GENOME SCALE METABOLIC MODEL RECONSTRUCTION & COMMUNITY DYNAMIC FLUX BALANCE ANALYSIS SIMULATIONS

Sample of Written Work

Abstract

Presented in this paper are sections of my modeling work, conducted within the context of the iGEM 2018 competition. First, the reconstruction protocol of a *Saccharomyces boulardii* genome scale metabolic model is briefly summarized. Next, the functionality and usage of COM-dFBA is presented. This is a RAVEN based MATLAB tool I developed to help carry out community dynamic flux balance analysis simulations.

Francisco Zorrilla
zorrilla@chalmers.se

Francisco Zorrilla
Sample of Written Work

Table of Contents

Saccharomyces boulardii Genome Scale Metabolic Model *2*
Community Dynamic Flux Balance Analysis 4
  COM-dFBA 5
    Initializing Models 6
    Initializing Environment and Kinetic Parameters 7
    Simulations 7
  Experimental Design 8
  Results Simulation 1: *S. boulardii* & Cancer Cells 9
  Results Simulation 2: Gut Community Interaction 10
Integrated Modeling 12
  Proof of Concept 12
  ***S. boulardii* Content in Pill** 13
Future Work 13
References 14

Saccharomyces boulardii Genome Scale Metabolic Model

The first step in our genome scale metabolic modeling approach was to obtain a *Saccharomyces boulardii* model. Unfortunately, there was no readily available model for this organism in the literature, so we opted to generate our own. Since it can take anywhere from months to years to develop a well curated model, we decided to generate a *Saccharomyces boulardii* model using the closely related and well curated *Saccharomyces cerevisiae* GEM. Having access to this GEM allowed us to use a homology-based approach to obtain the desired *Saccharomyces boulardii* GEM with the help of the RAVEN toolbox (Agren, 2013). Please refer to the script reconstruction.m to see our MATLAB code for this GEM reconstruction. The script is divided into the following sections:

1. Install RAVEN & Set Up Working Directory

We downloaded and installed the RAVEN toolbox. We used RAVEN 2.0.2. The exact version of RAVEN used for reconstruction, curation, and simulations is also available on our GitHub page. This GitHub repository also contains our community dFBA framework, the genome scale metabolic models we employed, and our simulation results.

2. Obtain Protein FASTA Files & Perform BLAST

Use following links to obtain protein FASTA files for our organisms of interest:

- *Saccharomyces boulardii*
- *Saccharomyces cerevisiae*

We ran a bi-directional BLAST using the FASTA files for our two organisms to obtain a BLAST structure. This structure contains information about which genes, and therefore reactions, from the template model are present in the target model.

3. Obtain High Quality *Saccharomyces cerevisiae* GEM

Download the yeast GEM here. We checked that model is functional and can produce biomass.

4. Get *Saccharomyces boulardii* Homology Model

We used the RAVEN function *getModelFromHomology*, which takes a template model and BLAST structure as inputs. A model is obtained by essentially subsetting reactions from the template model which are also present in the target model, based on the similarity of their protein FASTA files.

5. *Saccharomyces boulardii* GEM Curation & Gap-Filling

Next, we added non-gene annotated reactions from *Saccharomyces cerevisiae* GEM. This includes spontaneous reactions, exchange reactions, pseudo-reactions, etc. which were not captured in the BLAST structure. The model is still not able to produce biomass, therefore gap-filling needs to be done. The RAVEN toolbox offers the *fillGaps* function, which uses a template model as a reference to fill gaps so that biomass can be produced.

6. Add Proteins of Interest to *Saccharomyces boulardii* Model

After the draft model for *Saccharomyces boulardii* was reconstructed, we added protein synthesis and exchange reactions for our peptides of interest: MFalpha2, Myrosinase, and P28. The protein synthesis reactions were constructed based on the amino acid content of each peptide, and account for the ATP needed for the addition of each amino acid each (~4 ATP/amino acid (Amthor, 2000)). While the RAVEN toolbox offers functions to add reactions and metabolites to a model, it does not readily construct the lengthy stoichiometric equations needed for each protein. Instead of writing these expressions by hand we wrote a function to do this, called *makeProteinEqn*, which is under review to be added to the RAVEN toolbox. This function takes amino acid sequences and protein names as inputs, while outputting balanced protein forming equations such as:

'26 Ala-tRNA(Ala)[c] + 20 Arg-tRNA(Arg)[c] + 31 Asn-tRNA(Asn)[c] + 36 Asp-tRNA(Asp)[c] + 11 Cys-tRNA(Cys)[c] + 13 Gln-tRNA(Gln)[c] + 24 Glu-tRNA(Glu)[c] + 39 Gly-tRNA(Gly)[c] + 15 His-tRNA(His)[c] + 28 Ile-tRNA(Ile)[c] + 44 Leu-tRNA(Leu)[c] + 35 Lys-tRNA(Lys)[c] + 10 Met-tRNA(Met)[c] + 30 Phe-tRNA(Phe)[c] + 28 Pro-tRNA(Pro)[c] + 35 Ser-tRNA(Ser)[c] + 31 Thr-tRNA(Thr)[c] + 11 Trp-tRNA(Trp)[c] + 36 Tyr-tRNA(Tyr)[c] + 24 Val-tRNA(Val)[c] + 2107 ATP[c] => 26 tRNA(Ala)[c] + 20 tRNA(Arg)[c] + 31 tRNA(Asn)[c] + 36 tRNA(Asp)[c] + 11 tRNA(Cys)[c] + 13 tRNA(Gln)[c] + 24 tRNA(Glu)[c] + 39 tRNA(Gly)[c] + 15 tRNA(His)[c] + 28 tRNA(Ile)[c] + 44 tRNA(Leu)[c] + 35 tRNA(Lys)[c] + 10 tRNA(Met)[c] + 30 tRNA(Phe)[c] + 28 tRNA(Pro)[c] + 35 tRNA(Ser)[c] + 31 tRNA(Thr)[c] + 11 tRNA(Trp)[c] + 36 tRNA(Tyr)[c] + 24 tRNA(Val)[c] + 2107 ADP[c] + 2107 phosphate[c] + Myrosinase[c]'

Refer to makeProteinEqn.m to see the MATLAB code for this function.

