Minimizing the number of optimizations for efficient community dynamic flux balance analysis.

James D. Brunner†*, Nicholas Chia††

1 Center for Individualized Medicine, Mayo Clinic, Rochester, MN, USA

†Current Address: Mayo Clinic, 200 First St. SW, Rochester, MN, USA
* brunner.james@mayo.edu

Abstract

Dynamic flux balance analysis uses a quasi-steady state assumption to calculate an organism’s metabolic activity at each time-step of a dynamic simulation, using the well-know technique of flux balance analysis. For microbial communities, this calculation is especially costly and involves solving a linear constrained optimization problem for each member of the community at each time step. However, this is unnecessary and inefficient, as prior solutions can be used to inform future time steps. Here, we show that a basis for the space of internal fluxes can be chosen for each microbe in a community and this basis can be used to simulate forward by solving a relatively inexpensive system of linear equations at most time steps, instead of the full optimization problem. Using our method, we can use this solution as long as the resulting metabolic activity remains within the optimization problem’s constraints (i.e. the solution remains feasible). As the solution becomes infeasible, it first becomes a feasible but degenerate solution to the optimization problem, and we can solve a different but related optimization problem to choose an appropriate basis to continue forward simulation. We show using an eight species community that this is an efficient and robust method for computing dynamic flux balance analysis simulations, and so is capable of simulating communities of organisms. We demonstrate that the method gives an approximately 85% speed-up per organism over the standard and widely used method. Our method has been implemented in the Python language and source code is available at https://github.com/jdbrunner/surfin_fba and in the Python Package Index (PyPI) as surfinFBA.

Author summary.

The standard method in the field for dynamic flux balance analysis carries a prohibitively high computational cost because it requires solving a linear optimization problem at each time-step. We have developed a novel method for producing solutions to this dynamical system which greatly reduces the number of optimization problems that must be solved. We prove mathematically that we can solve the optimization problem once and simulate the system forward as an ordinary differential equation for some time interval, and solutions to this ODE provide solutions to the optimization problem. Eventually, the system reaches an easily checkable condition which implies that another optimization problem must be solved. We compare our method with the classical method to validate that it provides equivalent solutions in much lower computational time.

Introduction.

Microbial communities and human health.

The makeup of microbial communities is often complex, dynamic, and hard to predict. However, microbial community structure has a profound effect on human health and disease [1-7]. These two facts have lead to significant interest in mathematical models which can predict relative abundances among microbes in a community. Recently, advances in genetic sequencing have allowed the creation of genome-scale models
(GEMs) which reflect the internal network of cellular metabolism, and can therefore be used to predict metabolite use and production [8–10]. This technique can be extended to microbial community modeling by combining GEMs of different species. There has been significant interest in using GEMs to predict relative populations of stable microbial communities [11–17]. Community metabolic modeling can not only predict relative populations, but also holds the potential to predict and explain the community metabolite yield, which can have a profound effect on health [4]. Furthermore, model repositories such as the online bacterial bioinformatics resource PATRIC [18] or the BiGG model database [19] make it possible to build community models using information from individual species investigations.

Additionally, various dynamical models have been proposed to explain and predict microbial community population dynamics [20–24]. Among these are models which propose that interactions between species are mediated by the metabolites that each species produces and consumes [25,26], and there is significant evidence that these models perform better than models which depend on direct interaction between species [27,28].

GEMs can be used to predict microbial growth rates as well as metabolite consumption and production rates using a process called flux balance analysis (FBA). Because these predictions appear in the form of rates of change, they can be used to define a metabolite mediated dynamical model, simply by taking as a vector field the rates of change predicted by FBA. We can therefore combine the techniques of metabolite mediated dynamic modeling and community metabolic modeling to produce dynamic predictions of microbial community population size and metabolite yield. This strategy is called dynamic FBA [29,30], and has recently been used to model microbial communities [31,32].

Dynamic FBA, when implemented naively, requires a linear optimization problem to be repeatedly solved, and so carries a high computational cost even for small communities. Furthermore, in silico experiments may need to be repeated many times over various environmental conditions or using various parameter choices in order to make robust conclusions or to fit accurate model parameters. As a result, a naïve implementation of dynamic FBA carries a prohibitively high computational cost to make conclusions about larger microbial communities. The implementation of dynamic FBA in the popular COBRA toolbox software package [8] is done in this way, and so carries a high computational burden. Dynamic FBA can be improved by taking advantage of the linear structure of the optimization problem, so that optimizations are not necessary for every time step [33,34].

In order to implement dynamic FBA without optimizing at each time step, we use an optimal basic set for the FBA linear optimization problem to create a system of linear equations whose solutions at future time-steps coincide with the solutions to the FBA optimization problem. It is possible that the basic index set chosen may not provide future solutions if the optimization problem’s solution is degenerate. This is potentially a meaningful limitation of the method, because most re-optimizations required by the method involve degenerate solutions. To solve this problem, we provide a method to choose among possible basic index sets for a degenerate solution to the optimization problem. We show that there exists a choice of basic index set that allows forward simulation. We then describe an algorithm that chooses this useable basis in order to simulate forward with as few full optimizations as possible. Additionally, our implementation makes use of a slightly less costly optimization problem. Our method is available as a Python package published in the Python Package Index (PyPI) as surfinFBA, and source code is available at https://github.com/jdbrunner/surfin_fba.

In this manuscript, we detail how dynamic FBA can be simulated forward without re-optimization for some time interval, and give a method for doing so. We propose conditions on an optimal basic set for the FBA linear optimization problem which allows for forward simulation, and we prove that such a choice exists. We then detail how to choose this basis set, and finally give examples of simulations which demonstrate the power of our method and its speed relative to the classical implementation introduced in [30].

Background

Flux balance analysis.

With the advent of genetic sequence and the resulting genome scale reconstruction of metabolic pathways, methods have been developed to analyze and draw insight from such large scale models [9]. To enable computation of relevant model outcomes, constraint based reconstruction and analysis (COBRA) is used to model steady state fluxes $v_i$ through a microorganism’s internal metabolic reactions under physically relevant constraints [9]. One of the most basic COBRA methods, called flux balance analysis (FBA) optimizes...
some combination of reaction fluxes $\sum \gamma_i v_i$ which correspond to increased cellular biomass, subject to the constraint that the cell’s internal metabolism is at equilibrium:

$$\Gamma v = 0$$

where $\Gamma$ is the stoichiometric matrix, a matrix describing the stoichiometry of the metabolic model.

This optimization is of interest because it reflects the optimization carried out by nature through evolution. The vector $\gamma = (\gamma_1, \gamma_2, ..., \gamma_d)$ is an encoding of cellular objectives, reflecting the belief that the cell will be optimized to carry out these objectives. The constraint Eq. (1) means that any optimal set of fluxes found by FBA corresponds to a steady state of the classical model of chemical reaction networks. This reflects the assumption that the cell will approach an internal chemical equilibrium.

