mathrm{E_E}$ which is needed for anaerobic growth. While its production is neglected during aerobic growth, after the switch the model focuses on the production of this component. (D) The enzyme $\mathrm{E_T}$ is needed for the respiratory cycle, presented as reaction $V_{\mathcal{X},9}$. It reaches a peak after 10min in aerobic growth to balance out the additional energy cost for the uptake of $\mathrm{E_{ext}}$ but returns to a fixed value in the presence of oxygen. Otherwise, it is not produced.

5 Robust deFBA

As we have shown in the previous example the short-term deFBA is unable to generate predictive solutions estimating how cells adapt in rapid changing environments. Of course, the problem lies in the optimality of the solutions for a deterministic environment. This means, we need to extend the sdeFBA to include some form of uncertainty in the availability of nutrients. Because the sdeFBA was already planned to be used in model predictive control applications we started looking for robust optimization techniques in Model Predictive Control (MPC) to adapt to the deFBA formalism. While a comparative review of all possible ways to include uncertainties in an optimization problem would be to much for this work, we refer to these review papers on the topic [9], [36].

We chose a method called *multi-stage Model Predictive Control* (msMPC) [68] as basis for our work due to its straightforward implementation using only deterministic modeling. We will discuss the method in more detail after formally introducing it.

5.1 Multi-stage MPC

The msMPC was introduced in [68], [69], and [67] as a new way to ensure robustness of the solutions of nonlinear MPC problems. When talking about robustness, we mean that the model contains some parameters $d \in [d_{\min}, d_{\max}], d \in \mathbb{R}^{n_d}$ for which the exact values are unknown or might change over time. One type of uncertainty can affect model parameters due to measurement errors, model approximations, etc. For our cellular models, we are more interested in a type of uncertainty used to describe limited knowledge of the environment and the future thereof, e.g., changes in temperature, pressure, or the availability of nutrients. A third type is the unknown behavior of counter-acting agents, e.g., other microorganisms in the medium. Competing with other species for nutrients might hugely impact the decision for our modeled organism, but we will leave this topic for subsequent studies.

A solution to an optimization problem including these kind of uncertainties is called *robust* in the uncertainty d if the solution is feasible for all values d can assume. This concept is sketched in Figure 5.1, which depicts a possible solution to a nominal problem with a fixed value for d in a continuous black line. These solutions often tend to be on the boundaries of the solution space and therefore ”touch” constraints, meaning if the constraint is given as a less-or-equal constraint, the solution trajectory fulfills the constraint as equality. But for other values of d the calculated controls might lead to all values inside the shown tube of states, depicted via the dashed lines.

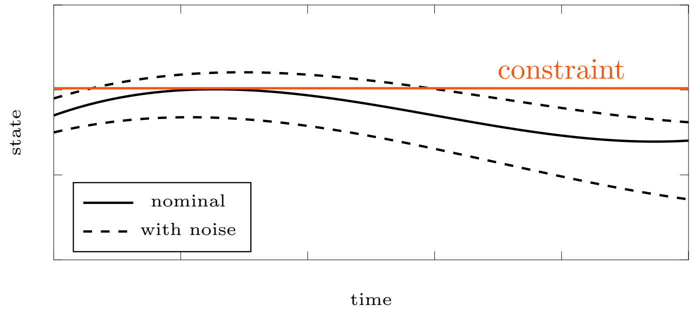


Figure 5.1: This sketches the meaning of robustness. On the y-axis the state space is shown and we assume there exists a constraint on the feasible state space given as the red line. The continuous black line shows an optimal solution to the attached problem using a fixed value for the uncertainty. But in reality all values inside the dashed tube might be the true state for the system using identical controls but other values of d. Hence, the solution is infeasible for the robust optimization problem.

In this case, the solution is obviously not robust as there exist d values for which the trajectory is infeasible and violates the constraint.

The best known approach to ensure this kind of robustness is the *min-max MPC* [19], which optimizes in a worst-case scenario setting. This method is a two-stage optimization problem in which an inner problem calculates the optimal objective value in dependency of the uncertainty value. The outer problem works opposing to the inner problem and chooses the value of the uncertainty to minimize the objective. This approach suffers at two fronts, these optimization problems are usually quite hard to solve and the predictions from min-max approaches are often very conservative. This originates in the fact that min-max approaches do not take the receding prediction horizon and the increase in information on future states into account. In some cases this might even lead to infeasible trajectories [100]. While extensions to closed-loop min-max MPC exist [42], [44] the resulting problems are even harder to solve and require some additional restrictions for the controls to be applicable.

The msMPC takes a different approach to this problem by assuming that the effects of an uncertainty can be represented via a scenario tree. The idea of these trees is shown in Figure 5.2. In this sketch the system starts in the state Z_0 and the model contains a single uncertainty d, which can assume three values d^1, d^2, and d^3. We assume that the values stay constant during the time step. Each of these values creates a branch in the tree, allowing for different inputs V_1^1, V_1^2, V_1^3 leading to the new system states Z_1^1, Z_1^2, Z_1^3. At this time point the uncertainty might change value and from each of the three new states another set of three branches springs. This continues on until the chosen prediction horizon is reached. We call a path starting at the root Z_0 to a final leaf *scenario*.

This kind of construction for a scenario tree is only possible, if the uncertainties

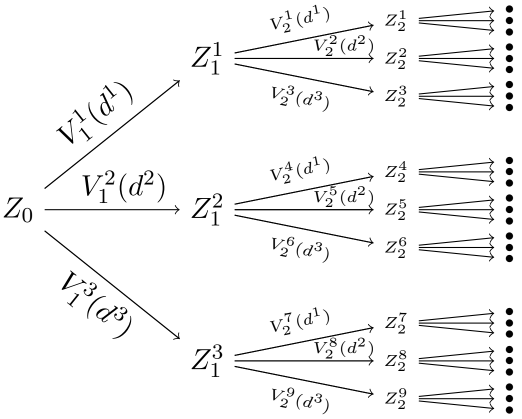


Figure 5.2: Scenario tree branching in every node.

assume only discrete values. We can elude this problem by substituting continuous uncertainties with sampled values from their range and thus approximating the continuous problem. Unfortunately, robust constraint satisfaction might not be guaranteed in this case [69].

The problem with the scenario based approach is to construct a single optimization problem dealing with all possible scenarios at once. The size of this resulting optimization problem grows exponentially with the prediction horizon, the number of uncertainties, and the number of values each uncertainty can attain. Hence, it is necessary to simplify the tree, which we will present in the next section.

5.2 Constructing a single optimization problem

Contrary to most applications of msMPC we are not directly interested in a model predictive application of our cellular models. Instead, we want to emulate the decision process behind the robust expression of enzymes necessary to survive in the uncertain environment. This means, we must prepare scenarios representing all possible changes in the environment. Without loss of generality we can assume the nutrient situation will only change once on the prediction horizon. This can be assumed as we will only apply a short part of the calculated trajectories as we did in the sdeFBA. Therefore, possible further changes in the environment are modeled in a later iteration.

This assumption alone simplifies the scenario tree immensely as it limits the branching process to a single event at the root. The example tree in Figure 5.3 was generated by three uncertain values d_1, d_2, and d_3, which can assume the discrete values 0 or 1. Thus, we have a total of eight possible realizations of the uncertainty vector $d \in \{0,1\}^3$ and each of these realizations corresponds to a single scenario in the tree. To distinguish the scenarios, we will use upper indices starting at zero. This means if the model contains n_d uncertain external species, we distinguish 2^{n_d} realizations $d^j \in \{0,1\}^{n_\mathrm{d}}$,

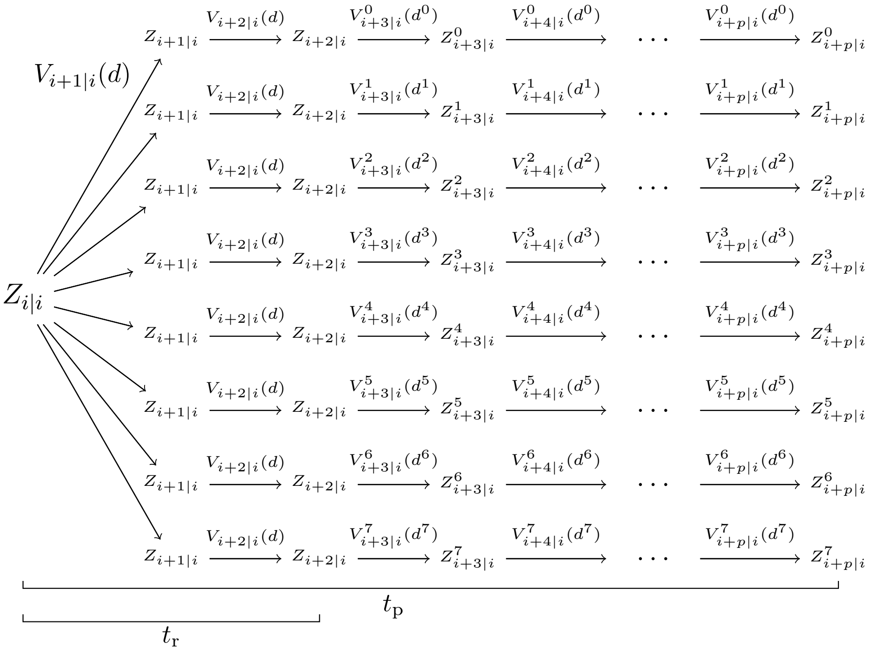


Figure 5.3: Reduced scenario tree only branching once. The constraint (5.3) enforces all reaction rates to be identical on the robust horizon, meaning they are feasible for all values of d. Hence, the first two sets of rates depend on d instead of the individual values. This also means that the first two calculated sets of states are also identical for each scenario $Z^k_{i+1|i} = Z^j_{i+1|i}$, $Z^k_{i+2|i} = Z^j_{i+2|i}$, $\forall j, k$.

$j \in \{0, \dots, 2^{n_\mathrm{d}} - 1\}$. We write shortly

$$\mathcal{J} = \{0, \dots, 2^{n_d} - 1\}. \tag{5.1}$$

For a unique identification of the scenarios we use binary mapping of the values to their respective vector d^j. For example, we can determine the value of d^6 via the binary representation of 6 as 110. Therefore, $d^6 = [1, 1, 0]$. This way d^0 always corresponds to the scenario in which all uncertainties assume the value zero. We call this the *zero scenario* and use it to represent the real current environment, while all other branches are "might-be" scenarios.

Due to the discrete nature of the scenario tree, we use the discrete formulation for all variables as presented in Section 4.5. The decision variables $V^j_{k|i}$, with the discrete notation as introduced in (4.41), are assumed to be independent from each other in the different scenarios. Additionally, Figure 5.3 depicts a new time variable $t_\mathrm{r} < t_\mathrm{p}$ called the *robust horizon*. We calculate the *discrete robust horizon* $r \in \mathbb{N}$ in the same way as we did with the iteration time depending on the step size h as

$$r = \lfloor \frac{t_\mathrm{r}}{h} \rfloor, \; r \geq 1. \tag{5.2}$$

Choosing the size of the robust horizon strongly depends on the step size h. From our experience, setting $t_\mathrm{r} = h$, resp. $r = 1$, is usually a good choice, but for very small values of h this might lead to unexpected results. Therefore, we see t_r currently as a tuning parameter.

The robust horizon is related to the iteration time t_i we encountered during the definition of the sdeFBA, as it defines the length of the trajectories which are assumed to be robust, meaning feasible for all scenarios. We do this by enforcing a new type of constraint on the reaction rates called the *nonanticipativity constraint* [28]

$$V^g_{i+k|i} = V^j_{i+k|i} = V_{i+k|i}, \forall g, j \in \mathcal{J},\ k \in \{1, \dots, r\}. \tag{5.3}$$

The constraint (5.3) is the direct reason for the robustness of the solution as it enforces that the reaction rates on the robust horizon must be feasible for all scenarios at once.