7. Simulations

Checked that proteins can be feasibly synthesized while biomass is produced.

8. Final Clean-up of *Saccharomyces boulardii* Model

Added information about model reconstruction, export model to excel, and save model as a .mat file.

Community Dynamic Flux Balance Analysis

The first step in designing a community dFBA framework is to determine which cell types are to be included. First and foremost, our community dFBA framework must contain the GEMs for colorectal cancer cells, human gut cells, and *S. boulardii*, since they are key aspects of our project. The question of how to model the complex system that is the gut microbiome is a more complicated problem. We decided to use GEMs of three representative species, in terms of composition and biosynthetic capabilities, which were taken from a gut microbiome study by Shoaie et al. (2013). In this paper, the authors generated GEMs for three key gut microbiome member species: *Bacteroidetes thetaiotaomicron*, *Eubacterium rectale* and *Methanobrevibacter smithii*, which represent the main phyla Bacteroidetes, Firmicutes, and Euryarchaeota, respectively. Below is a schematic depicting the system modeled by our community dynamic Flux Balance Analysis framework. Note that the schematic below corresponds to a scenario of a healthy person without colorectal cancer.

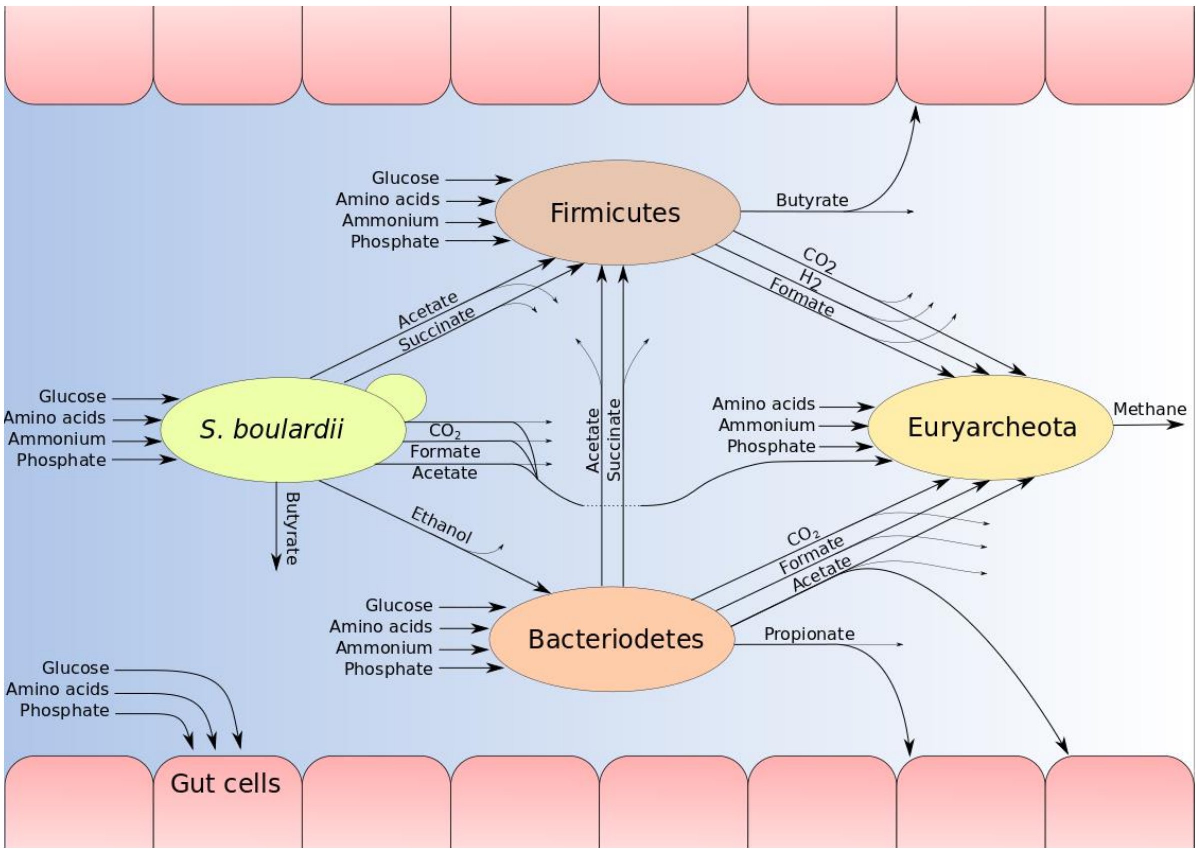

Figure 1: *Exchange of metabolites between the key microbiotic species in the colon and the engineered probiotic yeast S. boulardii*

However, we are also interested in modeling the specific metabolic interactions between colorectal cancer and our engineered yeast. Below is a schematic showing this interaction.