The optimization is done over a polytope of feasible solutions defined by the inequalities $v_{i,\text{min}} \leq v_i \leq v_{i,\text{max}}$, or possibly more complicated linear constraints. See Fig. 2 for a geometric representation of an example of the type of linear optimization problem that is carried out. By convention, forward and reverse reactions are not separated and so negative flux is allowed. Linear optimization problems like FBA often give rise to an infinite set of optimal flux vectors $v = (v_1, v_2, ..., v_d)$. Geometrically, this set will correspond to some face of the polytope of feasible solutions. To draw conclusions despite this limitation, many methods have been developed to either characterize the set of optimal solutions, as with flux variability analysis (FVA), or enforce more constraints on the network to reduce the size of this set, as with loopless FVA.

**Dynamic FBA.**

Flux balance analysis provides a rate of increase of biomass which can be interpreted as a growth rate for a cell. Furthermore, a subset of the reactions of a GEM represent metabolite exchange between the cell and its environment. By interpreting constraints on nutrient exchange reactions within the metabolic network as functions of the available external metabolites and fluxes of exchange reactions as metabolite exchange rates between the cell and its environment, the coupled system can be modeled. The simplest way to do this is to use an Euler method, as in [30]. The algorithm from [30] is summarized in Fig. 1.

This method is implemented in the COBRA toolbox [8], but notably requires a complete recalculation of the network fluxes at each timestep. Furthermore, for complex systems, time-steps must be small to avoid numerical error which causes the simulation to fail entirely. In our simulations of an 8-species system, a time-step of $10^{-4}$ was necessary to avoid negative metabolite biomasses, which cause the simulation to behave extremely inaccurate if allowed to continue forward. Thus, to simulate to 1 hour of growth, $10^4$ linear programs must be solved for each species.

However, resolving the system at each time step is not necessary, as the solution the optimization problem at some initial time can actually be used to compute future optimal solutions. Höfler et al. [34], used this observation to introduce a variable step-size method for dynamic FBA. In that method a basic index set is chosen by adding biological constraints to the optimization problem hierarchically until a unique optimal solution is found. The challenge of such an approach is in choosing the basis for the optimal solution, as the optimal basis is not guaranteed to be unique. We describe how to choose a basis from among the possibilities.
provided from an FBA solution which is most likely to remain optimal as simulation proceeds forward. We therefore prioritize reducing the number of times the linear program must be solved.

Additionally, a method described as “dynamic optimization approach” was introduced in Mahadevan et al., [29], however this method is computationally expensive. In particular, the method given in [29] involves optimizing over the entire time-course simulated, and so is formulated as a non-linear program which only needs to be solved once. While this method requires only one optimization, this optimization is itself prohibitively difficult due to the dimensionality of the problem growing with the fineness of time-discretization.

The dynamic FBA model for communities.

We can write a metabolite mediated model for the population dynamics of a community of organisms \( x = (x_1, ..., x_p) \) on a medium composed of nutrients \( y = (y_1, ..., y_m) \):

\[
\dot{x}_i = g_i(\psi_i(y))x_i \\
\dot{y}_j = -\sum_{i=1}^{p} \psi_{ij}(y)x_i
\]

where \( \psi_i \) is a vector of the fluxes of nutrient exchange reactions for organism \( x_i \) as determined by FBA. Using FBA to determine \( \psi_i \) is therefore a quasi-steady state assumption on the internal metabolism of the organisms \( x_i \) [36–38].

Recall that the basic assumption of flux balance analysis is that, given a matrix \( \Gamma_i \) that gives the stoichiometry of the network of reactions in a cell of organism \( x_i \) that growth \( g_i(y) \) is the maximum determined by solving the following linear program [9]:

\[
\begin{cases}
\max(v_i \cdot \gamma_i) \\
\Gamma_i v_i = 0 \\
c_1^i \leq v \leq c_2^i(y)
\end{cases}
\]

The key observation allowing dynamic FBA is that the optimal solution to this problem also determines \( \psi_i \) simply by taking \( \psi_{ij} \) to be the value of the flux \( v_{ij} \) of the appropriate metabolite exchange reaction. For clarity, we will relabel the elements of \( v_i \) so that \( \psi_{ik} = v_{ij} \) if \( v_{ij} \) is the \( k^{th} \) exchange flux, and \( \phi_{ik} = v_{ij} \) if \( v_{ij} \) is the \( k^{th} \) internal flux. The objective vector \( \gamma_i \) indicates which reactions within the cell contribute directly to cellular biomass, and so is non-zero only in elements corresponding to internal fluxes. We can therefore rewrite this vector to include only elements corresponding to internal fluxes, so that the objective of the optimization is to maximize \( \gamma_i \cdot \phi_i \).

The stoichiometry of metabolite exchange reactions is represented by standard basis vectors [9]. Therefore, we can partition \( \Gamma_i \) as

\[
\Gamma_i = \begin{bmatrix}
I & -\Gamma_i^* \\
0 & \Gamma_i^\dagger
\end{bmatrix}
\]

where \( I \) is the identity matrix of appropriate size, and \( \Gamma_i^* \) and \( \Gamma_i^\dagger \) contain the stoichiometry of the internal reactions [9,39,40].

We can see from Eq. (5) that \( \ker(\Gamma_i) \) is isomorphic to \( \ker(\Gamma_i^\dagger) \), and so we can maximize over this kernel. Then, the exchange reaction fluxes are determined by the internal fluxes according to the linear mapping \( \psi_i = \Gamma_i^* \phi_i \). The maximization of FBA becomes a maximization problem over the internal fluxes \( \phi_i \). We rewrite Eq. (4) using Eq. (5) and combine with Eqs. (2) and (3) to form the differential algebraic system

\[\text{Eq. (4)}\]

\[\text{Eq. (5)}\]

\[\text{Eq. (2)}\]

\[\text{Eq. (3)}\]

1In fact, we can project onto the kernel of the matrix \( \Gamma_i^\dagger \) and so reduce the dimensionality of the problem. However, in practice this projection is not numerically stable.
\[
\frac{dx_i}{dt} = x_i (\gamma_i \cdot \phi_i) \\
\frac{dy}{dt} = - \sum_i x_i \Gamma_i^* \phi_i
\]

where each \( \phi_i \) is determined by the optimization Eq. (8), all carried out separately. Note that this is a metabolite mediated model of community growth as defined in [27]. That is, the coupling of the growth of the separate microbes is due to the shared pool of metabolites \( y \). Each separate optimization which determines \( \phi_i \) at a single time-step depends on \( y \), and each \( \phi_i \) determines some change in \( y \).