If the rates are feasible the associated states $Z^j_{i+k|i}$ are as well and the solution is robust in the sense from Figure 5.1. Please note, that this form of the nonanticipativity constraint is a bit weakened in comparison to the one used in the original msMPC introduction [69], which reads:

$$V^g_{k|i} = V^j_{k|i},\ \text{if}\ Z^g_{k-1|i} = Z^j_{k-1|i},\ \forall g, j \in \mathcal{J},\ k \in \mathcal{K} = \{1, \dots, p\}. \tag{5.4}$$

For our application this would mean that the reaction rates must be identical if the parent states are identical. We do not plan to introduce uncertainty to the dynamics of our models, hence, a nonanticipativity constraint like (5.4) would force all reaction rates to be identical for all scenarios. The weakened formulation allows the rates to deviate after the robust horizon, meaning the rates can be chosen freely for $t > t_\mathrm{r}$.

While the nonanticipativity constraint guarantees robustness, we still have to define a coupling of the objective values of the different scenarios. We introduce a shared objective function defined as

$$J = \sum_{j \in \mathcal{J}} o^j_\mathrm{w} B^j_\mathrm{o}(t), \tag{5.5}$$

with the objective biomass B^j_o for each scenario and their *probability weights* $o_{\mathrm{w}^j} \in \mathbb{R}_{\geq 0}$, $o_\mathrm{w} \in \mathbb{R}^{2^{n_d}}_{\geq 0}$. While in our case each scenario utilizes the same objective biomass functional (3.12)

$$B^g_\mathrm{o} = B^j_\mathrm{o} = B_\mathrm{o},\ \forall g, j \in \mathcal{J}, \tag{5.6}$$

it is possible to include uncertainty in the objectives as well; meaning uncertainty in the molecular/objective weights of the species. The probability weights for the objectives are very useful in this case, as they enable us to include the probability of a scenario happening. We enforce the weights to add up to one

$$\sum_{j \in \mathcal{J}} o^j_\mathrm{w} = 1. \tag{5.7}$$

Therefore, each weight directly reflects the probability of a scenario occurring. The beauty in this approach lies in the fact that we can make these dependent on the current time $o_{\mathrm{w}}^{j}(t)$ to emulate progression in the knowledge of the cell on their environment.

Consider as an example a population of bacteria living on a beach and thus being influenced by the tides. After the sea level changes the bacteria "knows" that the environment will stay this way for roughly the next six hours. This can simply be translated into mappings from time to scenario probability. But, to really implement this an extensive knowledge about the environmental changes and their timing is necessary. Nevertheless, these timed behaviors can be observed in most kinds of cells and are synchronized via the circadian clocks of the respective species, e.g., cyanobacteria [90], mammalian cells [86], or yeast [29].

5.3 Uncertainty in the environment

So far, we have explained the meaning of "robustness" as attribute of robust optimization techniques to ensure that no constraints are violated regardless of the uncertainty. While this is important for our application to ensure physical feasibility, we are more interested in *robust cell cultures*. This means the population has developed an inherit ability to survive regardless of sudden changes in their living conditions. We want to exploit the robust optimization to predict how this robust adaptation looks like. We approach this by introducing uncertainty in the environment which forces our models to anticipate the possible change in the environment and prepare for it.

For the construction of these uncertain environments, we orient ourselves at the setup for switching experiment described in Section 4.6.2.2 using the *carbon core* model. We consider an identical setting with sudden switches from aerobic to anaerobic growth conditions.

To model the robust decision process in this case the model can not be constraint to the current availability of oxygen but also on the possibility that this might change. We have to distinguish between two problem classes:

1. A nutrient N_{u} is unavailable but might become available again.
2. A nutrient source N_{a} is currently available and might run out.

In both cases, we express the uncertainty via a new constraint in the form

$$dV_{\mathrm{in}}(t) \leq c_{\mathrm{limit}}(t), \tag{5.8}$$

with the uncertainty $d \in \{0, 1\}$, the uptake reaction V_{in}, and the time dependent uptake limit $c_{\mathrm{limit}}(t) \geq 0$. Using this kind of constraint conveys multiple benefits: The constraint is inactive for $d = 0$, the nutrient dynamics are not interfered with, existing limitations to the nutrient uptake stay intact, and modifying the uptake limit

$c_{\text{limit}}(t) \geq 0$ is easy. When using this kind of constraint, we want to ensure that $d = 0$ reflects the current situation, while $d = 1$ encodes the possibility for change to keep the notation with the zero scenario intact.

This translates in situations in which a nutrient source N_{u} might become available again to the following scenarios. In the first one with $d_{\text{u}} = 0$ no uptake of N_{u} is possible. We can express this by forcing the uptake reaction $V_{\text{in,u}}$ of N_{u} to zero, via an adapted form of (5.8)

$$(1 - d_{\text{u}})V_{\text{in,u}}(t) \leq 0, \; d_{\text{u}} \in \{0, 1\}. \tag{5.9}$$

Due to the nonanticipativity constraint (5.3), the uptake rate of N_{u} will be zero in all scenarios on the robust horizon $V^{j}_{\text{in,u}}(t) \leq 0, \; t \in [0, t_{\text{r}}], \; \forall j \in \mathcal{J}$. Therefore, the question is how we construct the scenario in which the nutrient becomes available again. There are a multitude of possible formulations but trial and error showed us that the formulation (5.9) is sufficient with a minimal adaptation of the model.

We assume the model can grow without the supply of N_{u} at its current state. Without this assumption the model is currently in a starvation period and we might run into an infeasible problem during the simulation; independent from the uncertainty in the availability of N_{u}. The addition of N_{u} under this assumption can only enrich the environment and might increase the achievable growth rate. Assuming the uptake of N_{u}, or another reaction down the pathway metabolizing it, is constrained by an enzyme or transporter, we can delete the dynamic species N_{u} from the model without risking any impact on the solution of the zero scenario. We realize this by substituting the uptake reaction

$$V_{\text{in,u}} : N_{\text{u}} \rightarrow N_{\text{u,internal}} \tag{5.10}$$

with

$$V_{\text{in,u}} : \emptyset \rightarrow N_{\text{u,internal}}. \tag{5.11}$$

This way, in the scenario in which (5.9) is inactive, uptake of N_{u} is possible after the robust horizon. This formulation is very simple and does not depend on the current state of the model. But due to our assumption the model must still invest in enzymes to be able to metabolize N_{u}. Therefore, we have created exactly the situation we were looking for: The model can decide whether it is reasonable to invest some nutrients to prepare for the possibility that N_{u} becomes available again. As the model can grow without N_{u} and the zero scenario will not benefit from investments done on the robust horizon, the possible growth gain from N_{u} must be very large to make this investment reasonable. Of course, this decision depends on the models state and the chosen robust horizon.

This approach can become a bit more complicated if N_{u} is also an secretion product or can become available from other deterministic sources. In these cases we suggest to create a copy $V_{\mathrm{in,u,2}}$ with the suggested stoichiometry and enforce the robust constraint only on this reaction. Please note that enzyme capacity constraints must then be adapted as well.

For the second case, in which a nutrient might deplete we are facing a different challenge. While we will again be using a constraint in the form (5.8), we must ensure that the scenario in which the uptake of N_{a} is constrained does not become infeasible. A simple approach would be to utilize the same idea from the inverse situation. Allow uptake of N_{a} over the robust horizon with additional constraints and then cut off the supply completely for scenarios with $d_{\mathrm{a}} = 1$. Unfortunately, we encountered cases in which this is not possible. Consider a model containing distinct pathways for aerobic and anaerobic growth, which utilize independent enzymes. If the model is grown exclusively under aerobic conditions, the enzymes for anaerobic growth will never be expressed and the model is unable to grow without oxygen. This situation further deteriorates, if the model contains maintenance reactions.

Using this way to construct the scenarios, the model can only express enzymes necessary to survive under the anaerobic conditions on the robust horizon. Depending on the models state, this is simply not enough time to generate enough enzyme to satiate maintenance constraints. Instead, we use an additional ramp function in the construction of the uptake limit as

$$c_{\mathrm{limit}}(t) = \begin{cases} \infty, & t \in [0, t_{\mathrm{r}}] \\ (1 - \frac{t}{t_{\mathrm{b}} - t_{\mathrm{r}}}) v_{\mathrm{in,r}}, & t \in (t_{\mathrm{r}}, t_{\mathrm{b}}] \\ 0, & t \geq t_{\mathrm{b}}, \end{cases} \tag{5.12}$$

with the new parameter *preparation time* $t_{\mathrm{b}} \in \mathbb{R}_{\geq 0}$, $t_{\mathrm{b}} \in (t_{\mathrm{r}}, t_{\mathrm{p}}]$ and the *robust uptake limit* $v_{\mathrm{in,r}}$. We want to emulate the total depletion of N_{a} with this formulation. With the preparation time we can give the model more time to adapt the cessation of N_{a} supply. A suitable value for the robust uptake limit can be calculated from previous values of the uptake rate

$$v_{\mathrm{in,r}} = v_{\mathrm{in},old} \cdot e^{\mu_{\mathrm{max}} t_{\mathrm{r}}}. \tag{5.13}$$

This means, we simply approximate the maximal uptake rate for N_{a} from the previous values of the uptake reaction under the assumption that the model grows exponentially. The old value for the uptake can either be taken from an RBA solution for the current environment or from a previous iteration during the run-time. The maximal exponential growth rate μ_{max} can also be determined by an RBA in a rich environment containing all possible nutrients in surplus.

The tuning parameter t_{b} is harder to determine and must either be guessed or fitted with experimental data. From the perspective of the robust optimization its impact is even larger than the robust horizon. A longer preparation time might enable the cells to fully adapt to the changed environment, but at the same time this lowers the adaptation stress. The faster the cells must adapt the bigger this stress becomes and the more likely, the cells will start adapting on the robust horizon. Choosing t_b too small might lead to the infeasible scenario we wanted to avoid in the first place.

We must distinguish between different types of stress. Preventing an infeasible scenario has always the highest priority as this means cell death. Therefore, this causes immediate adaptation. If other kinds of nutrients are available to substitute the uncertain source N_a, the robust formulation can only infer minor adaptation stress. Most likely in these cases, the optimal robust behavior is to utilize N_a only if necessary and focus on more reliable food sources.

5.4 Implementing the robust deFBA

With the uncertain environments in place, we can formalize the *robust dynamic enzyme-cost Flux Balance Analysis* (rdeFBA) as a discrete iteration scheme on the discretized time $\Delta_t(h)$ with the step size h, cf. Section 4.5. As before we will use the forward Euler to simplify the dynamics for easier reading. As the computational cost increases in comparison to the sdeFBA we lowered these by implementing a new discretization scheme presented in Section 5.5. The rdeFBA reads in pseudo code:

```
procedure ROBUST DEFBA($Y0, P0, t_\mathrm{end}, h, p, r$)
    Initialize empty solution trajectories $Z^*$, $V^*$
    $i = 0$
    $Z^*_0 = [Y0, P0]$
    $t = 0$
    while $t < t_\mathrm{end}$ do
        solve:
                        $\max_{(V^j_{k|i}), j \in \mathcal{J},\ k \in \mathcal{K}} \sum_{j \in \mathcal{J}} o^j_\mathrm{w} \sum_{k \in \mathcal{K}} B^j_{k|i}$      (5.14a)
            with given  $Z^j_{i|i} = Z^*_i,\ \forall j \in \mathcal{J}$      (5.14b)
            subject to: $Z^j_{k|i} = Z^j_{k-1|i} + hSV^j_{k-1|i}$      (5.14c)
                        $S_\mathcal{X} V^j_{k|i} = 0$      (5.14d)
                        $H_a Z^j_{k|i} - H_f V^j_{k|i} \leq 0$      (5.14e)
                        $H_b Z^j_{k|i} \leq 0$      (5.14f)
                        $H_c V^j_{k|i} - H_e Z^j_{k|i} \leq 0$      (5.14g)
                        $H^j_d V^j_{k|i} \leq C^j_{k|i}$      (5.14h)
                        $Z^j_{k|i} \geq 0$      (5.14i)
```

$$V_{\min} \leq V^j_{k|i} \leq V_{\max}, \forall (k,j) \in (\mathcal{K} \times \mathcal{J}) \tag{5.14j}$$

$$\text{and} \quad V^g_{k|i} = V^j_{k|i} = V_{k|i}, \ \forall g, j \in \mathcal{J}, k \in \{i, \ldots, i+r\}, \tag{5.14k}$$

```
            $Z^*_{i+1} = Y_{i+1|i}, P_{i+1|i}$
            $V^*_i = V_{i|i}$
            $i+ = 1$
            $t+ = h$
            Additional break conditions can be placed here
        end while
        return $Z^*, V^*$
    end procedure
```

with $\mathcal{K} = \{i, i+1, \ldots, i+p-1 | hp = t_\mathrm{p}\}$, the discrete horizons p and r. This scheme is very similar to the short-term deFBA as it also utilizes the receding prediction horizon, but beside the extension to the different scenarios we also included the newly introduced uncertainty constraint (5.14h).

