# Generic model

Here we present the basic components of our kinetic model designed to simulate the flexible life strategies of human gut bacteria. The model is based on experiments conducted in monocultures and cocultures of a synthetic community of three species.

## Model States:

### 1. Subpopulations

Bacteria dynamically change their environments by consuming nutrients and excreting metabolic by-products. Within a community, species are often subjected to changing environments. For example, gut bacteria can be challenged by their host's feeding habits and circadian rhythms, as well as by the metabolic byproducts of neighboring species. To cope with these variations, cells within a population can alter their physiology, leading to phenotypic heterogeneity and the emergence of subpopulations within a single species, which may or may not coexist in space and time.

Subpopulations materialize due to changes in cellular physiology in responses to environmental variations (either triggered by intrinsic cellular programs or, often, as an unavoidable consequence of being propelled towards intolerable conditions). In our model, we represent these subpopulations as distinct model structures for a given species, each having its own set of kinetic parameters. These structures are linked through transition rates and transition functions, which encode the likelihood that cells transition from one subpopulation to another in response to environmental cues.

#### Transition rates ()

Cellular transitions can result from a single process, multiple independent processes, or multiple dependent processes. These transitions may occur rapidly, affecting cells concurrently or they may take place gradually, leading to the formation of co-existing subpopulations. The velocity at which a population transitions from one state to another is modulated by the transition rate ().

#### Transition functions ()

In our model, transition functions encode the environmental cues that trigger transitions between subpopulations. These functions operate like sensors, responding to changes in factors such as pH, metabolite concentrations, or other environmental variables. Transitions can occur independently or involve activation or inhibition mechanisms.

##### *Independent transitions*

These transitions occur at a constant rate, unaffected by any environmental conditions. In this case, equals one, indicating a fixed transition rate. This transition is commonly assumed in scenarios such as when there is a constant death rate.

The following equation exemplifies an independent transition rate. Here, subpopulation contributes towards the derivative of subpopulation , given the counts and for each subpopulation, respectively:

The first term on the right-hand side accounts for the contribution of subpopulation to the growth of subpopulation , where is the transition rate from subpopulation to subpopulation . The ellipsis represents any additional terms that may affect the time derivative of subpopulation , such as its growth rates or the reactor's dilution rate.

##### *Activation-like transition*

An activation-like transition is used when a specific molecule or condition triggers the transition between different subpopulations. This transition occurs when the concentration of the molecule or the strength of the condition increases to a certain threshold. For example, a subpopulation of cells in a slow growth mode (subpopulation A) may transition to a fast growth mode (subpopulation B) when the concentration of a metabolite, such as glucose, increases in the environment, and the subpopulation can sense it in a concentration-dependent manner (measured by ).

We use a Hill function1 to modulate the transition rate based on the metabolite concentration. The Hill function is a sigmoidal curve that varies from 0 to 1 and represents the fraction of cells that transition from subpopulation A to subpopulation B at a given metabolite concentration. The transition rate is multiplied by the Hill function to obtain the effective transition rate, which is then used to compute the contribution towards the derivative of subpopulation B.

is the half-max or constant that is in the same unit as ( if we are talking about a metabolite) and defines the concentration where the transition function equals to (the subpopulation transitions to at half the transition rate).

is the Hill coefficient that determines how steep the transition will occur around the half-max constant (its slope).

The contribution towards the derivative is then:

The model can also support a pure logical rule instead of a Hill function (equivalent to a Hill function with an infinite exponent), for instance:

This can be used initially to fit the model to data (explained below), as it requires less parameters. It can then easily be replaced by the smooth activation function described above.

##### *Inhibition*

Used when the transition is triggered by the 'lack' or 'depletion' of something. For example, when the depletion of a metabolite that is used as an energy source leads to an increase in cell death rate, which is represented by an increase in the rate by which cells in subpopulation A (live) transition to subpopulation B (dead). The transition function used for inhibition is similar to the activation function, but is inversed:

The pure logical rule would be:

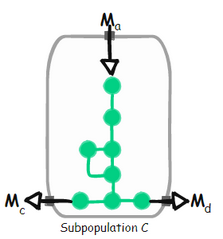
### Metabolites

Species interactions and survival within an environment are largely shaped by the production and consumption of metabolites. In our model, we identified some of the nutrients available in the growth medium used, namely the [Wilkins-Chagrin anaerobic broth](https://www.thermofisher.com/order/catalog/product/CM0643B). This medium contains glucose and pyruvate as added carbon sources. In addition, we measured the time series concentration of trehalose, a disaccharide produced by yeast and introduced into the medium through yeast extract. We also confirmed the presence of glutamate and mannose through single measurements (not time series).