We write, for full generality, upper and lower dynamic bounds on internal and exchange reactions, and assume that each function \( c_{ij} (y) \in C^\infty \). We let

\[
A_i = [(\Gamma_i^*)^T, -(\Gamma_i^*)^T, I, -I]^T
\]

so that we can rewrite the optimization problem Eq. (8) as

\[
\begin{cases}
\max (\phi_i \cdot \gamma_i) \\
A_i \phi_i \leq c_i (y, t) \\
\Gamma_i^* \phi_i = 0
\end{cases}
\]

for ease of notation.

We now hope to select a basic index set \( I_i \) for Eq. (10) for each organism \( x_i \) so that each \( \phi_i(t) \) is a solution to the resulting linear system of equations.

**Methods.**

**Linear optimization preliminaries.**

In this manuscript, we will rewrite the FBA optimization problem in the form

\[
\begin{cases}
\max (\phi \cdot \gamma) \\
A \phi \leq c \\
\Gamma^\dagger \phi = 0
\end{cases}
\]

where the matrices \( A \) and \( \Gamma^\dagger \) are derived from the stoichiometric matrix and flux constraints. Such a problem is often referred to as a *linear program* (LP). We now recall some well known results from the study of linear programming (see, for example [34][41]).

First, we note that Eq. (11) can be rewritten in the so-called *standard form* with the addition of slack variables \( s = (s_1, ..., s_n) \) which represent the distance each of the \( n \) constraints is from its bound as follows:

\[
\begin{cases}
\max (\tilde{\phi} \cdot \gamma) \\
[\tilde{A} \ I] \begin{bmatrix} \tilde{\phi} \\ s \end{bmatrix} = c \\
\tilde{\phi}_i \geq 0, s_i \geq 0
\end{cases}
\]

Standard form requires that we rewrite \( \phi_i = \phi_i^+ - \phi_i^- \) and then define \( \tilde{\phi} = (\phi_1^+, \phi_2^+, ..., \phi_d^+, \phi_1^-, \phi_2^-, ..., \phi_d^-) \) so that we require non-negativity of each variable, and the matrix \( \tilde{A} = [A B], B = -A \). We rewrite the problem in this form to make use of established results, and for ease of notation will write \( \phi \) instead of \( \tilde{\phi} \) when it is clear which form of the problem we are discussing.
We will make use of the well-known result that there exists an optimal basis or basic set for a bounded linear program \[42\]. To state this result, we first define the notation \(B_I\) to be the matrix with columns of \([A I]\) corresponding to some index set \(\{k_1, k_2, ..., k_n\} = I\), and if \(B_I\) is invertible we define the notation \(w_J(a)\) so that

\[
(w_J(a))_l = \begin{cases} (B_I^{-1} a)_l & l \in J \\ 0 & l \notin J \end{cases}
\]  

(13)

for any \(a \in \mathbb{R}^n\). We may now define an optimal basis and optimal basic set.

**Definition 1.** A basic optimal solution to a linear program is an optimal solution along with some index set \(\{k_1, k_2, ..., k_n\} = I\) such that \(w = w_I(c)\), where \(c\) is the vector of constraints as in Eq. (12). The variables \(w_i | i \in I\) are referred to as basic variables, and the index set \(I\) is referred to as the basic index set.

Finally, if there exists a bounded, optimal solution to Eq. (12), then there exists a basic optimal solution and corresponding basic index set.

For a given basic optimal solution vector \(w\), there may be more than one basic index set \(I\) such that \(w = w_I(b)\). Such a solution is called degenerate. Clearly a necessary condition for such non-uniqueness is that there exists some \(k \in I\) such that \(w_k = 0\). This is also a sufficient condition as long as there is some column of \([A I]\) which is not in the column space of \(B_I(k)\).

**Forward simulation without re-solving.**

Consider again Eq. (10), the linear program that must be solved at each time point of the dynamical system for each microbial population. Information from prior solutions can inform future time-steps as long as the region of feasible solutions has not qualitatively changed. Thus, we may only need to solve the optimization problem a few times over the course of a simulation, rather than at every time step. The key observation making this possible is that the simplex method of solving a linear program provides an optimal basis for the problem a few times over the course of a simulation, rather than at every time step. The key observation is that there exists some \(k\) such that \(w_k = 0\). This is also a sufficient condition as long as there is some column of \([A I]\) which is not in the column space of \(B_I(k)\).

One major difficulty in this technique is that there exists a unique \(w_i\) does not guarantee a unique basic set \(I_i\). If we have some \((w_{I_i})_j = 0\) for \(j \in I_i\), then there exists some alternate set \(\hat{I_i}\) such that \(w_{I_i} = w_{\hat{I_i}}\). Such a solution \(w_{I_i}\) is called degenerate. In a static implementation of a linear program, the choice of basis of a degenerate solution is not important, as one is interested in the optimal vector and optimal value. However, as we will demonstrate with Example 1, the choice of basis of a degenerate solution is important in a dynamic problem. In fact, if the system given in Eqs. (14) and (15) is evolved forward until \(w_{I_i}(c_i(t))\) becomes infeasible, the time at which the system becomes infeasible is the time at which we have some \((w_{I_i})_j = 0\) for \(j \in I_i\). Thus, we need to resolve Eq. (10) whenever \(w_{I_i}(c_i(t))\) becomes degenerate, which will be the final time-point at which the \(w_{I_i}(c_i(t))\) is feasible.

\[
\frac{dx_i}{dt} = x_i(\gamma_i \cdot \phi_i(c_i(t)))
\]

(14)

\[
\frac{dy}{dt} = -\sum_i x_i \Gamma_i^* \phi_i(c_i(t))
\]

(15)
Example 1. Consider the dynamic linear program

$$\begin{align*}
\max \left( (1,1) \cdot v \right) \\
\left\{ \begin{array}{l}
1 0 \\
0 1 \\
1 2 \\
v_i \geq 0
\end{array} \right.
\begin{bmatrix}
v \leq \begin{bmatrix}
10 \\
10 \\
30 - t
\end{bmatrix}
\end{bmatrix}
\end{align*}$$