This constraint uses a new filter matrix $H_d \in \{0,1\}^{n_d,m}$ and the discrete approximation of the continuous $C_{limit}(t)$ vector which entries correspond either to c_{limit} (5.12) or zero (5.9).

As the dynamics are not subject to any uncertainty (5.14c) the nonanticipativity constraint (5.14k) leads to identical states

$$Z^j_{k|i} = Z^g_{k|i} = Z_{k|i}, \ k \in \{i+1, \ldots, i+r\}. \tag{5.15}$$

Therefore, it would be possible to utilize the robust horizon in the same sense we used the iteration time in the short-term deFBA. However, we assumed during the construction of the scenario tree, that only a single environmental change happens during the prediction horizon. To keep the influence of this assumption minimal and allow the model to react more quickly, we usually use a minimal robust horizon $r = 1$, respectively $t_\mathrm{r} = h$.

5.5 Solving rdeFBA with `deFBA-Python`

When we started implementing the robust deFBA into the `deFBA-Python` toolbox we quickly ran into two problems. The increased problem size for the rdeFBA made the use of the complex collocation used by the `LinOpt` class very costly. Furthermore, it turned out to be very complex to include all features of the rdeFBA with varying robust horizons, probability weights, and the new environmental constraints (5.14h) in the existing collocation. Thus, we were forced to implement a new collocation method, which results in a simpler linear program to solve and which we can control more easily at the same time.

We chose to adapt a collocation based on the implicit midpoint method as presented

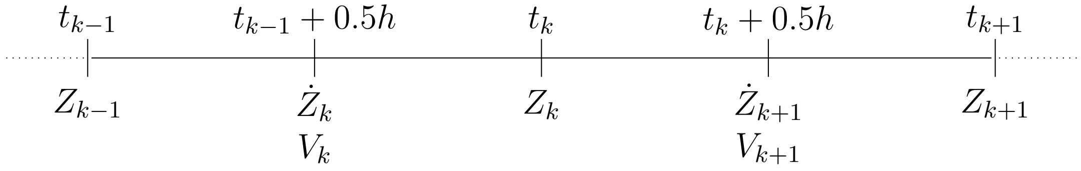


Figure 5.4: Visualization of the location for the discrete approximations. States are evaluated on the time grid and derivatives, fluxes are evaluated in between the grid points.

in [92] for the discretization of the ODEs. The idea is to solve an initial value problem in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = f(t, y(t)),\ y(t_0) = y_0, \tag{5.16}$$

via the approximation

$$y_{k+1} = y_k + hf\left(t_k + \frac{h}{2}, \frac{1}{2}(y_{k+1} + y_k)\right). \tag{5.17}$$

As presented in [92] this translates for deFBA models to different evaluation points for states and reaction rates on the time grid $\Delta_t(h)$. This is visualized in Figure 5.4. While the states are evaluated at the grid points t_k, the fluxes and derivatives are evaluated at $t_k + h/2$. Applying this as a collocation method to the deFBA we end up with the following LP

$$\max_{(Z_k,\dot{Z}_k,V_k)_{k\in\mathcal{K}}} \sum_{k\in\mathcal{K}} w_o^T Z_k \tag{5.18a}$$

$$\text{s.t. } Z_k = Z_{k-1} + h\dot{Z}_k \tag{5.18b}$$

$$\dot{Z}_k = SV_k + \frac{1}{2}(Z_{k-1} + Z_k) \tag{5.18c}$$

$$S_{\mathcal{X}}V_k = 0 \tag{5.18d}$$

$$-\frac{1}{2}H_a(Z_{k-1} + Z_k) + H_f V_k \leq 0 \tag{5.18e}$$

$$H_b Z_k \leq 0 \tag{5.18f}$$

$$H_c V_k - \frac{1}{2}H_e(Z_{k-1} + Z_k) \leq 0 \tag{5.18g}$$

$$Z_k \geq 0 \tag{5.18h}$$

$$v_{\min} \leq V_k \leq v_{\max}, \forall k \in \mathcal{K}, \tag{5.18i}$$

with $\mathcal{K} = \{1, \ldots, N\}$, $Nh = t_{\text{end}}$ and given Z_0 as initial values. While it is possible to eliminate the state derivatives $\dot{Z}_k$ completely by substituting (5.18c) into (5.18b), we keep them for simpler implementation and easier modification of the method.

In the collocated problem we encounter three types of constraints: Path constraints

(5.18c), (5.18g), (5.18e); control constraints (5.18d), (5.18i); and state constraints (5.18f), (5.18h). We also have to consider the update rule (5.18b), of course.

To utilize the midpoint collocation in the robust setting we must handle two additional constraint types: Contemporary constraints acting only on parts of the trajectories to handle the nonanticipativity constraint (5.14k) and control constraints with time dependent right hand sides for the scenario based constraints. The numerical implementation of these features was realized in the `MidOpt` class, which we designed in a way to be compatible with the existing code using the `LinOpt` class. Therefore, it is possible to utilize the midpoint collocation scheme in the regular deFBA and the short-term deFBA as well. As the midpoint collocation is not as sophisticated as the midpoint rule, it might be necessary to reduce the step size h when using this method. Even when using the smaller step size, the midpoint collocation very effectively decreased computing times in our examples. The results in these cases are nearly identical to the ones we calculated with the `LinOpt` class using larger step sizes. We present an example in the upcoming section.

The robust deFBA can be accessed from the `DefbaModel` class via the method `DefbaModel.robust()`. While the options are basically identical to the ones from short-term deFBA, the important addition is the necessary attribute `uncertain_values`. In the current version of `Python-deFBA` it is possible to enter either model parameters in form of k_{cat} values, objective values w_o, biomass fractions ϕ, or maintenance coefficients φ. The identifiers for the parameters are derived from the SBML containing the model. It is possible to enter an arbitrary amount of possible values for each parameter and the system generates the according scenario tree. While we have not discussed this function in this text, we feel it might be very useful for some users.

In the software the scenario trees are generated based on uncertainties in these parameters. To include uncertain scenarios in this workflow, the user has to generate a pseudo parameter by running `DefbaModel.addUncertainEnvironment(Reaction_name, decreasing = Boolean)`. Here the `Reaction_name` must correspond to the name of a exchange reaction and `decreasing` defines whether the nutrient source transported by the reaction will run out or might become available again. The user might then call the robust deFBA via `DefbaModel.robust(uncertain_environment)`, with `uncertain_environment` being a dictionary defining the uncertain environment and containing the preparation time as fraction of the prediction horizon, e.g. `uncertain_environment = {environment:[0.5]}` represents a preparation time $t_{\text{b}} = 0.5 t_{\text{p}}$.

5.6 Numerical examples

In the following sections we will showcase results from the robust deFBA and compare them to results obtained with regular short-term deFBA. This way we validate the

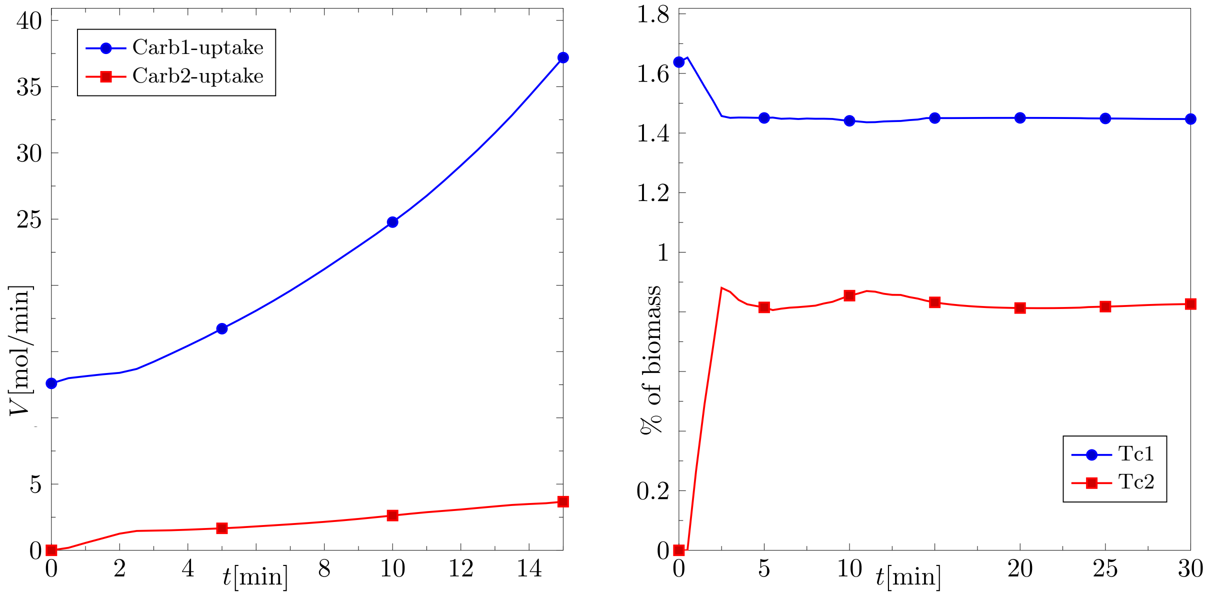


Figure 5.5: Results for the robust carbon switching experiment. (left) Uptake rates over time. After an initial focus to increase the uptake capabilities for Carb2, the rates stabilize after $t = 2.5$min and increase both exponentially. (right) Biomass percentage for carbon transporters over time. After the initial focus on the production of Tc2 and a bit of adaptation afterwards, the system enters balanced growth after 20min in which the biomass percentages for the transporters stay constant.

robust deFBA and show how robust modeling affects the simulations.

5.6.1 Carbon-core model

In this section we will use the *carbon core* model from Section 4.6.2. We will investigate how the availability of the two carbon Sources Carb1 and Carb2 influence the results and repeat the switching experiment from Section 4.6.2 with the robust deFBA afterwards.

5.6.1.1 Carbon source switching

Focusing on the availability of the carbon sources, we can construct a very simply but interesting problem setting. As Carb1 is the preferred carbon source, the model will never utilize Carb2 as long as enough Carb1 is present. Hence, we want to investigate the effects of the robust method by assuming Carb1 might suddenly run out. In reality both Carb1 and Carb2 are present in excess, but we create a scenario as described before. For the numerical solution we use the following parameter values: discretization step size $h = 0.5$min, prediction horizon $t_{\mathrm{p}} = 30$min, robust horizon t_{r}=1.5min, and preparation time $t_{\mathrm{b}} = 2$min. The preparation time was chosen very small as the model does not contain maintenance and it can adapt very quickly to changes as shown in the switching experiment with the sdeFBA (cf. Section 4.6.2.2). We also supply oxygen in limitless quantities and calculated the initial values shown in Table 5.1 via an RBA. The initial biomass amount was chosen as 1g. The RBA ran in the deterministic setting, which means that the necessary transporter Tc2 is missing from the initial values.

The results are shown in the plots in Figure 5.5. Immediately, the model starts focusing on the production of Tc2 to be able to consume Carb2 and continue to grow in the Carb1-limiting scenario. This means in the first 2.5 minutes, the share of Tc1 in the biomass decreases and the uptake rates for Carb1 stay roughly level. Afterwards, the cell must equalize the amounts of the other enzymes again which leads to small variations in the Tc2 share of the biomass. But after just 19 min, the model enters a new balanced growth phase, which stills favors the usage of Tc1 as carbon source, but also utilizes Carb2 and thus be prepared for a sudden depletion of Carb1.