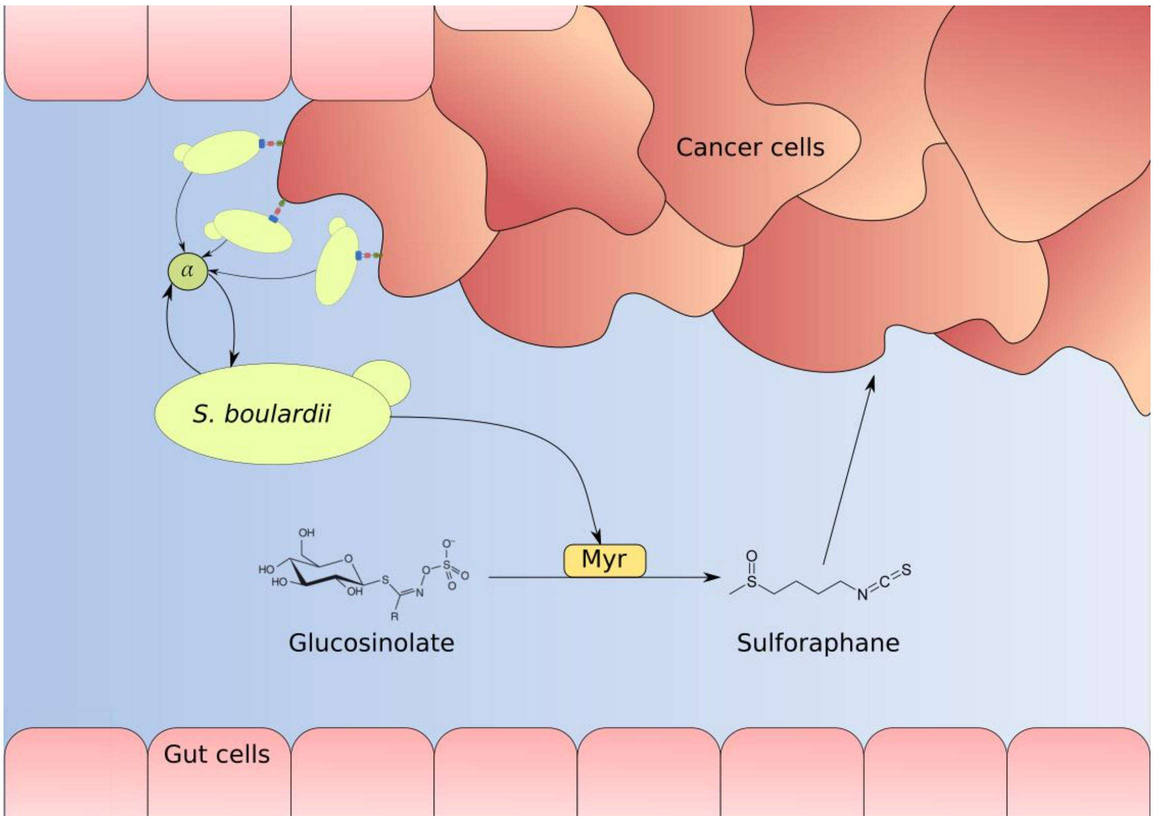

Figure 2: *Interaction between the engineered probiotic yeast S. boulardii and the cancer cells*

We decided to carry out some form of community dynamic flux balance analysis simulations early on in our modeling approach development. The advantage of dFBA over traditional FBA is that it allows us to simulate dynamic behavior. However it does require knowledge of certain kinetic parameters for the uptake of metabolites from literature. At first we considered compartmentalizing each model to generate one large network, but this would have been tedious and would have presented difficulties in determining a community objective function. To avoid this problem, we opted to not merge the models, and instead have them connect through mass balances of user-defined shared exchange metabolites. While there are existing tools which could be used for dFBA simulations, such as DyMMM, the COBRA toolbox, and DFBAlab, we decided to create our own dFBA tool, since the scheme is relatively straightforward to implement in MATLAB. Furthermore, the most recent and cutting edge tool, DFBAlab, does not seem to be easily accessible online. We requested access to the appropriate person, but we have yet to get a response from them.

COM-dFBA

Once we established the system to be modeled, we move onto the implementation of our dFBA scheme. We developed COM-dFBA, short for COMmunity dynamic Flux Balance Analysis, a simple framework for carrying out dFBA simulations for complex communities, such as the one shown above. COM-dFBA makes use of the RAVEN toolbox functionalities, and consists of six original functions and one script, which can be found in our GitHub repository under the folders COM-dFBA/Scripts. To understand the workings of COM-dFBA it is important to know that the chore of this simulation is to perform an iterative cycle consisting of three parts:

the kinetic block, the FBA block, and the dynamic block. This framework is summarized in the figure below. Based on user-defined initial metabolite concentrations, the kinetic block first determines the upper uptake rate of each metabolite *j* by each organism *i*, based on Michaelis-Menten kinetics. These rates are then used to update the allowed uptake bound of each exchange metabolite *j* by each organism *i*. Next, an FBA simulation is performed for each model in the FBA block. In the dynamic block, the fluxes for each exchange metabolite *j* is identified in the FBA solution of each organism *i*, and mass balances are generated for each exchange metabolite. An ODE solver then integrates these mass balances to determine the concentrations of each exchange metabolite *j* at a given time point t. These new metabolite concentrations are then used by the kinetic block to recalculate the allowed uptake bounds of metabolite *j* by organism *i*. This continues in a cyclic fashion until a steady state is reached, and thus dynamic simulation is obtained. Note that in the dynamic block, X_i corresponds to the biomass of organism *i*, and D is the dilution rate, defined as flow rate divided by volume.