In standard form at $t = 0$, this linear program becomes

$$\begin{align*}
\max \left( (1,1) \cdot v \right) \\
\left\{ \begin{array}{l}
1 0 1 0 0 \\
0 1 0 1 0 \\
1 2 0 0 1 \\
v_i, s_i \geq 0
\end{array} \right.
\begin{bmatrix}
v \\
s
\end{bmatrix}
\begin{bmatrix}
10 \\
10 \\
30
\end{bmatrix}
\end{align*}$$

which has the unique solution $w = (10, 10, 0, 0, 0)$. There are three choices of basic index sets: $\mathcal{I}_1 = \{1, 2, 3\}$, $\mathcal{I}_2 = \{1, 2, 4\}$, and $\mathcal{I}_3 = \{1, 2, 5\}$. The resulting bases are

$$B_{\mathcal{I}_1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix}, \quad B_{\mathcal{I}_2} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}, \quad B_{\mathcal{I}_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Computing Eq. (13) at $t > 0$ for each, we have that $B_{\mathcal{I}_1}$ yields $w_{\mathcal{I}_1}(c(t)) = (10 - t, 10, t, 0, 0)$, $B_{\mathcal{I}_2}$ yields $w_{\mathcal{I}_2}(c(t)) = (10, 10 - t/2, 0, 0, t/2)$, and $B_{\mathcal{I}_3}$ yields $w_{\mathcal{I}_3}(c(t)) = (10, 10, 0, 0, -t)$, shown in Fig. 2 for $t > 0$. Thus, only $w_{\mathcal{I}_2}(c(t))$ solves the dynamic problem because $w_{\mathcal{I}_1}(c(t))$ is not optimal and $w_{\mathcal{I}_3}(c(t))$ is not feasible for $t > 0$. Notice that the correct choice of basis fundamentally depends on the time-varying bound function $c(t) = (10, 10, 30 - t)$. To see this, consider other possible time-varying bounds $c(t)$ which have $c(0) = (10, 10, 30)$. For example, if $c(t) = (10 - t, 10 - t, 30)$, then only $B_{\mathcal{I}_3}$ would give the correct $w(c(t))$ for $t > 0$.

![Fig 2. Geometric representation of Example 1 for $t_3 > t_2 > t_1 > 0$, showing the three options for bases which are equivalent at $t = 0$. Note that the best choice depends on the function $c(t) = (10, 10, 30 - t)$ and cannot be chosen using the static problem alone. The feasible region of the optimization problem is shown in gray.](image)

A basis for the flux vector.

We now provide a method to choose a basis $\mathcal{I}_i$ for each organism $x_i$ in the case of a degenerate solution. Consider an optimal solution $w_i$ to the linear program Eq. (12). To simulate forward according to Eqs. (14) and (15), we need for each organism $x_i$ a basic index set $\mathcal{I}_i$ such that

$$\begin{align*}
\dot{w}_i = w_{\mathcal{I}_i} \left( \frac{d}{dt} c_i \right) \\
\begin{bmatrix}
\dot{A} & I
\end{bmatrix} \begin{bmatrix}
\dot{w} = \frac{d}{dt} c_i
\end{bmatrix}
\end{align*}$$

where $(w_{\mathcal{I}_i})_{ij} = 0 \Rightarrow \dot{w}_{ij} \geq 0$.

\[\text{(18)}\]
so that the solution remains feasible, and furthermore that $\dot{w}$ is optimal over the possible choice of basic index sets for $w$. This is obviously a necessary condition for forward simulation within some non-empty time interval, and can be made sufficient (although no longer necessary) by making the inequality $(w^\top x)_j = 0 \Rightarrow \dot{w}_j \geq 0$ strict. We use the relaxed condition for more practical applicability.

First, we show that such a solution exists.

**Lemma 1.** For a linear program with the form given in Eq. (12) with a basic optimal solution $w$, there exists a basic index set $I$ such that Eq. (18) holds and $\dot{w}$ is optimal over the possible choice of basic index sets for $w$.

**Proof.** For convenience, we now restate Eq. (12):

$$\max(\tilde{\phi} \cdot \tilde{\gamma})$$

$$\begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} \tilde{\phi} \\ s \end{bmatrix} = c$$

$$\tilde{\phi}_i \geq 0, s_i \geq 0$$

where we write $(\tilde{\phi}, s) = w$.

We note that there is a finite number of basic index sets for $w$, and so we need only show that there exists $I$ such that Eq. (18) holds. Then, the existence of an optimal such $I$ follows trivially.

If $w$ is not degenerate, then the unique choice of basic index set $I$ satisfies Eq. (18). To see this, simply note that if $w$ is non-degenerate, then for every $i \in I$, $w_i > 0$. Thus, Eq. (18) only includes non-negativity constraints on $\dot{w}_i$ if $i \not\in I$, and for any $i \not\in I$, $\dot{w}_i = 0$. Thus, the non-negativity constraints are enforced.

The equality constraints are enforced by the definition of $w_\pi(a)$ given in Eq. (13), which implies that $[A I]w_\pi(a) = a$ for any vector $a \in \mathbb{R}^n$.

In the case of a degenerate solution $w$, we use the following procedure to choose a set of basic variables. Let $J \subset \{1, ..., n\}$ be the indices of the $n_1$ slack variables such that $s_j = 0$ if $j \in J$ (recalling that each $s_i$ is a component of the vector $w$). Then, let $\hat{A}_J$ be the matrix with rows $m_j$ of $\hat{A}$ for $j \in J$. Next, let $J^*$ be the indices of the $n_2$ non-slack variables such that $\phi_j = 0$ and $I_{J^*}$ the corresponding rows of the identity matrix $I$. Notice that we now have that

$$M\tilde{\phi} = \begin{bmatrix} \hat{A}_J \\ -I_{J^*} \end{bmatrix} \begin{bmatrix} \tilde{\phi} \\ c_j \end{bmatrix}.$$  \hfill (19)

and that if $w_j = 0$ then either $j \in J^*$ or $w_j = s_k$ where $k \in J$ so that $m_k \cdot \tilde{\phi} = c_k$ (i.e. $s_k$ is a slack variable and $s_k = 0$). Notice that because Eq. (12) has a bounded solution, then we can assume without loss of generality that if $M \in \mathbb{R}^{q \times r}$, then $\text{rank}(M) = r$ (i.e. $M$ is full rank) because $w$ must satisfy at least $r$ linearly independent constraints. If this is not the case, then the problem can be projected onto a lower dimensional subspace.

Consider the linear program

$$\max(y \cdot \gamma)$$

$$\begin{bmatrix} M & I \end{bmatrix} \begin{bmatrix} y_{\tilde{\phi}} \\ y_s \end{bmatrix} = \begin{bmatrix} d^\top c_j \\ 0 \end{bmatrix},$$

$$y_i \geq 0$$

Assume that there is some basic optimal solution to Eq. (20) with a basic index set $\hat{I}$ such that exactly $r$ slack variables are non-basic, where again $r = |\tilde{\phi}|$ is the rank of the matrix $M$. This implies that there are $r$ linearly independent rows of $M$ (which we index by $J^\dagger$) which form an invertible matrix $\hat{M}$ such that

$$\hat{M}y_{\tilde{\phi}} = \begin{bmatrix} d^\top c_j \\ 0 \end{bmatrix}$$

and we can then determine $y_s$ by

$$y_s = \begin{bmatrix} d^\top c_j \\ 0 \end{bmatrix} - M\tilde{\phi}$$
and note that each \((y_s)_i \geq 0\). We now rewrite \(\dot{w} = (\dot{w}_\phi, \dot{w}_s)\) from Eq. \((18)\) and define \(\dot{w}_\phi = y_\phi\) and

\[
\dot{w}_s = \frac{d}{dt}c - M\dot{w}_\phi
\]  

and conclude that this satisfies the constraints of Eq. \((18)\). Next, we take \(\bar{\phi} = \frac{\dot{c}}{\dot{w}_\phi}\) to be the unique solution to

\[
\dot{M}\bar{\phi} = \begin{bmatrix} C_{J^1} & 0 \\
\end{bmatrix}
\]

and \(s = c - A\bar{\phi}\).