This result is quite sensitive to the robust horizon and the preparation time. Larger preparation times for example can lead to results without any expression of Tc2. This can easily be explained by the lack of starvation penalty in the model. The fast growth on Carb1 makes it sub-optimal to invest in Tc2 in this case; even under the zero-growth phase at the end of the limiting scenario.

On the other hand, a combination of a small robust horizon and short preparation time can make any prolonged utilization of Carb1 sub-optimal and the model focuses solely on the uptake of Carb2, which is fully available in both scenarios.

This example already shows that we have to investigate the impact of these parameters in more detail and find a way to choose the values systematically as we did with the iteration time in the previous chapter. At the same time these results proof our concept, as the rdeFBA predicted results fit to a robust population, exchanging some growth rate for the ability to cope with sudden changes.

5.6.1.2 Aerobic-anaerobic switching

In this example, we repeat the switching experiment from Section 4.6.2.2 in the robust setting and compare the results to the ones from the sdeFBA. As a reminder, in this experiment we have 'pregrown' the cells under anaerobic conditions and transfer them suddenly into an oxygen rich environment. After 30 min we switch the environment

Table 5.1: Initial values for the simulation in which Carb1 supply is uncertain. Unlisted species are set to zero. Quantities are given in molar amounts.

Species	Eb	Ec	Ed	Ee	Ef	Eg
Amount	$6.82{\cdot}10^{-3}$	$6.29{\cdot}10^{-3}$	$2.39{\cdot}10^{-3}$	$2.96{\cdot}10^{-7}$	$5.33{\cdot}10^{-4}$	$2.94{\cdot}10^{-3}$

Species	Eh	Et	R	S	Tc1	Tc2
Amount	$2.94{\cdot}10^{-3}$	$6.69{\cdot}10^{-3}$	$6.12{\cdot}10^{-3}$	$4.67{\cdot}10^{-2}$	$4.10{\cdot}10^{-3}$	0

from one oxygen state to the other. To ensure comparability between the results, we recalculated the sdeFBA solution with the midpoint collocation and use an identical step size. Furthermore, we use the same initial values for the robust deFBA as for the short-term solution, cf. Table 4.7. The parameters for the simulation are chosen as: Step size $h = 0.5$ min, prediction horizon $t_\mathrm{p} = 33$ min, robust horizon $t_\mathrm{r} = 1.5$ min, and preparation time during aerobic growth $t_\mathrm{b} = 16.5$ min.

The results are presented in Figure 5.6. First, we see an identical periodicity of the solutions for the robust deFBA as we do with the sdeFBA solutions. This means, the solutions curves for the biomass percentage for the time 60 min to 120 min will repeat afterwards.

We are very interested into the performance of the robust solution and how it differs from the deterministic model. As we can see in plot (A), the overall biomass increase is larger for the robust solution because the lag phase after entering the anaerobic phase is shorter. This is due a less aggressive investment into the aerobic pathways during the aerobic phases. Therefore, the robust optimization handles the possible loss of oxygen by being more careful during investment.

However, the robust simulation does not contain large deviations from the short-term behavior during anaerobic growth. While we include a scenario in which oxygen becomes available again the model predicts no investment into $\mathrm{E_T}$ before oxygen becomes available again. This shows, that a possible return of a nutrient is only relevant, if expression of the necessary transporter is cheap and metabolization is simple.

The biggest deviation between the solutions is depicted in (B) and (D) of Figure 5.6. While the sdeFBA solution produces increased amounts of $\mathrm{E_T}$ to metabolize the previously secreted fermentation product $\mathrm{E_{ext}}$, the robust solution excretes it continuously even if oxygen is available. With the possibility that the oxygen might deplete again, it is infeasible to invest into the restructuring needed to metabolize E.

In this example the growth modes differ stronger than in the previous example. This decreases the impact of the parameters t_r and t_b and makes it much easier to find suitable values. At the same time, the anaerobic growth mode is still a backup growth mode, which can be used in all scenarios. But a test has shown, that just staying in the anaerobic growth mode yields worse results than the short-term solution. While the robust solution predicts an oxygen consumption due to the increase in growth rate, it does not overshoot in the production of $\mathrm{E_T}$. Instead of 'wasting' nearly 10 minutes of most production capacity on $\mathrm{E_T}$ just for a short growth burst in the sdeFBA, the robust solution even increases it auto catalytic capacity in the early phase of aerobic growth. Therefore, it can keep up with the sdeFBA solution even without using $\mathrm{E_{ext}}$.

5.6.2 Yeast model

One of the major results in the ROBUSTYEAST project was the creation of a new model called *deFBA_yeast* for the organism *Saccharomyces cerevisiae* by A.-M.

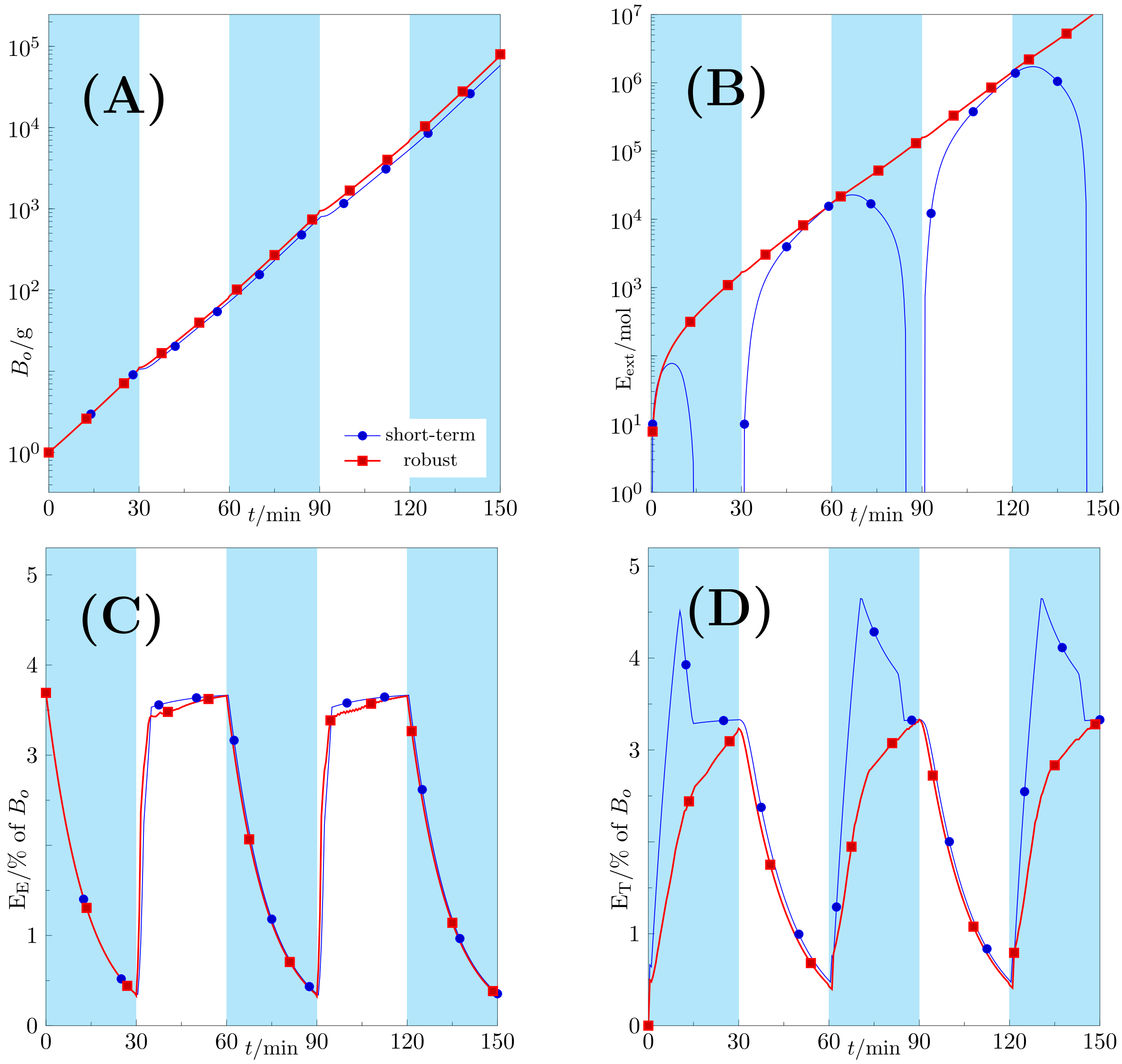


Figure 5.6: Comparison between short-term and robust results for the aerobic-anaerobic switching experiment. Phases with oxygen have a blue background. (A) Shows the development of total biomass over time. While the growth rates for both methods is nearly identical for both methods. The robust solution outgrows the regular one in the long run. This is due to the shortened lag phase when entering the anaerobic conditions. (B) The most relevant difference between the solutions is the consumption of $\mathrm{E_{ext}}$. While the sdeFBA takes it up again as quickly as possible to speed up growth, the robust solution secretes excess of internal E and does not take any up again. (C) The enzyme $\mathrm{E_E}$ is essential for anaerobic growth. While both solutions neglect the production after entering the aerobic phases. The robust solution is more reluctant in building it up again after entering the anaerobic phase. (D) The production of $\mathrm{E_T}$ spikes in the blue parts to enable the utilization of the previously secreted fermentation product E for the sdeFBA. The robust solution reaches the same level at the end of the growth phase but slowly builds to it. Both solutions do not produce it during the anaerobic growth phases.

Reimers and H. Lindhorst. The model was first published in the thesis [92]. The current version of the model in SBML using the RAM extension can be found at [1]. We refer to the thesis for a detailed explanation on how the model was constructed and model verification via the reproduction of published experimental results.

The *deFBA_yeast* model is derived from the *Yeast 6.06* model [46] using the protocol presented in Section 3.3. As the size of *Yeast 6.06* was too large for dynamic resource allocation methods, the *minimal network finder* [94] was applied to reduce the network size. It was ensured that the reduced model can attain at least 99% of the original biomass flux in the following environmental conditions:

- aerobic glucose
- aerobic galactose
- anaerobic glucose
- anaerobic galactose.

Additionally, all fermentation pathways were retained to enable the study of ethanol production. The resulting FBA model has 454 metabolites, 438 reactions, and 429 genes; which is a suitable size for deFBA applications.

To ensure the possibility that the model can also consume by-products like ethanol, some reactions from *Yeast* 7 [5] were included by hand to the reduced model. Starting with this reduced model, available at [2], the *deFBA_yeast* model was created by following the protocol described in Section 3.3. The final deFBA model contains

- 3 dynamic external species (glucose, galactose, ethanol)
- 26 exchange reactions
- 421 metabolic species
- 514 metabolic reactions
- 2 storage species
- 4 artificial quota species
- 378 enzymatic species
- 386 biomass producing reactions
- and zero maintenance reactions.

---

[1] `https://doi.org/10.15490/fairdomhub.1.model.555.1`

[2] `https://ndownloader.figshare.com/files/8653207`

At this network size, we can simulate the model using a regular personal computer. To clearly address species and reactions in the model, we use their ids given by the SBML file, e.g. external glucose has the id ”s_0565_c06” and the oxygen uptake reaction is called ”r_1979”. Oxygen is not modeled as a dynamic species in this version of the model and the possibility to utilize it is controlled by constraining reaction r_1979.

For the numerical simulation of the model with sdeFBA, we could not utilize the automatic way to choose the prediction horizon as described in Section 4.3. The model has a strong affinity to exponential growth, meaning that any calculated linear growth rates for the given environments are smaller than the respective exponential growth rates $\lambda_{\mathrm{r}} \leq \mu_{\mathrm{exp}}$. Therefore, we determined suitable values for the prediction horizon by looking for a balance of the prediction horizon and step size, which can reproduce experimental results presented in [92] while keeping the individual problem size small enough to effectively be able to produce results for the robust deFBA. Choosing the discretization step-size $h = 20$min and the prediction horizon $t_{\mathrm{p}} = 300$min$= 5$h fulfilled these conditions for the robust deFBA but the short-term solutions were not reaching appropriate growth rates.