Based on these observation and certain justified assumptions, we included in the model the concentration of several compounds, including glucose, pyruvate, trehalose, lactate, acetate, succinate, butyrate, formate, glutamate, and mannose. These compounds are consumed or produced by at least one of the three species in our model, as we explained in the model state equations (described below).

### Feeding terms

Active subpopulations utilize nutrients for growth and maintenance, leading to the secretion of by-products. This process can be conceptualized as a directed graph, with nodes representing metabolites and edges representing biochemical reactions that transform one metabolite into another. The graph initiates with nutrients present in the medium and follows the flow of metabolites through the network until the by-products are secreted back into the medium, as illustrated below:



Mathematically, the consumption of metabolite and production of metabolites and are connected by the consumption rate of metabolite . For consumption, we employ the Monod equation2 (from here on we denote to the concentration of metabolites measured in and the subpopulation counts measured in by their identifying names).

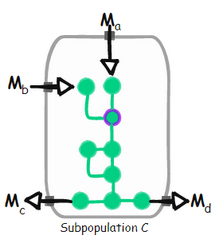
Where represent the relative contribution (weight) of the subpopulation growth towards the consumption (if negative) or production (if positive) of the respective metabolite and is the Monod constant.

In our model, we represented each species' core energy metabolism as an independent subnetwork that we extracted from its respective genome-scale metabolic model. We focused on the fraction of the network that could explain the changes in metabolite concentration that were measured during growth in monocultures. This approach was supported by the analysis of gene expression and differential expression of metabolic genes (see Figure 1 in the main text).

Analyzing the core energy graphs allows us to link subpopulations to sets of metabolites that contribute to their growth. These sets of metabolites are represented as feeding terms, which can display either an "or" (additive) relationship or an "and" (multiplicative) relationship between metabolites, based on the topology of the metabolic network.

Sometimes, a feeding term may require more than one metabolite to contribute to growth, while in other cases, a second metabolite may boost growth but not independently contribute to it. Below, we provide artificial toy examples of these two cases; Keep in mind, however, that the relationships among co-utilized metabolites are primarily determined by the topology of the metabolic network (see the model's state equations below).

#### AND



The network requires the influx of both and to function (otherwise the intracellular metabolite that is highlighted in purple cannot be produced)

#### 

#### Boost

may increase the flux through the network, but only in the presence of . If is absent, the network can still be functional, provided thar is present, but not the other way around.

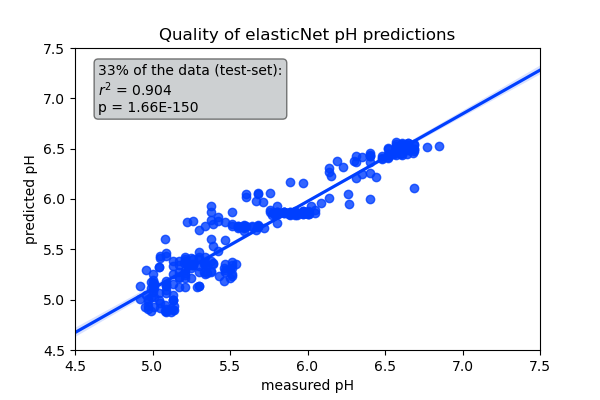
### Environment (reactor)

Environmental conditions substantially influence microbial growth and interactions. Changes in the environment can lead to shifts in both the taxonomic composition and the function of the microbial system. A common approach to understanding these impacts is through collecting experimental data under varying conditions. However, we can take this a step further by encoding our knowledge into a dynamic model that explicitly accounts for environmental conditions. Once calibrated with parameters best fitting the experimental data, this model helps us predict the impact of environmental conditions on our system. These predictions, can be validated through further experiments, and if the results diverge, they serve to highlight key knowledge gaps.

Our model considers the impact of different environmental conditions by encoding the system's nutrient flows and pH dynamics. Specifically, we consider the consumption and production of metabolites by growing subpopulations combined with their responses to environmental changes. The composition of the environment affects the pH, which, in turn, impacts the growth and survival of the different subpopulations.

#### Environment pH

To predict pH changes in the environment, we developed an elastic net model using paired measurements of pH and fermentation acid concentrations. The model considers the concentrations of lactate, acetate, formate, and butyrate to estimate the environmental pH. We trained it using a randomly selected subset constituting 67% of all our experiments, encompassing both monocultures and cocultures. We evaluated the model's performance by comparing predicted pH values with actual measurements from the remaining 33% of the test cases, which were not part of the model training process.



#### pH limitation

In our model, the pH value at each time point is determined by the elastic net model, which is based on the concentrations of fermentation acids. The pH varies as the subpopulations consume and produce these acids. The fluctuating pH influences the growth rates of the subpopulations, according to their preferred pH and their sensitivity to pH variations. It is important to note that our model is only concerned with pH fluctuations below the initial anaerobic pH of the system (6.7), as we did not observe any pH increases beyond this value.