Finally, we take \(I = (\hat{I} \setminus J^*) \cup J^c\) and note that this basis set enforces exactly the same \(r\) linearly independent constraints as \(M[I]\).

We now prove that there is some basic optimal solution to Eq. \((20)\) with a basic index set \(\hat{I}\) such that exactly \(r\) slack variables are non-basic, where \(r\) is the rank of the matrix \(M\).

First we note that for any basic optimal solution, if there are \(r^* > r\) slack variables which are non-basic, then there are \(r^*\) rows of \(B_{\hat{I}}\) which are non-zero only in the columns of \(M\). Therefore, \(B_{\hat{I}}\) is not invertible. We can conclude that the number of non-basic slack variables is at most \(r\).

Next, suppose \(\dot{w}^*\) is a basic optimal solution with basis \(\hat{I}^*\) such that there are \(r^* < r\) slack variables which are non-basic.

We would like to assume that there are at least \(r\) slack variables \(s^*_k\) corresponding to \(r\) linearly independent constraints such that \(s^*_k = 0\). Recall that \(A\) was formed with repeated (negated) columns in order write the problem in standard form (the non-negativity bounds of Eq. \((12)\) are artificial). Therefore, we can find some vector \(x\) in the kernel of the matrix formed by the rows of \(A\) corresponding to zero slacks which also has \(x \cdot \gamma = 0\). We can therefore find a vector \(y\) in the kernel of

\[
\begin{bmatrix}
\hat{A}_J & I & 0 \\
-I_{J^*} & 0 & I
\end{bmatrix}
\]

which has \(y_k = 0\) if \(s_k = 0\) and \(y_j \neq 0\) if \(s_j \neq 0\) and \(s_j\) corresponds to a constraint that is not a linear combination of the constraints corresponding to the \(s_k = 0\). There is at least one such constraint as long as the 0 slack variables correspond to constraints with span less than dimension \(r\), and so we can take \(\dot{w} + \lambda y\) for some \(\lambda\) and so increase the number of non-zero slack variables. We can therefore assume without loss of generality that there are at least \(r\) slack variables \(s^*_k\) corresponding to \(r\) linearly independent constraints such that \(s^*_k = 0\), as was desired.

We can finally choose some linearly independent set of \(r\) constraints which correspond to \(0\) slack variables, and call the matrix whose rows are these constraint vectors \(M^*\). Now, because there are \(r^* < r\) non-sack basic variables, there is some non-sack, non-basic variable \(v_j\) such that the column \(m_j^*\) of \(M^*\) is linearly independent from the columns corresponding to the \(r^*\) non-sack basic variables. We can conclude that if

\[
B_{\hat{I}} \cdot \lambda = m_j
\]

then there is some \(\lambda_k \neq 0\) where \(k\) corresponds to the index of a slack variable with \(s_k = 0\). We can remove \(k\) from the basic index set and add \(j\) without changing \(\dot{w}^*\), and therefore preserving optimality and feasibility. We have then increased the number of non-basic slack variables, and we can repeat if necessary to form \(\hat{I}\) with exactly \(r\) non-basic slack variables.

\[\square\]

**Implementation.**

We implement dynamic FBA for python in the package surfinFBA. The main algorithm begins by computing an initial optimal internal flux vector each species according to Eq. \((10)\). A secondary optimization is then carried out, which can be specified by the user. If no secondary objective is provided, the program default is to minimize total internal flux. If the optimal solution is not degenerate, we may begin simulating forward
Algorithm: Efficient Dynamic FBA

Input: Final time $T$, initial microbial biomasses $x_i(0)$, initial nutrient concentrations $y_j(0)$, maximum inflow rates of nutrients $o_i$, stoichiometric matrices $\Gamma_i$

Output: Timecourse simulation of biomass and nutrient concentrations

for each microbial population $i$ do
  Set $w_i(0)$ to be solution to eq. (13) which lies on a vertex of the feasible polytope.;
  Solve eq. (21) to find initial basis $B_i$
end

while $t < T$ do
  Integrate eqs. (14) and (15) from $t$ to $t + \Delta t$ with $\phi_i = B_i^{-1}c_j(y(t),t)$;
  if $B_i^{-1}c_j(y(t + \Delta t), t + \Delta t)$ is not feasible then
    reset $x_i = x_i(t), y_i = y_i(t)$;
    Solve eq. (21) to find new basis $B_i$, with additional constraints representing bounds violated by $B_i^{-1}c_j(y(t),t)$.
  end
  if $B_i^{-1}c_j(y(t + \Delta t), t + \Delta t)$ is still not feasible then
    set $\Delta t = \frac{\Delta t}{2}$
  else
    set $\Delta t = 2\Delta t$
  end
end

Fig 3. Dynamic FBA algorithm following Lemma 1. Note that for numerical stability and speed, we may store the matrices $Q_i, R_i$ such that $Q_iR_i = B_i$ is the QR-factorization of $B_i$ rather than either storing $B_i^{-1}$ or solving completely during each time step of numerical integration.

until the solution becomes infeasible, and then repeat. Either the initial solution to Eq. (10) or the solution found when the initial solution becomes infeasible will be degenerate. Then, a new basis must be chosen as implied by Lemma 1. Note that in practice, the way that we form the matrix $A$ guarantees that rank$(A) = d$, where $\phi \in \mathbb{R}^d$ (i.e. $d$ is the number of internal fluxes).

To choose among possible bases of a degenerate solution, we form a second linear program analogous to Eq. (18) in the following way. We first find all constraints $a_i$, (i.e. rows of $A_i$ or $\Gamma_i^T$) such that $a_{ij} \cdot \phi_i = c_{ij}(t)$, calling this set $S_i$. Note that this set contains all the rows of $\Gamma_i^T$, for which we regard $c_{ij}(t) = 0$ for all $t > 0$. Note that if the solution given is a basic optimal solution, the rank of the matrix whose rows are $a_{ij}$ for $a_{ij} \in S_i$, is $d$, where again $d$ is the number of internal fluxes. This is true because we include constraints of the type $a < \phi_{ij} < b$ as rows of $A_i$.