We simulate the model in an aerobic environment containing both glucose and galactose in excess. For the sdeFBA we set the iteration time to minimum $t_{\mathrm{i}} = 20$min and determined the initial biomass composition again via RBA with an initial biomass amount of 1g. The solution for the external species shown in Figure 5.7 (left) are qualitatively comparable to the experimental results. This means that glucose is the preferred carbon source and even though oxygen is available the yeast cells produce ethanol, which is called overflow metabolism [106] as it is a very wasteful growth mode. Nevertheless, the short-term deFBA predicts a constant growth rate of $\mu = 0.308\ \mathrm{h}^{-1}$ while experimental results show a growth rate of $\mu = 0.32\ \mathrm{h}^{-1}$ in this medium. The influence of the step size and the chosen was prediction horizon was investigated, but we could eliminate this as the reason for the mismatch. Therefore, it should be experimentally verified, whether all predicted gene expression do take place *in vivo*.

The mismatch in the achieved growth rates becomes even more interesting, when investigating the results from the robust deFBA. For this simulation we assumed oxygen might become unavailable again and constrained the reaction r_1979 as described in Section 5.3 with a robust horizon of $t_{\mathrm{b}} = 20$ min and a preparation time of $t_{\mathrm{b}} = 90$ min. This solution has increased glucose uptake, ethanol secretion and growth rates compared to the short-term solution. The growth rate closely matches the experimental predictions with $\mu = 0.32\ \mathrm{h}^{-1}$ at all times.

Due to these large differences in effectiveness, we also generated a solution using the regular deFBA using the same step size and an end-time $t_{\mathrm{end}} = 20$h. The regular deFBA matches the growth rate of the robust solutions as well as the predictions for the external species. The difference in biomass accumulation for the different methods is shown in Figure 5.7 (right). As the short-term solution depicts a smaller biomass yield, we decided to compare the enzyme levels of the robust solution with the ones

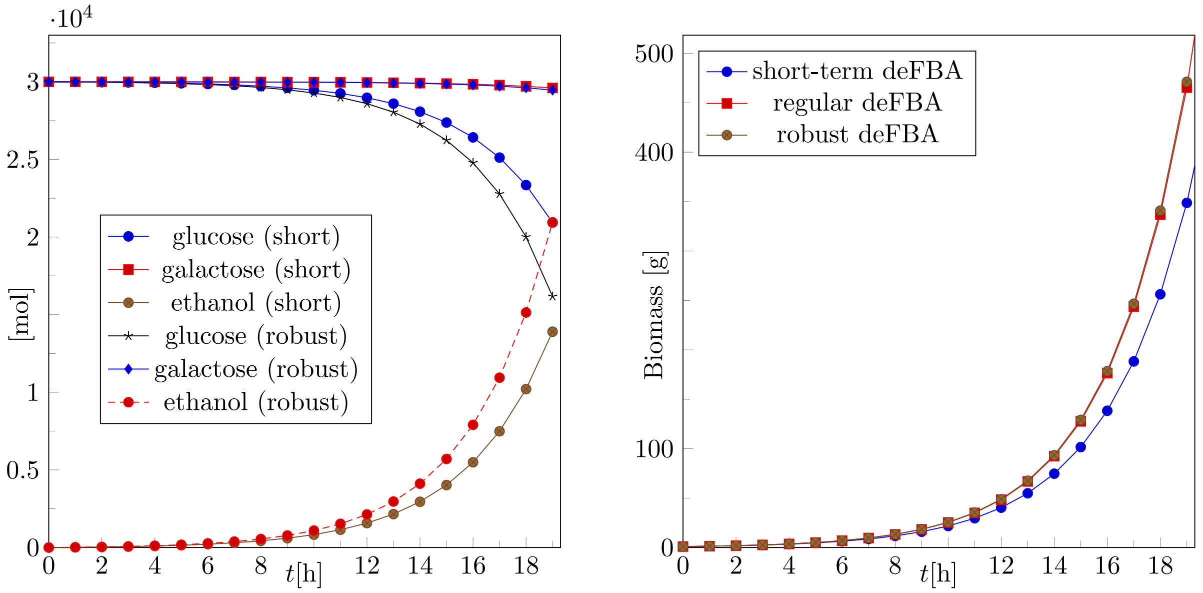


Figure 5.7: Results for external species using the short-term deFBA. (Left) Total amount of nutrients and fermentation products in the medium. Galactose is only marginally utilized. (Right) Comparison between biomass yield for different simulation methods. While the amounts of the regular and robust deFBA are nearly identical, short-term results show a reduced growth rate.

from deFBA.

Due to the complexity of the network, we can only compare the results by numbers and must refrain from a detailed analysis as with the *carbon core* network. First, we investigate the difference between enzyme predictions in RBA and the dynamic results. We find 7 enzymes used in both the regular and robust deFBA which were not predicted by the RBA. A single predicted enzyme was utilized in the robust deFBA solution and not in the regular deFBA. Additionally, the robust deFBA solution utilizes 8 enzymes not predicted by RBA and not used in the regular deFBA solution. Of these enzymes E_r_1172_c03 stands out, which is needed for the uptake of glycerol (s_0765_c03). Thus, only the robust deFBA solution utilizes glycerol, while the regular deFBA solution growths completely without it. Accordingly, the enzyme E_r_0489_1_c03 catalyzing a glycerol phosphatase is also only produced in the robust deFBA solution. The other enzymes unique to the robust deFBA solution can be explained by a shift in the choice of isoenzymes. For a complete list of the mentioned enzymes we refer to Table B.2.

To analyze the difference between the two dynamic solutions, we define a multiplier

$$x_i = \begin{cases} \frac{E_{r,i}}{E_{d,i}}, & E_{d_i} > 0. \\ 0, & E_{d,i} = 0, \end{cases} \tag{5.19}$$

with the amount of an enzyme in the robust solution as $E_{r,i}$ and using the regular deFBA solution as reference $E_{d,i}$. For almost all biomass species this multiplier ranges between 0.85 and 1.15, which we mostly appoint to numerical difference in the solutions. But there a few very interesting outliers. We find three enzymes, whose expression rates are noticeable larger in the rdeFBA compared to the sdeFBA:

- The multiplier for E_r_0491_2_c03 reaches values around 350, meaning the robust solution heavily relies on reaction r_0491, which is a glycerol-3-phosphate dehydrogenase, while the regular solution uses this only marginally. There exist an isoenzyme in this case, which is not utilized in either solution.
- The rates for threonine aldolase r_1040 are slightly increased in the robust solution with the multiplier reaching 1.21 for the enzyme E_r_1040_c03.
- The enzyme E_r_0216_c03 has an attached multiplier of $x = 2.242$, which catalyzes an aspartate transaminase in the cytoplasm.

While the change in the rate of r_0491 shows a clear shift in the dynamic strategy of the robust solution, the overall impact of the other two is not obvious. Especially, the utilization of the aspartate transaminase is interesting as we can find an opposing strategy in the deFBA solution, as we will show later in the text.

Looking at smaller multipliers with $x < 0.85$, we find 17 enzymes. Of these, seven enzymes are isoenzymes corresponding in function to seven of the enzymes only expressed in the robust solution. Thus, this proves that the solutions are mostly using identical reaction pathways but using different enzymes to realize those. Why this difference occurs is yet to be investigated.

From the remaining enzymes with small multipliers, the enzyme E_r_0217_c11 stands out as it catalyzes an aspartate transaminase in the mitochondrion. The robust solution prefers therefore the production of L-aspartate in the cytoplasm, while the regular deFBA focuses on the production inside the mitochondrion coupled with a transport reaction. The other 9 enzymes are all connected to the energy production using the respiratory cycle in the mitochondrion. We listed all enzymes discussed in Table B.3.

As the solutions mainly differentiate in the oxygen consumption, we plotted the oxygen uptake flux r_1979 over time in Figure 5.8. To be able to compare the different methods, we weigh the rates of the uptake reaction with the current biomass. The robust method clearly decreases the utilization of oxygen until reaching a new steady state after 3 hours. The non-robust methods on the other hand keep up the oxygen uptake predicted by the RBA. The minor variations of the uptake rate in the regular deFBA predictions can be traced back to numerical inaccuracies leading to minimal variations in the biomass composition.

To summarize, we have identified a strategy for robust growth using less oxygen than the predictions from regular deFBA. Interestingly, the output of ethanol is very much the same for both solutions so they are indistinguishable from the outside without

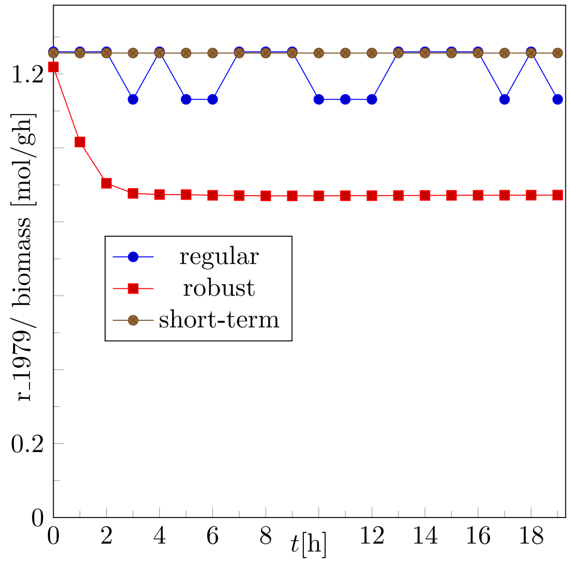


Figure 5.8: Oxygen uptake rate for the different methods weighted by their biomass. While the robust solution slowly decreases the oxygen uptake until it reaches a new equilibrium, the non-robust solution sustain the oxygen uptake predicted by the RBA.

measuring either glycerin uptake or the oxygen transfer rate. To verify which strategy is implemented by wild-type yeast, we will have to conduct *in vitro* experiments and measure the gene expressions in the cells.

5.7 Conclusion

The robust deFBA in its current form is a very powerful tool, but we still have to get our hands on experimental results generated in a very dynamic environment to compare these to our predictions. Our colleagues at the chair of Frank Bruggeman at the VU Amsterdam are currently working on experiments with yeast using a sudden switch from aerobic to anaerobic conditions. Very much alike to our presented example in the *carbon core* model. We are looking forward to compare these experimental results to the ones generated with the robust deFBA.

Furthermore, we must further investigate the impact of robust horizon and preparation time on the quality of results. Currently, we interpret these as pure tuning parameters, which have to be determined by hand. But some inherit qualities must exist in robust solutions, which we can exploit to choose these values systematically as with the prediction horizon in the short-term deFBA.

On the more technical side, we have experienced some problems with running the robust deFBA on genome-scale models as the problem size explodes in comparison to a regular sdeFBA. While the simplified midpoint collocation method helps to keep the computational cost low, the linear programs themselves can be very bad conditioned.

In the simulations for the *deFBA_yeast* model for example a single iteration step in the short-term deFBA took on average 10min, while a robust iteration step took on average 30 min with some iterations taking over an hour to solve. We believe this is to some factor caused by us using a local solver, `gurobi` [43] in this case, and having multiple local solutions in close vicinity to each other. A possible solution might be using `SoPlex` [114], which provides the possibility to increase the accuracy of the solution at the cost of an increased number of calculations needed.

In the future we plan also plan to investigate how the models behave if we delete all but one isoenzyme in the network, which is a likely reason for the existence of closely neighboring local solutions.

6 Conclusions

This study is dedicated to the simulation of metabolic networks as encountered in single cell organisms. As presented in the second chapter, metabolic networks are intensively studied under the lens of resource allocation since the work [75] in 2009. Two major methods for the simulation of metabolic networks coupled with protein expression arose from these investigations: Firstly, we have the RBA [39] optimizing the reaction rates and gene expression levels for a maximal growth under given external conditions. While the RBA allows to predict growth rates, preferred carbon sources, byproducts of metabolism, and enzyme levels it is limited to a fixed environment.

The second class of methods are dynamic resource allocation methods like the dynamic FBA [70] and the deFBA [110], which presents the foundation for this thesis. These lift the resource allocation problem to a dynamic setting and allow therefore to study the adaptation process inside the cells when the nutritional situation changes. Therefore, we can observe at which point the cells start adapting to the new situation and how the enzyme levels change over time. The dynamic nature of deFBA allows for new applications not previously possible. For example, the work [51] implements a bi-level optimization problem with deFBA at its core to maximize the product yield.