To model the impact of pH on growth, we used the probability density function of the gamma distribution, with some modifications. The gamma distribution is commonly used in statistics to model positive random variables with a skewed distribution. Since we aim to multiply the pH sensitivity by the growth rate, we require a function (referred to as ) that takes the current pH as an input and returns a value between zero and one. This output depends on the subpopulation's sensitivity to deviations from its preferred pH. The function must output a value of one at the optimal pH (or zero if we refer to pH sensitivity, where sensitivity = ).

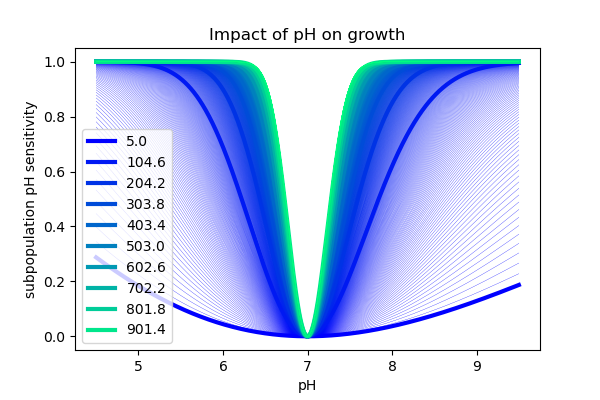
Starting from the probability distribution function of the gamma distribution:

Where and are the shape and rate parameters of the gamma distribution and is the gamma function. To obtain the properties we desire, we parameterized the above function with the optimal pH instead of the rate (), and we wanted its maximum value to be exactly one. To achieve this, we set the mode ( ) of the gamma distribution to be the optimal pH and solve for :

Next, for the optimal pH to return a value of one, we divide the function by its maximum value. Note that the maximum value of the function is exactly at the optimal pH. Tying it together, the impact of pH on the growth of subpopulation is modelled by:

( is defined above)

Below we show the pH sensitivity of a subpopulation that has for different values of (note that needs to be greater than one):



### 

### Growth

We now have all the ingredients for our general growth model. Our general growth model comprises a kinetic equation for each subpopulation, describing changes in its concentration in units of , comparable to flow cytometer counts. These growth equations have four components:

a) Maximum intrinsic growth rate ()

This represents the maximum growth rate that a subpopulation's physiology supports. In the absence of any limiting factors, cells would grow at a rate equal to .

b) pH limitation ()

This component considers the impact of pH on growth, using a modified probability density function of the gamma distribution. The function returns a value between zero and one, depending on how sensitive the subpopulation is to deviations from its preferred pH.

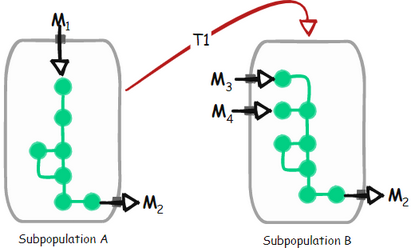
c) Nutrient consumption

This is based on the additive combination of the feeding terms that are actively used by a subpopulation.

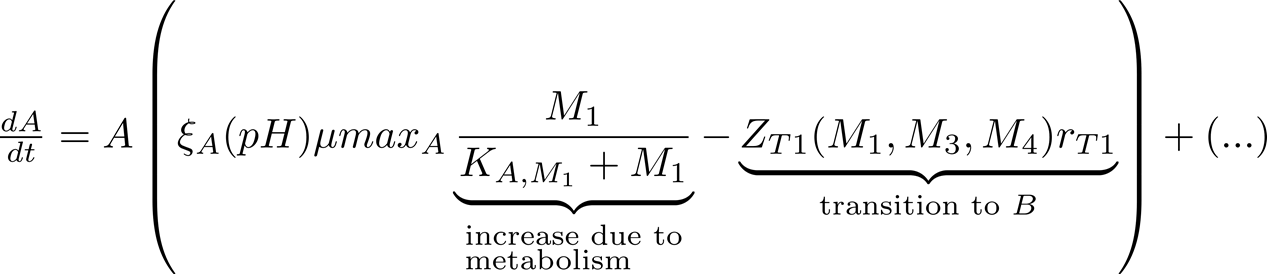
d) Subpopulation transitions

This component considers incoming and outgoing subpopulation transitions.

Next, we illustrate the model equations for the following toy diagram where subpopulation grows on and secretes . When is depleted from the media, if both and are available, it switches to a new growth mode, summarized by subpopulation that grows by simultaneously consuming and while still secreting .