Then, we solve the linear program

$$\begin{align*}
\text{max} & \quad \mathbf{a}_j \cdot \gamma_i \\
\text{subject to} & \quad \mathbf{a}_j \cdot \phi_i \leq \frac{\mathbf{d}_c_{ij} \cdot \gamma_i}{\mathbf{d}_a} \quad \mathbf{a}_j \in S_i
\end{align*}$$

(26)

We may then use any basis $B_i^T$ which solves Eq. (26) as long as it has exactly $d$ non-basic slack variables. Lemma 1 tells us that such a choice exists, although it may be necessary to manually pivot non-slack variables into the basis set given by the numerical solver. Note that we do not need the entire basis $B_i^T$, but instead only need the $d \times d$ submatrix formed by rows of $A_i$ or $\Gamma_i^T$ which correspond to non-basic slack variables in the solution to Eq. (26). These appear as rows $(\mathbf{a}_i, 0)$ in $B_i^T$, and so this sub-matrix uniquely determines $\phi_i$. We call this smaller matrix $B_i$, and label the set of row indices as $J$.

The chosen basis $J$ and corresponding constraints is used to simulate forward until the solution becomes infeasible. At that time, we have an optimal solution to Eq. (10) simply by continuity. We therefore do not need to resolve Eq. (10) but instead re-form and solve Eq. (26).

Results.

Error estimation.

Figure 3 implies that a simulation of a microbial community can be divided into time intervals on which the algorithm is exact. That is, there exists some sequence $t_0 = 0 < t_1 < \cdots < t_{n-1} < t_n = T$ such that if we know the optimal flux vectors $w_i(t_l)$ at time $t_l$, then Lemma 1 implies the existence of a set of invertible

---

3In testing the algorithm, this was necessary when using IBM ILOG CPLEX Optimization Studio to solve, but not when using The Gurobi Optimizer.
matrices $B_l^t$ such that solutions to Eqs. (14) and (15) are solutions to Eqs. (6), (7) and (10) for $t \in [t_l, t_{l+1}]$. Therefore, if we are able to identify the $t_l$ exactly, then Fig. 3 provides exact solutions to the dynamic FBA problem Eqs. (6), (7) and (10). Of course, numerical limitations imply that we will not re-optimize precisely at each $t_l$, and so we must investigate the impact of this error. However, once re-optimization is done, the method is again exact. The result is that we have no local truncation error for any time step taken between re-optimization after $t_l$ and the interval endpoint $t_{l+1}$, except for error due to numerical integration.

Assume that $t_{l-1}$ is known exactly, and $N$ is such that $t^1 = t_{l-1} + (N-1)\Delta t \leq t_l < t_{l-1} + N\Delta t = t^2$, so that there is some possible error in the interval $[t^1, t^2]$. We can estimate the accumulated error in this time interval using a power series expansion. Let $\mathbf{x}(t), \mathbf{y}(t)$ be solutions to Eqs. (6), (7) and (10) and $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ be solutions given by Fig. 3 for $t \in [t^1, t^2]$. Furthermore, let $B_l^{-1}$ be the invertible matrices derived by solving Eq. (10) at $t_{l-1}$ and $B_l^t$ those derived by solving at $t_l$. Then, $\mathbf{x}(t^1) = \tilde{\mathbf{x}}(t^1)$ and $\mathbf{y}(t^1) = \tilde{\mathbf{y}}(t^1)$. For each $x_i$, we expand, assuming some regularity of the functions $c(\mathbf{y}),$

$$x_i(t^2) - \tilde{x}_i(t^2) = (\Delta t)x_i(t^1)(\gamma_i \cdot ((B_l^{t^1})^{-1} - (B_l^{t^1})^{-1}) \cdot \tilde{c}_i(y(t^1))) + o(\Delta t) \quad (27)$$

and see that this method gives first order local error in time steps that require a re-optimization.

The local error, while first order, only appears at time steps in which a re-optimization occurred, and so global error will scale with the number of necessary re-optimizations. This is in contrast with Fig. 1 which gives first order local error at every time-step.

**Number of optimizations.**

One way in which we can assess the usefulness of Fig. 3 in relation to the classical algorithm (Fig. 1) is to estimate the number of times a linear program must be solved in a simulation with a given error tolerance. In addition to the obvious relationship with the speed of the simulation, the number of re-optimizations needed also effects the accuracy of Fig. 3 as we showed in the previous section.

Clearly, Fig. 1 requires $N$ linear programs to be solved, where $N$ is the number of time steps taken in the simulation. On the other hand, we see in Fig. 4 that Fig. 3 requires a linear program be solved far fewer times. See supplemental S1_File.pdf for details on community simulations using both Fig. 3 and Fig. 1.

**Computational time.**

We can also compare implementations of Fig. 3 and Fig. 1. Figure 1 is analogous to Euler’s method for computing solutions to ordinary differential equations, and is likewise dependent on time-step size for both speed and accuracy. Simulating communities of more than a single organism required a time step size of $10^{-4}$ in order to avoid numerical error which caused the solution to terminate. For too large a step size, numerical error caused metabolite concentrations to become less than zero and the simulation to terminate. To compare Fig. 3 and Fig. 1, we implemented Fig. 1 for communities using the FBA methods of the COBRApy Python package [8]. The exact implementations of Fig. 3 and Fig. 1 that we compared are available for download as a Python package at https://github.com/jdbrunner/surfin_fba.

Figure 5 (a) shows the wall time to simulate dynamic flux balance analysis with no microbial dilution, metabolite dilution, or metabolite replenishing for various small communities. To test the algorithms, we assumed linear nutrient uptake with randomly drawn nutrient uptake rates. That is, in Eq. (10) we took for each organism $i$

$$c_{ij}^2(y) = \kappa_{ij} y_j \quad (28)$$

with $\kappa_{ij} \in (0, 1)$ chosen uniformly at random. For each community and choice of $\{\kappa_{ij}\}$, we simulated using both algorithms and recorded wall time to simulate. Figure 5 (b) shows the time length of the simulation per organism. We can see that, for small communities at least, the time to simulate using Fig. 3 scales linearly with community size.
Examples & applications.

An 8 species community model.

Our method can be used to simulate a community of eight microorganisms using “off the shelf” models downloaded from the online bacterial bioinformatics resource PATRIC. We simulate the eight species used in Friedman et. al using Fig. 3 in a low flow chemostat with random metabolite uptake rates and random microbial death rates. The community dynamics are shown in Fig. 6.

We can also inspect the dynamics of the metabolite biomass in the medium. Interestingly, many metabolites show non-monotonic dynamics, and thus interesting transient behavior before equilibrium is reached. In Fig. 7 we show one such metabolite, uracil. Furthermore, we can inspect the effect microorganism $i$ has on the metabolite over time by inspecting $(-\Gamma_i^*\phi_{ij})$, where $j$ corresponds to this metabolite. For example, we see that $P. citronellolis$ initially consumes uracil $((-\Gamma_i^*w_{ij}) < 0)$, but later switches to producing the metabolite.
Fig 5. (a) Comparison of time to simulate community models to $t = 1$, with no inflow or outflow of metabolites or microbes. (b) Comparison of time to simulation per community size of community models with no inflow or outflow of metabolites or microbes.