But with the deFBA still being very new, guidelines for the creation and evaluation of these kinds of models had been established yet. We focused in the third chapter of this work, to fully formalize the handling of deFBA models by introducing a set of model standards on the creation, exchange and simulation of the models. For the creation of the model, we rely on existing metabolic reconstructions containing gene annotations for metabolic reactions. Typically, the user will be able to find FBA models as a starting point for the construction of deFBA models. While most of this construction process is quite straightforward, like the addition of enzymes and enzyme producing reactions by interpretation of the gene annotations, two aspects are indeed complex to realize: The construction of biomass composition constraints derived from the original biomass reactions from the FBA model and the collection of catalytic constants or turnover numbers. We were able to identify a systematic approach for the biomass constraints as discussed in Chapter 3. The data collection problem was only partially solved as we could identify suitable databases like BRENDA [97] and BioNumbers [74], but failed to implement a fully automatic data gathering tool so far. The problem is centered around two facts: Firstly, automatically generated entries in BRENDA can not be fully trusted as they might contain a mixture of experimentally determined parameter values under vastly diverging conditions. Secondly, to our knowledge there exists no unique identifier mapping isoenzymes and their catalytic constants to the corresponding reactions. From our perspective the problem originates in the E.C. numbers, which do not take the existence of isoenzymes into account. Yet, E.C. numbers are widely used to order the databases. Secondly, a full text search for reaction names instead is very challenging as reaction names do not follow a set standard. These issues must be solved in the future by either establishing suitable conventions or developing advanced methods for text search in online databases.

To enable the user to share their models, we also introduced a new extension to SBML [48] called the *Resource Allocation Modeling* (RAM) extension [65]. This standard is sufficient to guarantee that models can be imported and exported to SBML in every detail. The way we encode model data was chosen to easily check for model or encoding errors. In the future we hope to extend this standard further, to be able to represent more types of resource allocation problems in RAM.

Another large contribution of this thesis is the introduction of the software package `deFBA-Python` [64]. This package contains every tool needed to successfully analyze deFBA models starting from import/export to RAM and ending with the newest extension for robust optimization. In the near future we plan to publish the automatic generation of deFBA models from FBA models as a new component in `deFBA-Python`.

In this thesis we extend the regular deFBA problem in two major ways. In Chapter 4, we introduce the short-term deFBA, which adapts the idea of a receding prediction horizon from MPC. It was necessary to adapt this as we encountered two flaws in the regular deFBA. The problem size for deFBA class problems can become so large that regular personal computers can not solve them. This is especially true for application in need of large end-times. Therefore, we are able to drastically lower the computational cost by introducing a small prediction horizon. The other flaw is cased by the fixed end-time in the objective of the deFBA as the chosen end-time can have an impact on the quality of the solution (cf. Section 4.1.1).

This problem directly translates to the impact of the chosen prediction horizon in the short-term deFBA. We derived a systematic way to calculate a prediction horizon which ensures, that the solutions can experience exponential growth phases.

The final contribution is the introduction of the robust deFBA, which extends the short-term deFBA to include uncertainties in the availability of nutrients. We adapt in this new method the ideas of the multi-stage Model Predictive Control [68], which describes the effects of the uncertainties on the model by a set of deterministic scenarios. As this formulation is quite simple and can be presented in the form of a scenario tree, the challenge in applying this method is choosing a suitable way to model the uncertain environment. We present ways to include two kinds of uncertain environments to predict how a population adapts if nutrients availability suddenly shifts: Either a nutrient is available and might suddenly be unavailable or a nutrient is not available and might become available again.

The problem with formulating these cases, is that they must keep the optimization problem feasible in the presence of maintenance reactions. Hence, our solutions might seem unnecessary complicated. Nevertheless, we apply this method in a situation where the availability of oxygen is highly dynamic and show that the results from the robust deFBA present a evolutionary advantage compared to regular short-term deFBA results. Therefore, we conclude that the robust deFBA will be beneficial in studying wild-types and understanding how the internal regulation of the metabolism expedites robustness.

A Models

The deFBA models used in this work are available as SBML files using the RAM extension. These files can be obtained via two ways. Firstly, they are part of the `defba-Python` package [64] and can be found in the examples folder. Secondly, we uploaded public copies of the model files to Fairdom HUB[1] and obtained digital object identifier (doi) numbers for each model. The respective numbers are listed in Table A.1. Additionally, we print the *resalloc* model in the next section as example for the usage of SBML and RAM.

Table A.1: List of models used in this work and the doi numbers linking to the SBML files containing the models.

Model name	doi address
carbon-core	`https://doi.org/10.15490/fairdomhub.1.model.557.1`
enzymatic-growth	`https://doi.org/10.15490/fairdomhub.1.model.556.1`
resalloc	`https://doi.org/10.15490/fairdomhub.1.model.558.1`
Yeast_deFBA	`https://doi.org/10.15490/fairdomhub.1.model.555.1`

A.1 Resalloc model

The resource allocation model, *resalloc* for short, was created to showcase all functions of the RAM [65] extension for SBML.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3"
 version="2" xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version2"
 fbc:required="false">
 <model id="resalloc" name="A resource allocation model" fbc:strict="false">

  <listOfCompartments>
   <compartment id="external" name="extracellular compartment. nutrients,
    waste, etc." spatialDimensions="3" size="1" constant="true"/>
   <compartment id="cytosol" name="cytosol. Collecting all non external
    components" spatialDimensions="3" size="1" constant="true"/>
  </listOfCompartments>
```

---

[1] fairdomhub.org

```
<listOfSpecies>
 <species id="N" compartment="external" initialAmount="2000"
  constant="false" boundaryCondition="true" hasOnlySubstanceUnits="true">
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:species ram:molecularWeight="zero" ram:objectiveWeight="zero"
     ram:biomassPercentage="zero" ram:speciesType="extracellular"/>
   </ram:RAM>
  </annotation>
 <species id="A" compartment="cytosol" initialAmount="0"
 constant="false" boundaryCondition="false" hasOnlySubstanceUnits="true">
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:species ram:molecularWeight="zero" ram:objectiveWeight="zero"
     ram:biomassPercentage="zero" ram:speciesType="metabolite"/>
   </ram:RAM>
  </annotation>
 <species id="M" compartment="cytosol" initialAmount="0.1" constant="false"
 boundaryCondition="false" hasOnlySubstanceUnits="true">
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:species ram:molecularWeight="weighM" ram:objectiveWeight="oWeightM"
    ram:biomassPercentage="zero" ram:speciesType="storage"/>
   </ram:RAM>
  </annotation>
 </species>
 <species id="E"  compartment="cytosol"   initialAmount="0.1"  constant="false"
 boundaryCondition="false" hasOnlySubstanceUnits="true">
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:species ram:molecularWeight="weighE" ram:objectiveWeight="oWeightE"
    ram:biomassPercentage="zero" ram:speciesType="enzyme"/>
   </ram:RAM>
  </annotation>
 </species>
</listOfSpecies>

<listOfParameters>
 <parameter constant="true" id="zero"   value="0"  />
 <parameter constant="true" id="weighM"  value="150" />
 <parameter constant="true" id="weighE"  value="100" />
 <parameter constant="true" id="oWeightM" value="150" />
 <parameter constant="true" id="oWeightE" value="100" />
 <parameter constant="true" id="kcatA"  value="150" />
 <parameter constant="true" id="kcatE"  value="1"  />
 <parameter constant="true" id="kcatM"  value="2"  />
</listOfParameters>

<fbc:listOfGeneProducts>
 <fbc:geneProduct fbc:id="E" fbc:label="enzymes" fbc:associatedSpecies="E"/>
```

```
</fbc:listOfGeneProducts>

<listOfReactions>
 <reaction id="VA" reversible="false" fast="false">
  <fbc:geneProductAssociation fbc:id="Enzymes">
    <fbc:geneProductRef fbc:geneProduct="E" />
  </fbc:geneProductAssociation>
  <listOfReactants>
   <speciesReference species="N" stoichiometry="1" constant="true"/>
  </listOfReactants>
  <listOfProducts>
   <speciesReference species="A" stoichiometry="1" constant="true"/>
  </listOfProducts>
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:reaction ram:kcatForward="kcatA" ram:kcatBackward="zero"
    ram:maintenanceScaling="zero"/>
   </ram:RAM>
  </annotation>
 </reaction>
 <reaction id="VE" reversible="false" fast="false">
  <fbc:geneProductAssociation fbc:id="Enzymes">
    <fbc:geneProductRef fbc:geneProduct="E" />
  </fbc:geneProductAssociation>
  <listOfReactants>
   <speciesReference species="N" stoichiometry="100" constant="true"/>
   <speciesReference species="A" stoichiometry="100" constant="true"/>
  </listOfReactants>
  <listOfProducts>
   <speciesReference species="E" stoichiometry="1" constant="true"/>
  </listOfProducts>
  <annotation>
   <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
    <ram:reaction ram:kcatForward="kcatE" ram:kcatBackward="zero"
    ram:maintenanceScaling="zero"/>
   </ram:RAM>
  </annotation>
 </reaction>
 <reaction id="VM" reversible="false" fast="false">
  <fbc:geneProductAssociation fbc:id="Enzymes">
    <fbc:geneProductRef fbc:geneProduct="E" />
  </fbc:geneProductAssociation>
  <listOfReactants>
   <speciesReference species="N" stoichiometry="100" constant="true"/>
    <speciesReference species="A" stoichiometry="100" constant="true"/>
  </listOfReactants>
  <listOfProducts>
   <speciesReference species="M" stoichiometry="1" constant="true"/>
  </listOfProducts>
<annotation>
```