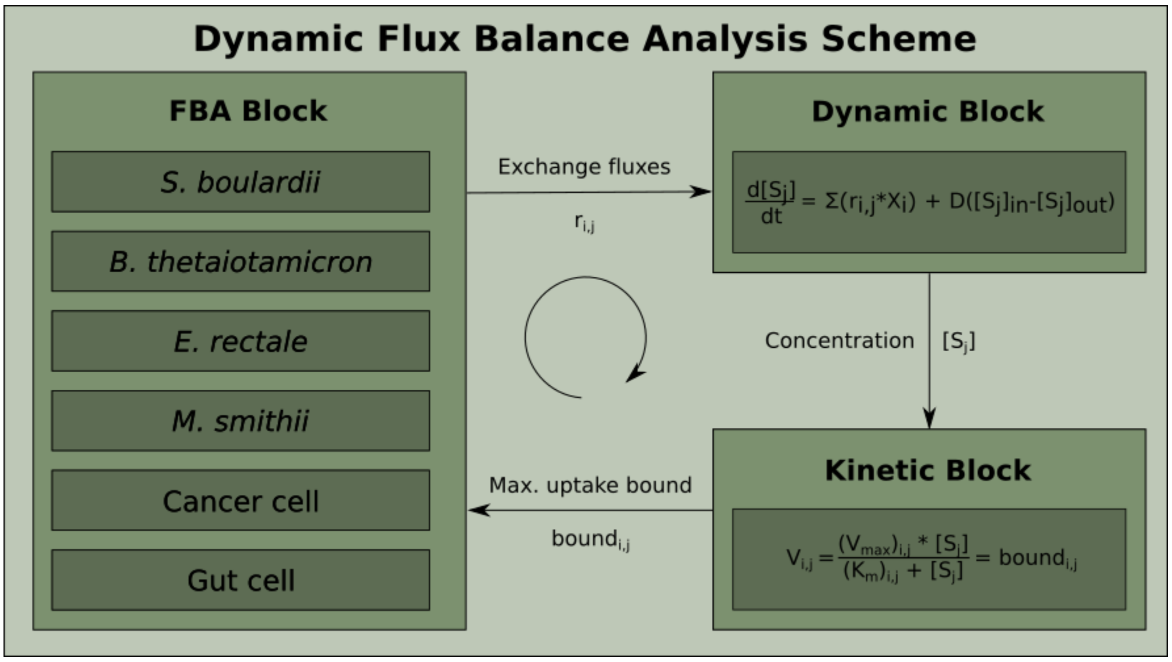
Dynamic Flux Balance Analysis Scheme
FBA Block
S. boulardii
B. thetaiotamicron
E. rectale
M. smithii
Cancer cell
Gut cell
Exchange fluxes
$r_{i,j}$
Dynamic Block
$\frac{d[S_j]}{dt} = \Sigma(r_{i,j}*X_i) + D([S_j]_{in}-[S_j]_{out})$
Concentration
$[S_j]$
Kinetic Block
$V_{i,j} = \frac{(V_{max})_{i,j} * [S_j]}{(K_m)_{i,j} + [S_j]} = bound_{i,j}$
Max. uptake bound
$bound_{i,j}$

Figure 3: *Detailed schematic of dFBA implementation*

The initialization and usage of this tool will now be described.

Initializing Models

Once the models have been curated we move on the initialization step. To view our code, please refer to section 2.0 of simulationScript.m. First, we create a cell structure containing the models to be used: modelSbo, modelBth, modelEre, modelMsi, modelCan, and modelCol representing *Saccharomyces boulardii*, *Bacteroides thetaiotaomicron*, *Eubacterium rectale*, *Methanobrevibacter smithii*, colorectal cancer cells, and colon cells, respectively. Next, we pass this cell structure to the *initModels* function, which outputs a structure containing the loaded models, called superModel, as well as other relevant information. Refer to initModels.m to see exactly what the function is doing. Next, we pass the superModel structure to the function *superModelFluxes*, which neatly summarizes the exchange fluxes of each model in a structure called fluxes. Note that these fluxes are calculated with no uptake limitations, aside from the ones imposed by each model's respective upper and lower bounds. Refer to superModelFluxes.m to see what the function is doing. Next, we create a cell structure with the names of the exchange metabolites for which mass balances will be generated. These include biomass for each model, substrates, and products. This structure, as well as the superModel structure, is passed to the *checkFluxes* function, which in turn outputs the structures metEquations, metIdx, bounds, blocked. This function is mostly meant to serve as a debugging tool to check that the metabolites have been correctly matched in each model, which can be double checked in the metEquations strucure. Refer to checkFluxes.m to see in detail what this function is doing.

Initializing Environment and Kinetic Parameters

Once the models have been initialized we move on the initializing the environment and kinetic parameters. To view our code, please refer to section 3.0 of simulationScript.m. First, we load data from the excel file called paramTables.xlsx, which can be found in the COM-dFBA/Scripts folder. This file contains four sheets, with information regarding Vmax values, Km values, initial and flow concentrations, and other miscellaneous parameters, respectively. The literature search and sources for these parameters are discussed in a further section. The excel data is loaded into a structure called params. Next, we pass the superModel and params structures to the function *updateFluxes*, which outputs the structures superModelConstrained, fluxesConstrained, boundsConstrained, and lbubConstrained. The function serves as a debugging tool and sanity check. It runs one iteration of the dFBA simulation to check that the bounds and fluxes are being adjusted properly, based on the chosen kinetic parameters and initial concentrations. The most interesting output to look at is perhaps fluxesConstrained, which displays similar information as the abovementioned fluxes structure, the difference being that the fluxes are now constrained by initial conditions and kinetic parameters. Refer to updateFluxes.m to see what the function is doing. Once everything is in working order, we finally move onto simulations.