Fig 6. 8 Species low-flow chemostat community model with random microbial death rates and metabolite uptake rates. This is model $S_{J5221630L160611764JTE-3a}$ R02da F1OS_65, which simulates the a community of the 8 species used in Friedman et. al [21] using Fig. [3]. Metabolic models for each species were downloaded from the online bacterial bioinformatics resource PATRIC [18].

Prediction dependence on nutrient uptake.

It is worth noting that dynamic FBA depends fundamentally on the nutrient uptake bounds $c(y)$ in Eq. (10). For our simulations, we take these to be linear,

$$c^2_{ij}(y) = \kappa_{ij} y_j$$ (29)
In the eight species community model shown in Fig. 6, uracil biomass changes in complicated ways. We can see which microorganisms are having an effect on this metabolite by inspecting each organism’s contribution to $\dot{y}$. When this is positive, this organism is producing the metabolite, and when it is negative, the organism is consuming the metabolite. Interestingly, $P. \text{citronellolis}$ transitions from initially consuming to later producing uracil.

meaning that the maximum uptake rate of nutrient $y_j$ by organism $x_i$ is proportional to the concentration of $y_j$. The choice of $\kappa_{ij}$ may have a profound effect on the outcome of a community simulation, as it represents how well an organism can sequester a resource when this will optimize the organism’s growth. In order study this effect in a small community, we sampled a three-species community model with $\kappa_{ij} \in (0,1)$ chosen uniformly at random. We used models for $Pseudomonas \text{putida F1}$, $Pseudomonas \text{chlororaphis strain PCL1606}$ and $Pseudomonas \text{veronii strain R02}$ downloaded from the online bacterial bioinformatics resource PATRIC[18]. We simulated with no dilution of metabolites or microbes, and no replenishment of nutrients. In every simulation, some critical metabolite was eventually depleted and the organisms stopped growing. We recorded the simulated final biomass of each organism from each simulation, and the results are shown in Fig. 8. While a wide variety of outcomes was observed, these did cluster about a result in which $P. \text{veronii}$ outgrew the other two organisms. This suggests that $P. \text{veronii}$ has some competitive advantage in the COBRApy default simulated medium.

Community growth effects.

As we saw in previous section, community growth outcomes depend on the choice of nutrient uptake rates $\kappa_{ij}$. Using Fig. 3 we can perform monte-carlo sampling in order to understand the possible effects on some microorganism of growing in some community. To do this, we randomly sample the set of uptake rates $\kappa_{ij}$ and run simulations of various communities for the chosen uptake rates. Then, the correlation between communities of final simulated biomass of some organism can be interpreted as the effect of the community on the growth of that organism. A correlation less than 1 between growth of an organism in different communities indicates that the community is having some effect. To see the direction of this effect, we can fit a simple linear regression model (best fit line) to the final simulated biomasses. Then, the slope of this line tells us if the organism benefits or is harmed by being in one community over another.

We again simulated $P. \text{putida}$, $P. \text{chlororaphis}$ and $P. \text{veronii}$ using models from PATRIC. Each organism grew to a larger final simulated biomass when alone compared to when in a trio with the other two. However, this difference was much less for $P. \text{veronii}$, which in general out-competed the other two species. These results are shown in Fig. 9.
Fig 8. (a) Histogram of the final simulated biomass of each of \textit{P. putida}, \textit{P. chlororaphis} and \textit{P. veronii} from 537 simulations, each with different metabolite uptake rates $\kappa_{ij}$. (b,c,d) Pair-wise comparison of the final simulated biomass densities using a kernel density estimation. In red is the result of uniform uptake rates $\kappa_{ij} = 0.5$ for all $i,j$.

Fig 9. Randomly sampled uptake parameters can give an indication of metabolite mediated community effects.
Supporting information.

**S1 File. Data from test simulations.** Test simulations were performed using models downloaded from PATRIC of species *E. aerogenes*, *P. chlororaphis*, *P. flourescens*, *P. citronellolis*, *P. veronii*, *P. putida*, and *S. marcescens*.

**S2 Software.** https://github.com/jdbrunner/surfin_fba. Available code for the algorithm described in the Python language. This package is designed for use with the popular COBRAPy package for metabolic models.

Acknowledgments

This work was supported by funding from the DeWitt and Curtiss Family Foundation, National Cancer Institute grant R01 CA179243, and the Center for Individualized Medicine, Mayo Clinic.

References

1. Braundmeier AG, Lenz KM, Inman KS, Chia N, Jeraldo P, Walther-António MRS, et al. Individualized medicine and the microbiome in reproductive tract. Frontiers in Physiology. 2015;6:97. doi:10.3389/fphys.2015.00097.

2. Calcinotto A, Brevi A, Chesi M, Ferrarese R, Perez LG, Grioni M, et al. Microbiota-driven interleukin-17-producing cells and eosinophils synergize to accelerate multiple myeloma progression. Nature communications. 2018;9(1):4832.

3. Flemer B, Lynch DB, Brown JM, Jeffery IB, Ryan FJ, Claesson MJ, et al. Tumour-associated and non-tumour-associated microbiota in colorectal cancer. Gut. 2017;66(4):633–643.

4. Hale VL, Jeraldo P, Chen J, Mundy M, Yao J, Priya S, et al. Distinct microbes, metabolites, and ecologies define the microbiome in deficient and proficient mismatch repair colorectal cancers. Genome Medicine. 2018;10(1):78. doi:10.1186/s13073-018-0586-6.

5. Ng KM, Ferreyra JA, Higginbottom SK, Lynch JB, Kashyap PC, Gopinath S, et al. Microbiota-liberated host sugars facilitate post-antibiotic expansion of enteric pathogens. Nature. 2013;502:96 EP –.

6. Round JL, Mazmanian SK. The gut microbiota shapes intestinal immune responses during health and disease. Nature Reviews Immunology. 2009:9:313 EP –.

7. Walsh DM, Mert I, Chen J, Hou X, Weroha SJ, Chia N, et al. The Role of Microbiota in Human Reproductive Tract Cancers. In: AMERICAN JOURNAL OF PHYSICAL ANTHROPOLOGY. vol. 168. WILEY 111 RIVER ST, HOBOKEN 07030-5774, NJ USA; 2019. p. 260–261.