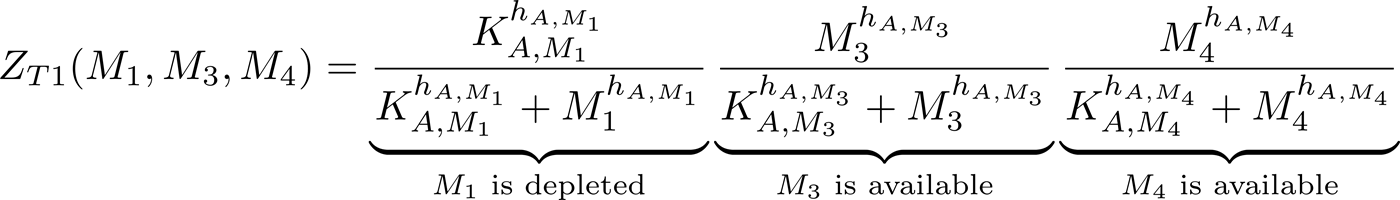


* Growth of subpopulation A



is explained in the “pH limitation” session above.

The transition function is:



* Growth of subpopulation



* Consumption of metabolite

* Consumption of metabolite

* Consumption of metabolite
* Production of metabolite
* pH

where is the elastic net weight attributed to .

### Pulses

In addition to subpopulations growing by the exchange of metabolites and influenced by the environmental pH, the community is also influenced by the environmental regime, particularly by the way in which matter flows in and out of the system. In our model, the environmental regime is controlled by subdividing the simulation into arbitrary time intervals, which we refer to as "pulses". Within a "pulse", the following events are supported:

* Noncontinuous inflow () or outflow () of volume, which is performed at a single step at the beginning of the "pulse". Multiple "pulses" are required to simulate multiple influx and outflux events as in a serial passage experiment. The content of the influx volume, defined by the user, could include fresh metabolites or even spent media from another culture. The user may also set its pH and, if necessary, a specific concentration of bacteria to simulate migration.
* Continuous inflow () and outflow () of volume per time unit, added or removed from the reactor during a "pulse". Similar to the non-continuous events, the user defines the metabolome, microbiome, and pH of the feed.

With these events, a wide range of community regimes can be simulated. For example, one may start a pulse in a batch culture, then transition to a chemostat with a nutrient-rich medium, then move back to a batch culture. Simulations could include serial passages or even the mass transfer between separate reactors, allowing the exploration of regimes that lead to multistability and trigger alternative community states.

# State equations

Here, we define the state variables and equations that simulate bacterial growth and life strategies in our three-species synthetic community (depicted in Figure 2A of the main text). The model components are assigned according to the description of the generic model explained above and informed by our investigation of the growth kinetics of each species (see Figure 1 in the main text). We assigned parameters that represent the best fit to three independent monoculture experiments, each with two or more biological replicates.

Some of the notable features that we found while investigating the physiology of our bacteria in WC media in experiments were incorporated into the model and are summarized below (also see Figure 1 of the main text):

##### *Blautia hydrogenotrophica* DSM 10507 (referred to as "Bh").

1. Bh shows preference for trehalose over glucose:
   * When both trehalose () and glucose () are present in the medium, Bh first consumes trehalose and neglects glucose until trehalose is depleted.
   * Bh's genome contains a trehalose-specific PTS gene, which is found to be overexpressed when Bh grows on trehalose. Interestingly, the genome does not contain the glucose-specific IIA component of the PTS system gene that is found in closely-related *Blautia* and *Ruminococcus* strains. In the presence of trehalose, the non-PTS glucose transporter is inhibited while the trehalose-specific PTS gene is actively expressed. To capture this behavior in our model, we used the transition function (), which triggers a switch from a subpopulation () that does not uptake glucose to a subpopulation that uptakes glucose () based on the concentration of trehalose.
   * We found that adding a higher concentration of trehalose to the media () completely prevented Bh from shifting to glucose consumption. This behavior was not observed when increasing the concentration of pyruvate.
   * We also found that increasing trehalose leads to an increase in lactate production, while increasing pyruvate results in an increase in acetate, but not lactate production.
   * In standard WC medium, pyruvate was depleted before glucose was consumed. However, when we supplemented the medium with pyruvate, we observed co-consumption of glucose and pyruvate. In contrast, the presence of trehalose inhibits the uptake of glucose.
2. Bh exhibits higher growth rates on glucose compared to trehalose and pyruvate:
   * We observed higher growth-rates during the glucose-consuming stage compared to the trehalose-consuming stage.
3. Bh exhibits glucose co-limitation:
   * Not all of the glucose is consumed from the media before the culture enters the stationary and death phases, suggesting co-limitation with another substrate. The core-metabolic pathway suggests that growth on glucose would be favored by glutamate fermentation, which would provide an additional mol of CO2 and a reduced ferredoxin that could be used to pump protons through the RNF system. This favors ATP production through the membrane ATPase (ATPS4), which is driven by a proton gradient. We modeled this behavior as a co-consumption of glucose and glutamate as the genes for glutamate fermentation are significantly overexpressed during growth on glucose. We confirmed glutamate depletion from the spent media by measuring the levels of amino acids before and after fermentation.