Simulations

Once the environment and models have been initialized we can begin running simulations. Refer to section 4.0 of the file simulationScript.m to see the MATLAB code. To begin a simulation, we simply use the following snippet of code: *[t,xa] = ode15s(@(t,x)f(t,x,superModel,params),[t0 tf], initialConditions ,odeoptions)*. As can be seen, we rely on one of MATLAB's ODE solver for integration. More specifically, we use the variable step continuous implicit solver ode15s. This solver was determined to be the most efficient through trial and error of different solvers, including ode23 and ode113. Next, we see that the ode15s function calls on another function called *f*. Additionally, we provide the ODE solver with a time span, initial conditions, and some options. The function *f* takes in the superModel and params structures as input arguments. Please refer to the file f.m to inspect the code in detail. This function is the foundation of our dFBA framework, and contains three distinct sections: 1. KINETIC BLOCK, 2. FBA BLOCK, and 3. DYNAMIC BLOCK. The kinetic block uses Michaelis-Menten kinetics to calculate the maximum possible uptake bound for each metabolite and organism, which depends on how much of a given metabolite is present in the environment. The following expression is calculated in a nested for loop, for each organism *i* and for each metabolite *j*.

$$bound\,(\,i\,,\,j\,)\;=\;\frac{Vmax\,(\,i\,,j\,)\cdot x\,(\,j\,)}{Km\,(\,i\,,j\,)+x\,(\,j\,)}$$

Next, the function uses these above calculated values to update the appropriate uptake bounds in each corresponding GEM. Finally, we compare the MFalpha2 concentration in the environment with the MFalpha2 threshold for anti-cancer protein production, and update the protein production in the *S. boulardii* model accordingly. The FBA block then solves the models with updated bounds to obtain exchange fluxes for each organism. If the solution for the *S. boulardii* model becomes infeasible due to protein production constraints, we attempt to successively diminish the protein production load until the model is solvable or protein production is halted. Finally, in the dynamic block we take the FBA solutions and construct mass balances for each exchange metabolite of interest. This is done in a loop, and the mass balances are formulated as follows:

$$\frac{dX}{dt} = \Sigma\, production \;-\; \Sigma\, consumption \;-\; D\,(outflow \;-\; inflow\,)$$

In other words, the change in the concentration of metabolite X is equal to the sum of production of X, minus the sum of consumption of X, minus the difference between outflow and inflow of X multiplied by the dilution rate. Please note that the summation terms refer to production and consumption of metabolite X by *all* models, and they take into account the biomass concentration of each organism consuming or producing metabolite X. The only exception to this formulation is the mass balance for cancer biomass, which accounts for the effect of anti cancer proteins instead of having a dilution term. In the following equation the term ACE represents the Anti Cancer Efficiency (1/hr), MW represents the molecular weight of the anti cancer protein (g/mmol), and X(Myr) represents the anti cancer protein (in this case Myrosinase) concentration (mmol/L):

$$\frac{dX}{dt} = \Sigma\, production \;-\; ACE * MW * X(Myr)$$

Note that the resulting expressions for biomass have units of g/L*hr, whereas the expressions for metabolites have units of mmol/L*hr.

The beauty of COM-dFBA is that it takes care of identifying exchange fluxes for user-defined metabolites in the FBA solution of each model *i*, it identifies if the fluxes corresponds to production or consumption, and automatically formulates a mass balance equation for each exchange metabolite *j*. To see exactly how COM-dFBA does this, please refer to the last part of the *f* function.

Experimental Design

To investigate the effect of our engineered organism, we set up two distinct *in silico* experiments:

1. *S. boulardii* and Cancer
2. Gut Community Interaction
    - Gut Microbiome and Colon before *S. boulardii* Inflow
    - Gut Microbiome and Colon with *S. boulardii* Inflow
    - Gut Microbiome and Colon after interrupting *S. boulardii* Inflow

In the first *in silico* experiment we can see the interaction dynamics between *S. boulardii* and colorectal cancer. More specifically, we aim to investigate: if our engineered yeast can compete with the cancer for substrates, if our engineered yeast can produce enough α pheromone to trigger the production of our anti-cancer agent, and if the resulting anti-cancer agent concentration is sufficient to disrupt the growth of the cancer. This simulation was run for 500 hours.

In experiment number two, which is composed of three consecutive simulations, we aim to investigate the effect of adding *S. boulardii* to the gut microbiome. In the first sub-simulation, we determine the steady state composition of the gut microbiome without *S. boulardii*. In the second sub-simulation, we then introduce an inflow of *S. boulardii* to the steady state obtained in the first sub-simulation. Finally, in the third sub-simulation, we stop the inflow of *S. boulardii*. The rationale behind this is to see the extent to which *S. boulardii* disrupts the gut microbiome composition, to see if *S. boulardii* will become out competed by the gut microbiome once the dosage/inflow is halted, and to see if the gut microbiome can return to its pre-*S. boulardii* composition once the dosage/inflow is halted.