8. Heirendt L, Arreckx S, Pfau T, Mendoza S, Richelle A, Heinken A, et al. Creation and analysis of biochemical constraint-based models: the COBRA toolbox v3. 0. arXiv. arXiv preprint arXiv:171004038. 2017.;

9. Lewis NE, Nagarajan H, Palsson BO. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. Nature Reviews Microbiology. 2012:10:291 EP –.

10. Lloyd CJ, Ebrahim A, Yang L, King ZA, Catoiu E, O’Brien EJ, et al. COBRAme: A computational framework for genome-scale models of metabolism and gene expression. PLOS Computational Biology. 2018;14(7):1–14. doi:10.1371/journal.pcbi.1006302.

11. Chan SHJ, Simons MN, Maranas CD. SteadyCom: Predicting microbial abundances while ensuring community stability. PLOS Computational Biology. 2017;13(5):1–25. doi:10.1371/journal.pcbi.1005539.
12. Diener C, Resendis-Antonio O. Micom: metagenome-scale modeling to infer metabolic interactions in the microbiota. bioRxiv. 2018; doi:10.1101/361907.

13. Gottstein W, Olivier BG, Bruggeman FJ, Teusink B. Constraint-based stoichiometric modelling from single organisms to microbial communities. Journal of the Royal Society Interface. 2016;13(124):20160627.

14. Mendes-Soares H, Mundy M, Soares LM, Chia N. MMinte: an application for predicting metabolic interactions among the microbial species in a community. BMC Bioinformatics. 2016;17(1):343. doi:10.1186/s12859-016-1230-3.

15. Zomorrodi AR, Islam MM, Maranas CD. d-OptCom: Dynamic Multi-level and Multi-objective Metabolic Modeling of Microbial Communities. ACS Synthetic Biology. 2014;3(4):247–257. doi:10.1021/sb4001307.

16. Borer B, Ataman M, Hatzimanikatis V, Or D. Modeling metabolic networks of individual bacterial agents in heterogeneous and dynamic soil habitats (IndiMeSH). PLoS computational biology. 2019;15(6).

17. Koch S, Kohrs F, Lahmann P, Bissinger T, Wendschuh S, Benndorf D, et al. RedCom: A strategy for reduced metabolic modeling of complex microbial communities and its application for analyzing experimental datasets from anaerobic digestion. PLoS computational biology. 2019;15(2):e1006759.

18. Wattam AR, Davis JJ, Assaf R, Boisvert S, Brettin T, Bun C, et al. Improvements to PATRIC, the all-bacterial bioinformatics database and analysis resource center. Nucleic acids research. 2017;45(D1):D535–D542.

19. King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, et al. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic acids research. 2016;44(D1):D515–D522.

20. Fisher CK, Mehta P. Identifying keystone species in the human gut microbiome from metagenomic timeseries using sparse linear regression. PLoS one. 2014;9(7):e102451.

21. Friedman J, Higgins LM, Gore J. Community structure follows simple assembly rules in microbial microcosms. Nature Ecology & Evolution. 2017;1:0109 EP –.

22. Goyal A, Maslov S. Diversity, Stability, and Reproducibility in Stochastically Assembled Microbial Ecosystems. Phys Rev Lett. 2018;120:158102. doi:10.1103/PhysRevLett.120.158102.

23. Stein RR, Bucci V, Toussaint NC, Buffie CG, Rätsch G, Pamer EG, et al. Ecological modeling from time-series inference: insight into dynamics and stability of intestinal microbiota. PLoS computational biology. 2013;9(12):e1003388.

24. Sung J, Kim S, Cabathat JJT, Jang S, Jin YS, Jung GY, et al. Global metabolic interaction network of the human gut microbiota for context-specific stability of intestinal microbiota. Nature communications. 2017;8:15393; 15393–15393. doi:10.1038/ncomms15393.

25. Niehaus L, Boland I, Liu M, Chen K, Fu D, Henckel C, et al. Microbial coexistence through chemical-mediated interactions. bioRxiv. 2018; doi:10.1101/358481.

26. Posfai A, Taillefumier T, Wingreen NS. Metabolic Trade-Offs Promote Diversity in a Model Ecosystem. Phys Rev Lett. 2017;118:028103. doi:10.1103/PhysRevLett.118.028103.

27. Brunner JD, Chia N. Metabolite-mediated modelling of microbial community dynamics captures emergent behaviour more effectively than species–species modelling. Journal of the Royal Society Interface. 2019;16(159):20190423.

28. Momeni B, Xie L, Shou W. Lotka-Volterra pairwise modeling fails to capture diverse pairwise microbial interactions. Elife. 2017;6:e25051.
29. Mahadevan R, Edwards JS, Doyle FJ. Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli. Biophysical Journal. 2002;83(3):1331–1340. doi:https://doi.org/10.1016/S0006-3495(02)73903-9.

30. Varma A, Palsson BO. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110. Applied and Environmental Microbiology. 1994;60(10):3724–3731.

31. Henson MA, Hanly TJ. Dynamic flux balance analysis for synthetic microbial communities. IET systems biology. 2014;8(5):214–229.

32. Song HS, Cannon WR, Beliaev AS, Konopka A. Mathematical modeling of microbial community dynamics: a methodological review. Processes. 2014;2(4):711–752.

33. Gomez JA, Höffner K, Barton PI. DFBAlab: a fast and reliable MATLAB code for dynamic flux balance analysis. BMC bioinformatics. 2014;15(1):409.

34. Höffner K, Harwood SM, Barton PI. A reliable simulator for dynamic flux balance analysis. Biotechnology and Bioengineering. 2012;110(3):792–802. doi:10.1002/bit.24748.

35. Feinberg M, Horn F. Dynamics of open chemical systems and the algebraic structure of the underlying reaction network. Chemical Engineering Science. 1973;29:775–787.

36. Baroukh C, Muñoz-Tamayo R, Steyer JP, Bernard O. DRUM: a new framework for metabolic modeling under non-balanced growth. Application to the carbon metabolism of unicellular microalgae. PloS one. 2014;9(8).

37. Öyäs O, Stelling J. Genome-scale metabolic networks in time and space. Current Opinion in Systems Biology. 2018;8:51–58.

38. Zazueta CL, Bernard O, Gouzé JL. Reduction of Metabolic Networks keeping Core Dynamics. Discrete Applied Mathematics. 2018;157(10):2483–2493.

39. Kondo A, Ishii J, Hara KY, Hasunuma T, Matsuda F. Development of microbial cell factories for bio-refinery through synthetic bioengineering. Journal of biotechnology. 2013;163(2):204–216.

40. Bordbar A, Monk JM, King ZA, Palsson BO. Constraint-based models predict metabolic and associated cellular functions. Nature Reviews Genetics. 2014;15(2):107–120.

41. Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. vol. 6. Athena Scientific Belmont, MA; 1997.

42. Tardella F. The fundamental theorem of linear programming: extensions and applications. Optimization. 2011;60(1-2):283–301.