##### *Bacteroides thetaiotaomicron* VPI-5482 (referred to as "Bt")

1. Bt produces a range of fermentation acids that significantly decrease the medium pH:
   * In our experiments, Bt was observed to rapidly consume glucose and pyruvate, while producing a variety of fermentation acids. This resulted in a swift drop in the medium's pH.
2. Bt is inhibited by low pH values:
   * Bt is known to be sensitive to low pH levels3, a fact we corroborated by incubating the cells across different pH ranges. These experiments revealed that while the population did not grow at pH levels < 5, most cells remained viable. As such, we modelled pH's impact as a growth inhibitor.
   * But we also found that when carbon sources are exhausted most Bt cells lose their viability. This was confirmed through assessing cell permeability with PI staining. To represent this in our model, we introduced transition functions from active to inactive subpopulations that are triggered by nutrient depletion at low pH.
3. Bt fixes CO2, producing succinate:
   * Bt fixes CO2 by converting phosphoenolpyruvate (a C3 molecule) into oxaloacetate (a C4 molecule) in a process that mirrors carbon fixation in plants. Via the reductive carboxylation of phosphoenolpyruvate4, Bt generates ATP and produces succinate.
4. Bt shows a second growth peak in WC medium:
   * We consistently observed a second growth peak before the majority of the cell population transitioned to an inactive state. We attributed this second peak to the consumption of mannose, which is present in low concentrations in the complex medium. Mannose depletion was confirmed through single-point measurements and suggested by the gene expression data.

##### *Roseburia intestinalis* L1-82 (referred to as "Ri")

1. Ri is a butyrate producer:
   * By studying its core metabolic pathway, we found that Ri produces butyrate through the reverse beta oxidation pathway. In our experiments, Ri quickly consumed glucose and pyruvate and produced butyrate, acetate, and lactate.
2. Ri has lesser impact on the medium pH
   * Unlike Bt, Ri exerts a weaker effect on the pH of the medium despite its high growth rate.
3. Ri enters a slow growth mode characterized by the consumption of lactate and acetate:
   * We observed that some of the lactate and acetate that are produced during growth in glucose and pyruvate, later get consumed, leading to a gradual increase in butyrate.
   * Following the consumption of glucose and pyruvate and production of butyrate, lactate, and acetate, most cells burst and are no longer detected by flow cytometry. However, a subset of cells can persist for several days, possibly entering a slow growth mode5. We modeled this behavior by having cells quickly die in the absence of glucose and transition to a slow growth mode when triggered by lactate.

### States

| symbol | type | state | units |
| --- | --- | --- | --- |
|  | subpopulation | Bh subpopulation that consumes trehalose |  |
|  | subpopulation | Bh subpopulation that consumes glucose |  |
|  | subpopulation | Bh inactive subpopulation |  |
|  | subpopulation | Bh dead subpopulation |  |
|  | subpopulation | Bt subpopulation that consumes glucose |  |
|  | subpopulation | Bt subpopulation that consumes mannose |  |
|  | subpopulation | Bt inactive subpopulation |  |
|  | subpopulation | Bt dead subpopulation |  |
|  | subpopulation | Ri subpopulation that consumes glucose |  |
|  | subpopulation | Ri subpopulation that consumes lactate and acetate |  |
|  | subpopulation | Ri inactive subpopulation |  |
|  | subpopulation | Ri dead subpopulation |  |
|  | metabolite | trehalose |  |
|  | metabolite | pyruvate |  |
|  | metabolite | glucose |  |
|  | metabolite | glutamate |  |
|  | metabolite | lactate |  |
|  | metabolite | acetate |  |
|  | metabolite | mannose |  |
|  | metabolite | succinate |  |
|  | metabolite | formate |  |
|  | metabolite | butyrate |  |
| pH | pH | potential of hydrogen |  |

### 

### Equations

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Bh subpopulation that grows on trehalose () or pyruvate (), and produces acetate () and lactate () |

The function is explained in the “pH limitation” section of the generic model.

transition from to , triggered by low trehalose concentration ():

transition from to triggered by high trehalose concentration ():

transition from to :

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Bh subpopulation that grows on glucose () and glutamate (), and produces acetate () |

transition from to triggered by low concentrations of glucose () or glutamate ():

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | Inactive Bh subpopulation that is PI-positive and eventually bursts |

fixed burst rate:

# 

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Bt subpopulation that grows on glucose () or pyruvate (), and produces acetate (), lactate (), formate (), and succinate () |

transition from to triggered by low glucose () and high mannose () concentrations:

transition from to triggered by low glucose concentration and low pH:

transition from to triggered by high glucose:

| Diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Bt subpopulation that grows on mannose (), and produces acetate () and succinate () |

transition from to triggered by low mannose concentration () and low pH:

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | inactive Bt subpopulation that is PI-positive and eventually bursts |

Fixed burst rate:

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Ri subpopulation that grows on glucose () or pyruvate (), and produces acetate (), lactate (), and butyrate () |

transition from to triggered by high lactate concentration ():

transition from to triggered by low glucose () concentration and pyruvate ():

transition from to triggered by high concentration of glucose () or pyruvate ():

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | live Ri subpopulation with slow growth that consumes acetate () and/or lactate (), and produces small amounts of butyrate () |

Fixed death rate:

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | inactive Ri subpopulation that is (PI-positive) and eventually bursts |

Fixed burst rate:

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | Trehalose concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | pyruvate concentration |

| Diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | glucose concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | glutamate concentration |

| Diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | lactate concentration |

| Diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | acetate concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | mannose concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | succinate concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | formate concentration |

| diagram | state | unit | description |
| --- | --- | --- | --- |
|  |  |  | butyrate concentration |