Results Simulation 1: *S. boulardii* & Cancer Cells

In this simulation we aim to investigate the interaction between a growing colorectal cancer and our engineered yeast. As can be seen on the top subplot, both the colon cancer and *S. boulardii* biomass are near zero at the starting time point. The cancer begins to grow in its characteristic exponential fashion, while the yeast accumulates and begins producing α pheromone. Just before the 50 hour mark, enough α pheromone accumulates to trigger the production of the anti cancer protein myrosinase. This causes the growth rate of the cancer to slow down, resulting in a global maximum of cancer biomass between 200 and 250 hours. After this phase there is enough myrosinase in the environment to “overtake” the growth rate of cancer, and we see cancer biomass begin to drop until it reaches zero just after 400 hours. *S. boulardii* approaches a steady state of around 12 g/L at the end of the simulation. The observed behavior of the α pheromone concentration, specifically after the peak around hour 50, is due to the fact that less α pheromone can be synthesized once myrosinase production begins. This seems reasonable considering there are amino acid and protein pool constraints. The production level of myrosinase relative to α pheromone concentration is validated by the myrosinase kinetic model. The system of ODEs defined by the exchange metabolite mass balances was solved for 1342 time steps, and the simulation took approximately 4.92 hours to complete. This simulation was run using MATLAB 2017b on the Hebbe computer cluster, which is part of the Chalmers Center for Computational Science and Engineering. The Hebbe cluster is built on Intel 2650v3 CPU's, and the system consists of a total 315 compute nodes (total of 6300 cores) with 26 TiB of RAM and 6 GPUs.

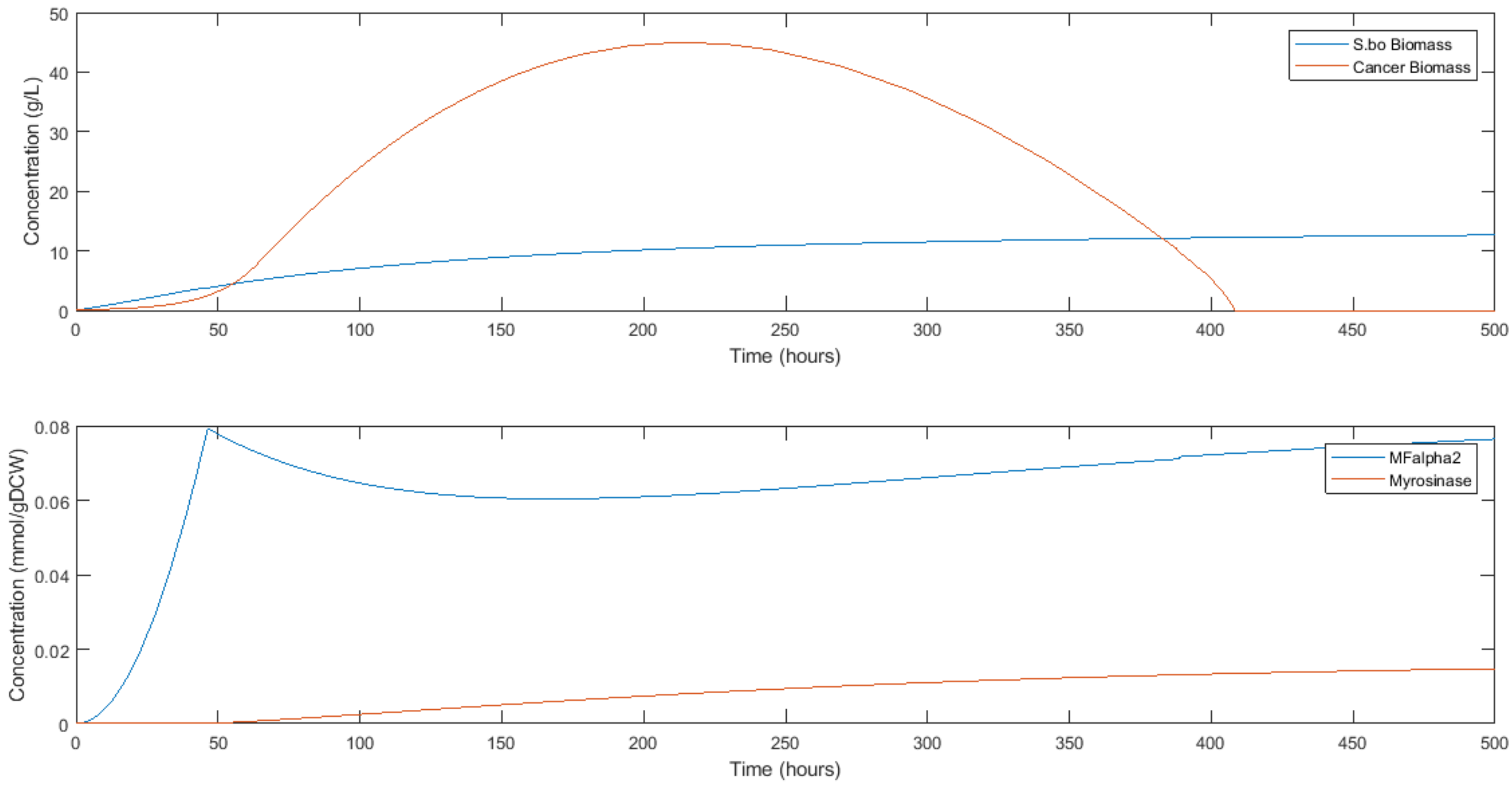

Figure 4: *Results Simulation 1: S. boulardii & Cancer Cells*

Results Simulation 2: Gut Community Interaction

As discussed previously, this simulation is actually composed of three successive simulations steps with the final concentrations of each step determining the initial concentrations of the successive one. The first simulation step serves to initialize the community model. The initial biomass and metabolite concentrations found in literature, which are summarized in the model design page, are used to find the a steady-state for the system. In the next step, *S. boulardii* is added to the inflow at a concentration of 1.00 g/L, which corresponds to a daily intake of in total 1.2 g of yeast. This is done to observe how our engineered yeast will establish itself, and how this will influence the rest of the gut microbiome. Finally, in the third step, the inflow of *S. boulardii* is interrupted, i.e. set to 0 g/L, to simulate the end of the treatment, in order to check whether *S. boulardii* will remain in the gut or if it will be washed-out.