```
     <ram:RAM xmlns:ram="https://www.fairdomhub.org/sops/304">
      <ram:reaction ram:kcatForward="kcatM" ram:kcatBackward="zero"
      ram:maintenanceScaling="zero"/>
     </ram:RAM>
    </annotation>
   </reaction>
  </listOfReactions>
 </model>
</sbml>
```

B Results for deFBA_yeast model

Here we collect the detailed analysis of the results from the *deFBA_yeast* model.

Table B.2: Enzyme usage comparing predictions from the static RBA to the regular and robust deFBA.

Not predicted by RBA but used

in regular and robust solution	only robust solution
E_r_0525_c03	E_r_1038_2_c03
E_r_1052_c03	E_r_0489_1_c03
E_r_0337_c03	E_r_1135_1_r_1166_7_c03_c06
E_r_0842_c03	E_r_0486_2_c03
E_r_0326_c03	E_r_1172_c03
E_r_1014_2_c03_c09	E_r_0565_1_c03
E_r_0243_c03	E_r_0570_1_r_0912_1_c03
	E_r_0883_1_c03

Predicted but not used

regular	robust
E_r_0438_1_c03_c11	$\emptyset$

Table B.3: Enzymes with extreme multipliers $x_i = E_{r,i}/E_{d,i}$ greater 1.15 or less then 0.85 at time $t = 16.333$h.

Enzyme	Multiplier
E_r_0491_2_c03	341.29
E_r_0216_c03	2.242
E_r_1040_c03	1.21
E_r_0226_1_co3_c11	0.681
E_r_0439_1_c03_c11	0.676
E_r_0438_3_c03_c11	0.676
E_r_0773_c11	0.676
E_r_1110_2_c03_c11	0.661
E_r_E_r_0569_c11	0.643
E_r_1099_1_r_2131_1_c03_c11	0.579
E_r_1118_r_1194_c03_c11	0.495
E_r_0714_c03	0.472
E_r_0713_c11	0.458
E_r_0217_c11	0.195
E_r_0486_3_c03	0.029
E_r_1135_5_c03_c06	0.014
E_r_0883_2_c03	0.009
E_r_1038_3_c03	0.008
E_r_0570_2_r0912_c03	0.005
E_r_0565_3_c03	0.005

.

Bibliography

[1] Bruce Alberts, Alexander Johnson, Julian Lewis, Peter Walter, Martin Raff, and Keith Roberts. Molecular biology of the cell 4th edition, 2002.

[2] M.S. Andersen, J. Dahl, and L. Vandenberghe. CVXOPT: A Python package for convex optimization, version 1.1.9., 2016. Available at `http://cvxopt.org/index.html`.

[3] Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl_1):D115–D119, 2004.

[4] Meric Ataman, Daniel F Hernandez Gardiol, Georgios Fengos, and Vassily Hatzimanikatis. redgem: Systematic reduction and analysis of genome-scale metabolic reconstructions for development of consistent core metabolic models. *PLoS computational biology*, 13(7):e1005444, 2017.

[5] Hnin W Aung, Susan A Henry, and Larry P Walker. Revising the representation of fatty acid, glycerolipid, and glycerophospholipid metabolism in the consensus model of yeast metabolism. *Industrial biotechnology*, 9(4):215–228, 2013.

[6] Arren Bar-Even, Elad Noor, Yonatan Savir, Wolfram Liebermeister, Dan Davidi, Dan S Tawfik, and Ron Milo. The moderately efficient enzyme: evolutionary and physicochemical trends shaping enzyme parameters. *Biochemistry*, 50(21):4402–4410, 2011.

[7] Bryson D Bennett, Elizabeth H Kimball, Melissa Gao, Robin Osterhout, Stephen J Van Dien, and Joshua D Rabinowitz. Absolute metabolite concentrations and implied enzyme active site occupancy in escherichia coli. *Nature chemical biology*, 5(8):593, 2009.

[8] Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic acids research*, 41(D1):D36–D42, 2012.

[9] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218, 2007.

[10] Lorenz T Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.

[11] Aarash Bordbar, Jonathan M Monk, Zachary A King, and Bernhard O Palsson. Constraint-based models predict metabolic and associated cellular functions. *Nature Reviews Genetics*, 15(2):107, 2014.

[12] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. LibSBML: An API library for SBML. *Bioinformatics*, 24(6):880–881, 2008.

[13] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66, 1997.

[14] George Edward Briggs and John Burdon Sanderson Haldane. A note on the kinetics of enzyme action. *Biochemical journal*, 19(2):338, 1925.

[15] Guy C Brown. Total cell protein concentration as an evolutionary constraint on the metabolic control distribution in cells. *Journal of theoretical biology*, 153(2):195–203, 1991.

[16] Anthony P Burgard, Shankar Vaidyaraman, and Costas D Maranas. Minimal reaction sets for escherichia coli metabolism under different growth requirements and uptake environments. *Biotechnology progress*, 17(5):791–797, 2001.

[17] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. K nitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[18] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[19] Peter J Campo and Manfred Morari. Robust model predictive control. In *American Control Conference, 1987*, pages 1021–1026, 1987.

[20] Thalia E Chan, Michael PH Stumpf, and Ann C Babtie. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell systems*, 5(3):251–267, 2017.

[21] Vijayalakshmi Chelliah, Nick Juty, Ishan Ajmera, Raza Ali, Marine Dumousseau, Mihai Glont, Michael Hucka, Gaël Jalowicki, Sarah Keating, Vincent Knight-Schrijver, Audald Lloret-Villas, Kedar Nath Natarajan, Jean-Baptiste Pettit, Nicolas Rodriguez, Michael Schubert, Sarala M. Wimalaratne, Yangyang Zhao, Henning Hermjakob, Nicolas Le Novère, and Camille Laibe. BioModels: ten-year anniversary. *Nucl. Acids Res.*, 43:D542–D548, 2015.

[22] J Michael Cherry, Caroline Adler, Catherine Ball, Stephen A Chervitz, Selina S Dwight, Erich T Hester, Yankai Jia, Gail Juvik, TaiYun Roe, Mark Schroeder, et al. Sgd: Saccharomyces genome database. *Nucleic acids research*, 26(1):73–79, 1998.

[23] Michaela Conrad, Joep Schothorst, Harish Nag Kankipati, Griet Van Zeebroeck, Marta Rubio-Texeira, and Johan M Thevelein. Nutrient sensing and signaling in the yeast saccharomyces cerevisiae. *FEMS microbiology reviews*, 38(2):254–299,

2014.

[24] GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 24.2.1. Washington, DC, USA, 2013.

[25] Mélanie Courtot, Nick Juty, Christian Knüpfer, Dagmar Waltemath, Anna Zhukova, Andreas Dräger, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, et al. Controlled vocabularies and semantics in systems biology. *Molecular systems biology*, 7(1):543, 2011.

[26] Dan Davidi, Elad Noor, Wolfram Liebermeister, Arren Bar-Even, Avi Flamholz, Katja Tummler, Uri Barenholz, Miki Goldenfeld, Tomer Shlomi, and Ron Milo. Global characterization of in vivo enzyme catalytic rates and their correspondence to in vitro kcat measurements. *Proceedings of the National Academy of Sciences*, 113(12):3401–3406, 2016.

[27] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 273–316. Springer, 2005.

[28] Tao Ding, Yuan Hu, and Zhaohong Bie. Multi-stage stochastic programming with nonanticipativity constraints for expansion of combined power and natural gas systems. *IEEE Transactions on Power Systems*, 33(1):317–328, 2018.

[29] Zheng Eelderink-Chen, Gabriella Mazzotta, Marcel Sturre, Jasper Bosman, Till Roenneberg, and Martha Merrow. A circadian clock in saccharomyces cerevisiae. *Proceedings of the National Academy of Sciences*, page 200907902, 2010.

[30] Steven Eker, Markus Krummenacker, Alexander G Shearer, Ashish Tiwari, Ingrid M Keseler, Carolyn Talcott, and Peter D Karp. Computing minimal nutrient sets from metabolic networks via linear constraint solving. *BMC bioinformatics*, 14(1):114, 2013.

[31] Péter Érdi and János Tóth. *Mathematical models of chemical reactions: theory and applications of deterministic and stochastic models.* Manchester University Press, 1989.

[32] Philipp Erdrich, Ralf Steuer, and Steffen Klamt. An algorithm for the reduction of genome-scale metabolic network models to meaningful core models. *BMC systems biology*, 9(1):48, 2015.

[33] Eliane Fischer and Uwe Sauer. Large-scale in vivo flux analysis shows rigidity and suboptimal performance of bacillus subtilis metabolism. *Nature genetics*, 37(6):636, 2005.

[34] Robert J Flassig, Melanie Fachet, Kai Höffner, Paul I Barton, and Kai Sundmacher. Dynamic flux balance modeling to increase the production of high-value compounds in green microalgae. *Biotechnology for biofuels*, 9(1):165, 2016.

[35] Marco Fondi and Pietro Liò. Genome-scale metabolic network reconstruction. In *Bacterial Pangenomics*, pages 233–256. Springer, 2015.

[36] Virginie Gabrel, Cécile Murat, and Aurélie Thiele. Recent advances in robust op-

timization: An overview. *European journal of operational research*, 235(3):471–483, 2014.

[37] Carlos Eduardo García Sánchez and Rodrigo Gonzalo Torres Sáez. Comparison and analysis of objective functions in flux balance analysis. *Biotechnology progress*, 30(5):985–991, 2014.

[38] Anne Goelzer and Vincent Fromion. Towards the modular decomposition of the metabolic network. In *A Systems Theoretic Approach to Systems and Synthetic Biology I: Models and System Characterizations*, pages 121–152. Springer, 2014.

[39] Anne Goelzer, Vincent Fromion, and Gérard Scorletti. Cell design in bacteria as a convex optimization problem. *Automatica*, 47(6):1210–1218, 2011.

[40] Anne Goelzer, Jan Muntel, Victor Chubukov, Matthieu Jules, Eric Prestel, Rolf Nölker, Mahendra Mariadassou, Stéphane Aymerich, Michael Hecker, Philippe Noirot, et al. Quantitative prediction of genome-wide resource allocation in bacteria. *Metabolic engineering*, 32:232–243, 2015.

[41] Jose A Gomez, Kai Höffner, and Paul I Barton. Dfbalab: a fast and reliable matlab code for dynamic flux balance analysis. *BMC bioinformatics*, 15(1):409, 2014.

[42] P. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42:523–533, 2006.

[43] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.

[44] Michael J Hadjiyiannis, Paul J Goulart, and Daniel Kuhn. An efficient method to estimate the suboptimality of affine controllers. *IEEE Transactions on Automatic Control*, 56(12):2841–2853, 2011.

[45] Arno J Hanekom. *Generic kinetic equations for modelling multisubstrate reactions in computational systems biology.* PhD thesis, Stellenbosch: University of Stellenbosch, 2006.

[46] Benjamin D Heavner, Kieran Smallbone, Nathan D Price, and Larry P Walker. Version 6 of the consensus yeast metabolic network refines biochemical coverage and improves model performance. *Database*, 2013, 2013.

[47] Andreas Hoppe, Sabrina Hoffmann, and Hermann-Georg Holzhütter. Including metabolite concentrations into flux balance analysis: thermodynamic realizability as a constraint on flux distributions in metabolic networks. *BMC systems biology*, 1(1):23, 2007.

[48] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[49] Michael Hucka and Lucian P Smith. Sbml level 3 package: Groups, version 1 release 1. *Journal of integrative bioinformatics*, 13(3):8–29, 2016.

[50] John L Ingraham, Ole Maaløe, Frederick Carl Neidhardt, et al. *Growth of the bacterial cell.* Sinauer Associates, 1983.

[51] Banafsheh Jabarivelisdeh and Steffen Waldherr. Improving bioprocess productivity using constraint-based models in a dynamic optimization scheme. *IFAC-PapersOnLine*, 49(26):245–251, 2016.

[52] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}, 2014.

[53] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[54] Jonathan R Karr, Jayodita C Sanghvi, Derek N Macklin, Miriam V Gutschow, Jared M Jacobs, Benjamin Bolival Jr, Nacyra Assad-Garcia, John I Glass, and Markus W Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, 2012.

[55] Hassan K Khalil. *Nonlinear systems, 3rd.* Pearson, 2002.

[56] Ali Khodayari and Costas D Maranas. A genome-scale escherichia coli kinetic metabolic model k-ecoli457 satisfying flux data for multiple mutant strains. *Nature communications*, 7:13806, 2016.

[57] Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2015.

[58] Edda Klipp and Reinhart Heinrich. Competition for enzymes in metabolic pathways:: Implications for optimal distributions of enzyme concentrations and for the distribution of flux control. *Biosystems*, 54(1-2):1–14, 1999.

[59] HC Lange and JJ Heijnen. Statistical reconciliation of the elemental and molecular biomass composition of saccharomyces cerevisiae. *Biotechnology and bioengineering*, 75(3):334–344, 2001.

[60] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(Database issue):D689–D691, Jan 2006.

[61] Joshua A Lerman, Daniel R Hyduke, Haythem Latif, Vasiliy A Portnoy, Nathan E Lewis, Jeffrey D Orth, Alexandra C Schrimpe-Rutledge, Richard D Smith, Joshua N Adkins, Karsten Zengler, et al. In silico method for modelling metabolism and gene product expression at genome scale. *Nature communica-*

tions, 3:929, 2012.