### Parameters

| # | species | Parameter | symbol | fitted values | description |
| --- | --- | --- | --- | --- | --- |
| 1 | Bh | xa\_mumax |  |  | maximum growth rate of |
| 2 | Bh | xb\_mumax |  |  | maximum growth rate of |
| 3 | Bh | xa\_pHopt |  |  | optimal pH |
| 4 | Bh | xb\_pHopt |  |  | optimal pH |
| 5 | Bh | xa\_pHalpha |  |  | pH sensitivity |
| 6 | Bh | xb\_pHalpha |  |  | pH sensitivity |
| 7 | Bh | xa\_k\_s1 |  |  | Monod constant for trehalose () consumption by |
| 8 | Bh | xa\_k\_s2 |  |  | Monod constant for pyruvate () consumption by |
| 9 | Bh | xb\_k\_s3 |  |  | Monod constant for glucose () consumption by |
| 10 | Bh | xb\_k\_s4 |  |  | Monod constant for glutamate () consumption by |
| 11 | Bh | xb\_k\_s2 |  |  | Monod constant for pyruvate () consumption by |
| 12 | Bh | xa\_g\_s1 |  |  | rate constant for trehalose () consumption by |
| 13 | Bh | xa\_g\_s2 |  |  | rate constant for pyruvate () consumption by |
| 14 | Bh | xa\_g\_s6\_s1 |  |  | rate constant for acetate () production from consuming trehalose () |
| 15 | Bh | xa\_g\_s5\_s1 |  |  | rate constant for lactate () production from consuming trehalose () |
| 16 | Bh | xa\_g\_s6\_s2 |  |  | rate constant for acetate () production from consuming pyruvate () |
| 17 | Bh | xb\_g\_s3 |  |  | rate constant for glucose () consumption by |
| 18 | Bh | xb\_g\_s4 |  |  | rate constant for glutamate () consumption by |
| 19 | Bh | xb\_g\_s2 |  |  | rate constant for pyruvate () consumption by |
| 20 | Bh | xb\_g\_s6\_s3\_s4 |  |  | rate constant for acetate () production from consuming glucose () and glutamate () |
| 21 | Bh | xb\_g\_s6\_s2 |  |  | rate constant for acetate () production from consuming pyruvate () |
| 22 | Bh | z1\_r |  |  | rate of transition from to |
| 23 | Bh | z2\_r |  |  | rate of transition from to |
| 24 | Bh | z3\_r |  |  | rate of transition from to |
| 25 | Bh | z4\_r |  |  | rate of transition from to |
| 26 | Bh | z5\_r |  |  | rate of transition from to |
| 27 | Bh | z1\_l\_s1 |  |  | half saturation constant for trehalose () |
| 28 | Bh | z2\_l\_s1 |  |  | half saturation constant for trehalose () |
| 29 | Bh | z4\_l\_s3\_s4 |  |  | half saturation constant for glucose () and glutamate () |
| 30 | Bh | z1\_h\_s1 |  |  | Hill coefficient for trehalose () |
| 31 | Bh | z2\_h\_s1 |  |  | Hill coefficient for trehalose () |
| 32 | Bh | z4\_h\_s3\_s4 |  |  | Hill coefficient for glucose () and glutamate () |
| 33 | Bt | xe\_mumax |  |  | maximum growth rate of |
| 34 | Bt | xf\_mumax |  |  | maximum growth rate of |
| 35 | Bt | xe\_pHopt |  |  | optimal pH |
| 36 | Bt | xf\_pHopt |  |  | optimal pH |
| 37 | Bt | xe\_pHalpha |  |  | pH sensitivity |
| 38 | Bt | xf\_pHalpha |  |  | pH sensitivity |
| 39 | Bt | xe\_k\_s3 |  |  | Monod constant for glucose () consumption by |
| 40 | Bt | xe\_k\_s2 |  |  | Monod constant for pyruvate () consumption by |
| 41 | Bt | xf\_k\_s7 |  |  | Monod constant for mannose () consumption by |
| 42 | Bt | xe\_g\_s2 |  |  | rate constant for pyruvate () consumption by |
| 43 | Bt | xe\_g\_s3 |  |  | rate constant for glucose () consumption by |
| 44 | Bt | xe\_g\_s5\_s2 |  |  | rate constant for lactate () production from consuming pyruvate () |
| 45 | Bt | xe\_g\_s6\_s2 |  |  | rate constant for acetate () production from consuming pyruvate () |
| 46 | Bt | xe\_g\_s6\_s3 |  |  | rate constant for acetate () production from consuming glucose () |
| 47 | Bt | xe\_g\_s8\_s3 |  |  | rate constant for succinate () production by consuming glucose () |
| 48 | Bt | xe\_g\_s9\_s2 |  |  | rate constant for formte () production by consuming pyruvate () |
| 49 | Bt | xf\_g\_s7 |  |  | rate constant for mannose () consumption by |
| 50 | Bt | xf\_g\_s6\_s7 |  |  | rate constant for acetate () prouduction from consuming mannose () |
| 51 | Bt | xf\_g\_s8\_s7 |  |  | rate constant for succinate () production by consuming mannose () |
| 52 | Bt | z6\_r |  |  | rate of transition from to |
| 53 | Bt | z7\_r |  |  | rate of transition from to |
| 54 | Bt | z8\_r |  |  | rate of transition from to |
| 55 | Bt | z9\_r |  |  | rate of transition from to |
| 56 | Bt | z10\_r |  |  | rate of transition from to |
| 57 | Bt | z6\_l\_s3 |  |  | half-saturation constant for glucose () |
| 58 | Bt | z6\_l\_s7 |  |  | half-saturation constant for mannose () |
| 59 | Bt | z7\_l\_s3 |  |  | half-saturation constant for glucose () |
| 60 | Bt | z7\_l\_pH |  |  | half-saturation constant for the pH |
| 61 | Bt | z8\_l\_s7 |  |  | half-saturation constant for mannose () |
| 62 | Bt | z8\_l\_pH |  |  | half-saturation constant for the pH |
| 63 | Bt | z10\_l\_s3 |  |  | half-saturation constant for glucose () |
| 64 | Bt | z6\_h\_s3 |  |  | Hill coefficient for glucose () |
| 65 | Bt | z6\_h\_s7 |  |  | Hill coeffiecient for mannose () |
| 66 | Bt | z7\_h\_s3 |  |  | Hill coefficient for glucose () |
| 67 | Bt | z7\_h\_pH |  |  | Hill coefficient for the pH |
| 68 | Bt | z8\_h\_s7 |  |  | Hill coefficient for mannose () |
| 69 | Bt | z8\_h\_pH |  |  | Hill coefficient for the pH |
| 70 | Bt | z10\_h\_s7 |  |  | Hill coefficient for mannose () |
| 71 | Ri | xi\_mumax |  |  | maximum growth rate of |
| 72 | Ri | xj\_mumax |  |  | maximum growth rate of |
| 73 | Ri | xi\_pHopt |  |  | optimal pH |
| 74 | Ri | xj\_pHopt |  |  | optimal pH |
| 75 | Ri | xi\_pHalpha |  |  | pH sensitivity |
| 76 | Ri | xj\_pHalpha |  |  | pH sensitivity |
| 77 | Ri | xi\_k\_s2 |  |  | Monod constant for pyruvate () by |
| 78 | Ri | xi\_k\_s3 |  |  | Monod constant for glucose () by |
| 79 | Ri | xj\_k\_s5 |  |  | Monod constant for lactate () by |
| 80 | Ri | xj\_k\_s6 |  |  | Monod constant for acetate () by |
| 81 | Ri | xi\_g\_s2 |  |  | rate constant for pyruvate () consumption by |
| 82 | Ri | xi\_g\_s3 |  |  | rate constant for glucose () consumption by |
| 83 | Ri | xj\_g\_s5 |  |  | rate constant for lactate () consumption by |
| 84 | Ri | xj\_g\_s6 |  |  | rate constant for acetate () consumption by |
| 85 | Ri | xi\_g\_s5\_s3 |  |  | rate constant for lactate () production from consuming glucose () |
| 86 | Ri | xi\_g\_s6\_s2 |  |  | rate constant for acetate () production from consuming pyruvate () |
| 87 | Ri | xi\_g\_s6\_s3 |  |  | rate constant for acetate () production from consuming glucose () |
| 88 | Ri | xi\_g\_s10\_s2 |  |  | rate constant for butyrate () production by consuming pyruvate () |
| 89 | Ri | xi\_g\_s10\_s3 |  |  | rate constant for butyrate () production by consuming glucose () |
| 90 | Ri | xj\_g\_s10\_s5 |  |  | rate constant for butyrate () production by consuming lactate () |
| 91 | Ri | xj\_g\_s10\_s6 |  |  | rate constant for butyrate () production by consuming acetate () |
| 92 | Ri | z11\_r |  |  | rate of transition from to |
| 93 | Ri | z12\_r |  |  | rate of transition from to |
| 94 | Ri | z13\_r |  |  | rate of transition from to |
| 95 | Ri | z14\_r |  |  | rate of transition from to |
| 96 | Ri | Z15\_r |  |  |  |
| 97 | Ri | z11\_l\_s5 |  |  | half-saturation constant for lactate () |
| 98 | Ri | z12\_l\_s3\_s2 |  |  | half-saturation constant for glucose () and pyruvate () |
| 99 | Ri | z15\_l\_s3\_s2 |  |  | half-saturation constant for glucose () and pyruvate () |
| 100 | Ri | z11\_h\_s5 |  |  | Hill coefficient for lactate () |
| 101 | Ri | z12\_h\_s3\_s2 |  |  | Hill coefficient for glucose () and pyruvate () |
| 102 | Ri | z15\_h\_s3\_s2 |  |  | Hill coefficient for glucose () and pyruvate () |