The simulation result can be seen in the figure below, where the biomass concentrations are plotted over time. Vertical lines were plotted to delimit each of the three simulation steps.

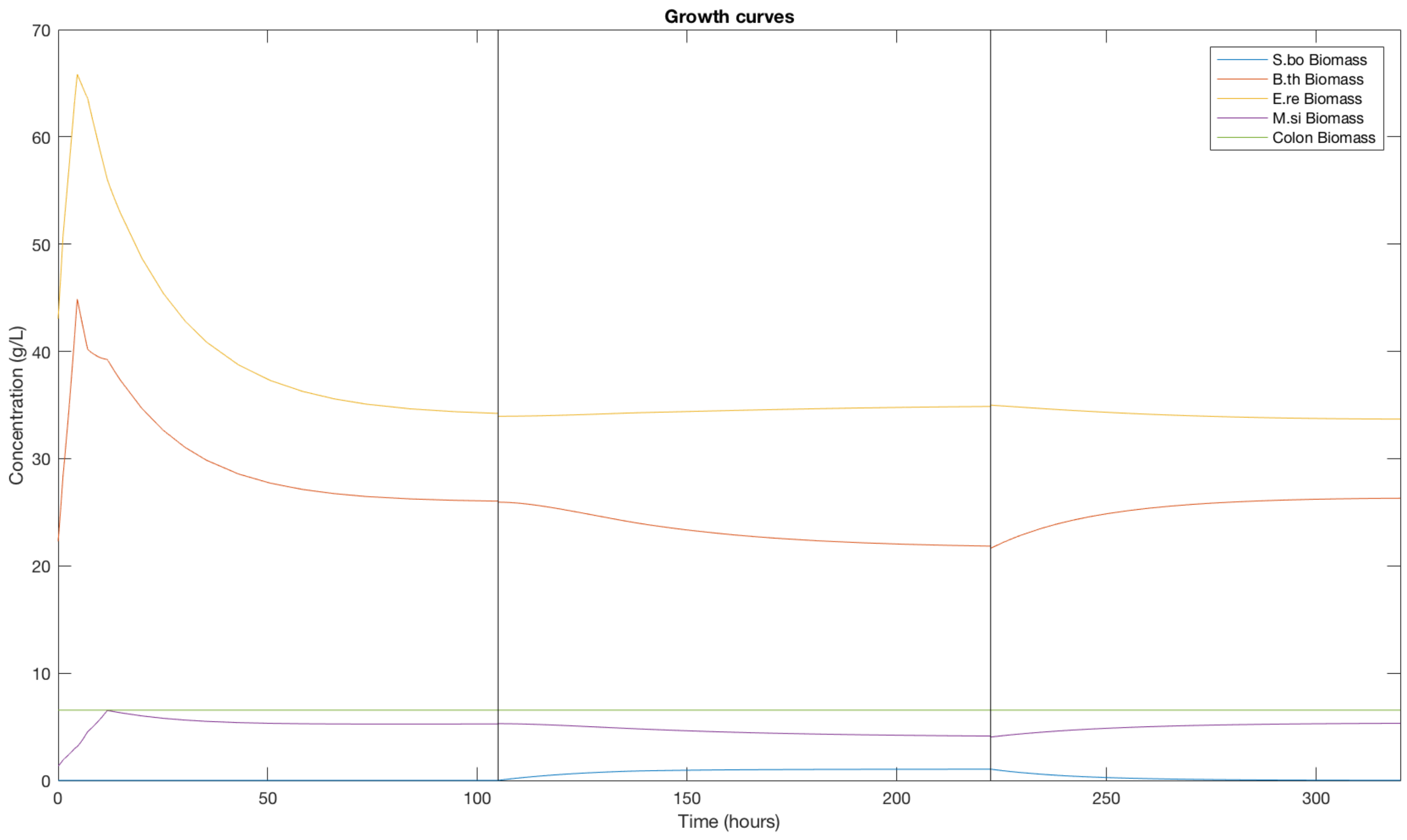

Figure 5: *Results Simulation 2: Community dFBA of Colon Cells and gut microbiome*

As can be seen in the figure above, the native gut microbiome composition reaches a steady state condition shortly after 100 hours. Soon after, an inflow of *S. boulardii* is introduced to simulate the start of probiotic consumption. With this new condition, a new steady state is reached at around the 225th hour mark. The introduction of our engineered yeast appears to have the most notable effect on the concentration of *B. thetaiotaomicron*, which decreases. Once the second steady state is reached, the inflow of *S. boulardii* is halted to simulate the end of probiotic consumption. As can be seen, the final steady state composition reached is almost identical to the first steady state composition, suggesting that our engineered yeast will not permanently disrupt the gut microbiome.

Table 1: *Final steady-state biomass concentrations*

	Sim. Step 1	**Sim. Step 2**	**Sim. Step 3**
S. boulardii	0.00	1.05	0.01
Bacteroidetes	25.94	21.64	26.31
Firmicute	33.95	34.98	33.69
Euryarcheota	5.23	4.04	5.32

First, it is noteworthy that the final *S. boulardii* concentration is 1.05 g/L while the inflow is 1.00 g/L. While this is not a large difference, it does mean that the *S. boulardii* is not only reaching the concentration in the inflow but is also able to grow.