[62] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I. Stefan, Jacky L. Snoep, Michael Hucka, Nicolas Le Novère, and Camille Laibe. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4:92, Jun 2010.

[63] Youdong Lin and Linus Schrage. The global solver in the lindo api. *Optimization Methods & Software*, 24(4-5):657–668, 2009.

[64] Henning Lindhorst. *deFBA-Python: A toolbox for simulating enzyme controlled metabolic networks*. Institute for Automation Engineering, Otto-von-Guericke-Universität, Magdeburg, Germany, 2017.

[65] Henning Lindhorst, Alexandra-M. Reimers, Alexander Bockmayr, and Steffen Waldherr. RAM: An annotation standard for SBML Level 3, 2017.

[66] Henning Lindhorst, Alexandra-M. Reimers, and Steffen Waldherr. Dynamic modeling of enzyme controlled metabolic networks using a receding time horizon. *Proceedings on 10th IFAC International Symposium on Advanced Control of Chemical Processes*, 51(1):203–208, July 2018.

[67] S. Lucia, J. Andersson, H. Brandt, M. Diehl, and S. Engell. Handling uncertainty in economic nonlinear model predictive control: a comparative case-study. *Journal of Process Control*, 24:1247–1259, 2014.

[68] Sergio Lucia, T Finkler, Dahn Basak, and Sebastian Engell. A new robust nmpc scheme and its application to a semi-batch reactor example. *IFAC Proceedings Volumes*, 45(15):69–74, 2012.

[69] Sergio Lucia, Tiago Finkler, and Sebastian Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306 – 1319, 2013.

[70] Radhakrishnan Mahadevan, Jeremy S Edwards, and Francis J Doyle. Dynamic flux balance analysis of diauxic growth in Escherichia coli. *Biophysical Journal*, 83(3):1331–1340, 2002.

[71] George Mavrotas and Danae Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[72] Leonor Menten and MI Michaelis. Die kinetik der invertinwirkung. *Biochem Z*, 49:333–369, 1913.

[73] Ron Milo, Paul Jorgensen, Uri Moran, Griffin Weber, and Michael Springer. Bionumbersâ€”the database of key numbers in molecular and cell biology. *Nucleic acids research*, 38(suppl_1):D750–D753, 2009.

[74] Ron Milo, Paul Jorgensen, Uri Moran, Griffin Weber, and Michael Springer. BioNumbers - the database of key numbers in molecular and cell biology. *Nucleic*

Acids Research, 38(suppl 1):D750–D753, 2010.

[75] Douwe Molenaar, Rogier Van Berlo, Dick De Ridder, and Bas Teusink. Shifts in growth strategies reflect tradeoffs in cellular economics. *Molecular systems biology*, 5(1):323, 2009.

[76] Jacques Monod. The growth of bacterial cultures. *Annual Reviews in Microbiology*, 3(1):371–394, 1949.

[77] Akihiro Nakao, Maki Yoshihama, and Naoya Kenmochi. Rpg: the ribosomal protein gene database. *Nucleic acids research*, 32(suppl_1):D168–D170, 2004.

[78] OM Neijssel and DW Tempest. The role of energy-spilling reactions in the growth ofklebsiella aerogenes nctc 418 in aerobic chemostat culture. *Archives of microbiology*, 110(2-3):305–311, 1976.

[79] David L Nelson, Albert L Lehninger, and Michael M Cox. *Lehninger principles of biochemistry*. Macmillan, 2008.

[80] ChiamYu Ng, Moo-young Jung, Jinwon Lee, and Min-Kyu Oh. Production of 2, 3-butanediol in saccharomyces cerevisiae by in silico aided metabolic engineering. *Microbial cell factories*, 11(1):68, 2012.

[81] Ali Nikdel, Richard D Braatz, and Hector M Budman. A systematic approach for finding the objective function and active constraints for dynamic flux balance analysis. *Bioprocess and biosystems engineering*, 41(5):641–655, 2018.

[82] Edward J O'brien, Joshua A Lerman, Roger L Chang, Daniel R Hyduke, and Bernhard Ø Palsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular systems biology*, 9(1):693, 2013.

[83] Andrea Ocone, Laleh Haghverdi, Nikola S Mueller, and Fabian J Theis. Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics*, 31(12):i89–i96, 2015.

[84] Brett G Olivier and Frank T Bergmann. The systems biology markup language (sbml) level 3 package: Flux balance constraints. *Journal of integrative bioinformatics*, 12(2):660–690, 2015.

[85] Diego A Oyarzún and Guy-Bart V Stan. Synthetic gene circuits for metabolic control: design trade-offs and constraints. *Journal of The Royal Society Interface*, 10(78):20120671, 2013.

[86] Satchidananda Panda, Marina P Antoch, Brooke H Miller, Andrew I Su, Andrew B Schook, Marty Straume, Peter G Schultz, Steve A Kay, Joseph S Takahashi, and John B Hogenesch. Coordinated transcription of key pathways in the mouse by the circadian clock. *Cell*, 109(3):307–320, 2002.

[87] WR Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods in enzymology*, 183(183):63–98, 1990.

[88] Doraiswami Ramkrishna and Hyun-Seob Song. *Cybernetic Modeling for Biore-*

action Engineering. Cambridge University Press, 2018.

[89] M Razzaghi, J Nazarzadeh, and KY Nikravesh. A collocation method for optimal control of linear systems with inequality constraints. *Mathematical Problems in Engineering*, 3(6):503–515, 1998.

[90] Alexandra-M. Reimers, Henning Knoop, Alexander Bockmayr, and Rald Steuer. Cellular trade-offs and optimal resource allocation during cyanobacterial diurnal growth. *Proceedings of the National Academy of Sciences*, 114(31):E6457–E6465, 2017.

[91] Alexandra-M Reimers, Henning Lindhorst, and Steffen Waldherr. A protocol for generating and exchanging (genome-scale) metabolic resource allocation models. *Metabolites*, 7(3):47, 2017.

[92] Alexandra-Mirela Reimers. *Understanding metabolic regulation and cellular resource allocation through optimization.* PhD thesis, Freie Universität Berlin, 2017.

[93] Mark J Rentmeesters, Wei K Tsai, and Kwei-Jay Lin. A theory of lexicographic multi-criteria optimization. In *Engineering of Complex Computer Systems, 1996. Proceedings., Second IEEE International Conference on*, pages 76–79. IEEE, 1996.

[94] Annika Röhl and Alexander Bockmayr. A mixed-integer linear programming approach to the reduction of genome-scale metabolic networks. *BMC bioinformatics*, 18(1):2, 2017.

[95] Marco Rügen, Alexander Bockmayr, and Ralf Steuer. Elucidating temporal resource allocation and diurnal dynamics in phototrophic metabolism using conditional fba. *Scientific reports*, 5:15247, 2015.

[96] Milton H Saier Jr. Multiple mechanisms controlling carbon metabolism in bacteria. *Biotechnology and bioengineering*, 58(2-3):170–174, 1998.

[97] Ida Schomburg, Antje Chang, Sandra Placzek, Carola Söhngen, Michael Rother, Maren Lang, Cornelia Munaretto, Susanne Ulas, Michael Stelzer, Andreas Grote, et al. Brenda in 2013: integrated reactions, kinetic data, enzyme function data, improved disease classification: new options and contents in brenda. *Nucleic acids research*, 41(D1):D764–D772, 2012.

[98] Ida Schomburg, Antje Chang, and Dietmar Schomburg. BRENDA, enzyme data and metabolic information. *Nucleic Acids Research*, 30(1):47–49, 2002.

[99] Robert Schuetz, Nicola Zamboni, Mattia Zampieri, Matthias Heinemann, and Uwe Sauer. Multidimensional optimality of microbial metabolism. *Science*, 336(6081):601–604, 2012.

[100] Pierre OM Scokaert and DQ Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic control*, 43(8):1136–1142, 1998.

[101] Shinichiro Shoji, Corey M Dambacher, Zahra Shajani, James R Williamson, and Peter G Schultz. Systematic chromosomal deletion of bacterial ribosomal protein genes. *Journal of molecular biology*, 413(4):751–761, 2011.

[102] IBM Software Solutions. IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, Last 2010.

[103] Reginald P Tewarson and P Reginald. Sparse matrices (part of the mathematics in science & engineering series), 1973.

[104] Ines Thiele and Bernhard Ø Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature protocols*, 5(1):93, 2010.

[105] Amlt Varma and Bemhard Palsson. Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Nature Biotechnology*, 12(10):994–998, 1994.

[106] Alexei Vazquez. *Overflow metabolism: from yeast to Marathon runners.* Academic Press, 2017.

[107] Carlos Vilas, Eva Balsa-Canto, Maria-Sonia G García, Julio R Banga, and Antonio A Alonso. Dynamic optimization of distributed biological systems using robust and efficient numerical techniques. *BMC systems biology*, 6(1):79, 2012.

[108] Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143. Springer, 1993.

[109] Steffen Waldherr and Henning Lindhorst. Optimality in cellular storage via the Pontryagin Maximum Principle. *Preprints of the 20th IFAC World Congress*, 20:10305–10311, 2017.

[110] Steffen Waldherr, Diego A Oyarzún, and Alexander Bockmayr. Dynamic optimization of metabolic networks coupled with gene expression. *Journal of Theoretical Biology*, 365:469–485, 2015.

[111] Kowda M Wasungu and Ronald E Simard. Growth characteristics of bakers' yeast in ethanol. *Biotechnology and bioengineering*, 24(5):1125–1134, 1982.

[112] Edwin C Webb et al. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* Number Ed. 6 in . Academic Press, 1992.

[113] Ulrike Wittig, Renate Kania, Martin Golebiewski, Maja Rey, Lei Shi, Lenneke Jong, Enkhjargal Algaa, Andreas Weidemann, Heidrun Sauer-Danzwith, Saqib Mir, et al. Sabio-rkâ€”database for biochemical reaction kinetics. *Nucleic acids research*, 40(D1):D790–D796, 2011.

[114] Roland Wunderling. Soplex: The sequential object-oriented simplex class library, 1997.

[115] R Young and Hans Bremer. Polypeptide-chain-elongation rate in Escherichia coli B/r as a function of growth rate. *Biochemical Journal*, 160(2):185–194, 1976.

List of symbols and abbreviations

Table B.4: Variables used in deFBA. Domain of all variables is $\mathbb{R}$. Each variable has a concentration based equivalent in bold font, e.g., $\mathbf{X}$ has the unit M = mol/L and V_x has M/h = mol/(L s). Elements of matrices are addressed via a small character, e.g. $H_c = (h_c)_{i,j}$.

Variable	**Dimension**	**Unit**	**Meaning**
S	$n \times m$	1	stoichiometric matrix
R	$n \times n$	1/h	dilution matrix
q	n	mol/h	fixed input vector
V	m	mol/h	Reaction rates, resp. fluxes $m = m_y + m_x + m_c + m_p$
V_y	m_y	mol/h	Exchange reactions
V_x	m_x	mol/h	Purely metabolic reaction
V_c	m_c	mol/h	Storage producing/depleting reactions
V_p	m_p	mol/h	Protein producing/depleting reactions
V_a	m_a	mol/h	Maintenance reactions
Y	n_y	mol	External species, e.g. oxygen, carbon sources
X	n_x	mol	Metabolites in quasi-steady-state, e.g. ATP, NADH
C	n_c	mol	Storage components, e.g. starch.
Q	n_q	mol	Quota components with no catalytic function
P	n_p	mol	Enzymatic macromolecules, enzymes, ribosome, etc.
B	1	g	Total biomass in the system $B = w^T P$
B_o	1	g	Objective biomass $B = w_o^T P$

Table B.6: Matrices used in deFBA. Elements of matrices are addressed via a small character, e.g. $H_c = (h_c)_{i,j}$.

Matrix	Dimension	Unit of entries	Meaning
H_c	$? \times m$	1/h	Enzyme capacity matrix containing the inverse k_{cat} values; number of constraints depends on reversibility of reactions.
H_f	$? \times n$	1	Filter matrix to determine which macromolecule catalyzes the reactions.
H_b	$? \times n_p$	g/mol	Biomass composition constraint matrix; rows correspond to percentage of total biomass
H_a	$? \times n$	1/mol h	Maintenance matrix
H_g	$? \times m$	1	Filter matrix to find correct maintenance flux

Table B.8: Index sets used to adress different types of species, reactions, etc.

Identifier	Meaning
$\mathcal{Y}$	Set associated to external species
$\mathcal{X}$	Set associated to metabolic species
$\mathcal{C}$	Set associated to storage
$\mathcal{Q}$	Set associated to quota components
$\mathcal{P}$	Set associated to macromolecules
a	Set associated to maintenance reactions
y	Set associated to exchange reactions
x	Set associated to metabolic reactions
c	Set associated to storage production
q	Set associated to quota component production
p	Set associated to macromolecule production

Table B.10: Parameters

Identifier	Unit	Meaning
k_{cat}	1/ h	Catalytic constant or turnover numbers. Defines effectiveness of enzymes.
ϕ	mol/g h	Maintenance coefficient scaling the maintenance reactions with regards to the total biomass.
v_{min}	mol/h	Lower bounds for flux rates. Assumes only values 0 or $-\infty$.
v_{max}	mol/h	Upper bounds for flux rates. Assumes only values 0 or ∞.
w	g/mol	Molecular weight of macromolecules
w_{o}	g/mol	Objective weight of macromolecules