As expected, the introduction of *S. boulardii* induced a change in the gut microbiome composition. We quantified this change by dividing the final concentration of each organism at the end of a simulation step by the steady state concentration of the corresponding organism obtained at the end of the first simulation step. These results are presented in Table 2 below.

Table 2: *Factor biomass concentrations change*

	Sim. Step 2	**Sim. Step 3**
Bacteroidetes	0.83	1.01
Firmicute	1.03	0.99
Euryarcheota	0.77	1.02

The change in Firmicute biomass concentration after the introduction and interruption of *S. boulardii* is negligible. However, the Bacteroidetes and Euryarcheota growth is significantly influenced by the introduction of *S. boulardii*. However, it is important to note that we know from literature that the exact composition of the gut microbiome is naturally variable (Karlsson et al., 2012). Especially variable is the proportion of Euryarcheota. Karlsson et al. (2012) measured it to be 2.0%±4.3, which means that our results are reasonable within these boundaries. Because of this, we can conclude that the introduction of *S. boulardii* as simulated does not disrupt the healthy gut microbiome in a significant way.

Additionally, we can conclude that our engineered yeast is unable to survive on its own without continuous dietary intake, since the *S. boulardii* concentration returns to values very close to 0 g/L around 50 hours after stopping inflow. Also, the other gut microbiome species return to concentrations very close to those before the treatment. This is an encouraging outcome, because patients would want the engineered yeast to leave their gastro-intestinal tract shortly after the treatment, without having their gut microbiome altered in a detrimental or permanent way.

Integrated Modeling

The purpose of the mathematical models created in this project is to make an impact on the course of the project and to be able to make inferences about the biological systems that we introduce into yeast. Although the models all have areas for improvement, the modeling made an impact on the overall project and the work in the wet lab. In this section we explain in what ways it did so.

Proof of Concept

First and foremost, the kinetic models and GEMs implemented in this project could be used as proof of concept and for illustrations of how our product will work in practice.

For the GEMs, the results from the simulations of the gut microbiota with and without *S. boulardii* indicate that the yeast has no dramatic effects on the composition of the gut microbiota. It demonstrates how *S. boulardii* can survive alongside gut microbial species while also not harming the patient. Moreover, simulations with *S. boulardii* , suggest that *S. boulardii* can produce myrosinase without depleting its amino acid resources. This means that the yeast can kill cancer cells while also growing.

S. boulardii **Content in Pill**

The simulation of together with the gut microbiota gave an indication of how much yeast to put inside of the final product, i.e. the pill. In the simulations, the inflow of *S. boulardii* was set to 1 g/L with a total liquid inflow of 0.05 L/h. This gives a daily intake of 1.2 g of yeast. Therefore, a patient should ingest 1.2 g of yeast every day during treatment. The patient could either take one pill per day or the content could be split into several pills to get an even flow of yeast throughout the day. Since part of the yeast cells will die both in the manufacturing process and on the journey from the patient's mouth to the gut, this should be considered a lower bound.

Future Work

The models created in this project gave us valuable insights into project design and helped us improve our visualised product. While this is true, the modeling could still be subject to improvement. Below, we bring up model improvements that did not fit the time scope of the project.

Regarding our dFBA framework implementation, it would be interesting to compare the accuracy and speed of COM-dFBA to other high quality dFBA tools, such as DFBAlab. To do this, we would simply run the same simulations using DFBAlab instead of COM-dFBA, and compare the results. Another aspect for future improvement would be to add a flux variability analysis (FVA) step to the COM-dFBA framework. This would help ensure that the FBA solutions obtained are unique, and that the models are sufficiently constrained. Additionally, improvements towards computational efficiency and speed would need to be made in order to feasibly run simulations with the all the GEMs implemented in this project, including the gut cell, the cancer cell, the gut microbiota and *S. boulardii*, in order to get an abstraction of the whole gut system. In addition to this, glucosinolate should be added to the gut inflow to get a better understanding of how the diet can affect the efficiency of myrosinase. Finally, it would be interesting to run a parameter sensitivity analysis, in order to determine which specific parameters have a significant influence on the interaction dynamics of interest. This would also provide us with potential engineering targets in the future.

To further analyze the required dosage of *S. boulardii*, we should model the yeast to cancer cell binding in the colon. A simulation with such a model would help us link the dosage required to get a stable growth of *S. boulardii* to the dosage required to effectively kill the cancer cells.

The optimal last step of this modeling project would be more closely integrate the results of the α pheromone and cell cycle models to the system of dFBA framework. In this way, we would obtain a large-scale model that can be used to evaluate the effects of our yeast on cancer cell survival and to further improve our product.

References

Agren, Rasmus, et al. “The RAVEN Toolbox and Its Use for Generating a Genome-Scale Metabolic Model for Penicillium Chrysogenum.” *PLoS Computational Biology*, vol. 9, no. 3, 21 Mar. 2013, doi:10.1371/journal.pcbi.1002980.

Amthor, J. “The McCree–De Wit–Penning De Vries–Thornley Respiration Paradigms: 30 Years Later.” *Annals of Botany*, vol. 86, no. 1, 2000, pp. 1–20., doi:10.1006/anbo.2000.1175.

Karlsson, Fredrik H., et al. “Symptomatic Atherosclerosis Is Associated with an Altered Gut Metagenome.” *Nature Communications*, vol. 3, no. 1, 2012, doi:10.1038/ncomms2266.

Shoaie, Saeed, et al. “Understanding the Interactions between Bacteria in the Human Gut through Metabolic Modeling.” *Scientific Reports*, vol. 3, no. 1, 2013, doi:10.1038/srep02532.