Computational implementation of the model

## Parameter database

The model structure and parameters values can be obtained from an sqlite3 database.

First, one must create a database schema, populate its tables with parameters, then load the database for simulations or parameter fitting. One may also load specific parameter values into a model database file.

These operations are easily performed with the scripts explained below.

### Creating a database schema (database file)

To generate a database file with the correct model schema, run the script: "/scripts/db/makeSchema.py" [here](https://github.com/danielriosgarza/hungerGamesModel/blob/main/scripts/db/makeSchema.py). First specify the name of the database file in the top of the script. By default, the database file will be stored in: ".../files/dbs/".

For example, creating a database to simulate grow of the three species (Bh, Bt, and Ri):

* open the script (".../scripts/db/makeSchema.py") in a text editor add the desired database name:

import os  
from pathlib import Path  
import sqlite3  
  
#Replace this name for the desired name.  
databaseName = 'modelDB\_bhbtri.sqlite3'

* Run the script.
* Check that "modelDB\_bhbtri.sqlite3" file was created in the folder ".../files/dbs/"

To edit and manipulate the database, it is useful to use a graphical interface database browser. I recommend using [DB Browser for SQlite](https://sqlitebrowser.org/)

Here is how it looks:

![fig:](data:text/html; charset=utf-8;base64,)

### Populating the database

Once the schema is created, one can load it into the browser and fill the parameters manually. Alternatively, run the script (".../scripts/db/inserParameters.py") [here](https://github.com/danielriosgarza/hungerGamesModel/blob/main/scripts/db/insertParameters.py). This script populates the database with default parameters tables (located in ".../files/strainSummaries/dbsTemplateTables").

Set the "databaseName" variable to the desired database file.

To modify, either edit the corresponding "tsv" table or edit tabs of the excel file "dbTemplate.xlsx" and save them as a ".tsv" file with their corresponding table name. The database tables are:

#### elements

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/elements.tsv)

stores information about chemical elements. The fields are:

* id (atomic symbol)
* name
* MolecularWeight

#### metabolites

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/metabolites.tsv)

* id
* color (used in the kinetic plots)
* MolecularWeight

#### metabolite2elements

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/metabolites2elements.tsv)

stores the relation between metabolites and chemical elements

* id (unique integer)
* metabolite
* element
* atoms

#### wc

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/wc.tsv)

stores the concentration of components in the media

* metabolite
* concentration (in mM)

#### species

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/species.tsv)

stores metadata for each species

* id (bt, bh, ri)
* name
* genomeSize
* geneNumber
* patricID
* ncbiID

#### feedingTerms

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/feedingTerms.tsv)

stores the feeding terms for each species

* id
* name
* species

#### feedingTerms2metabolites

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/feedingTerms2metabolites.tsv)

* id
* feedingTerm (the feeding term id)
* metabolite
* yield (positive for consumed, negative for produced metabolites)
* monodK (Monod's constant, zero for produced metabolites)

#### subpopulations

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/subpopulations.tsv)

stores information about the intraspecific subpopulations

* id
* species
* mumax (maximum growth rate)
* pHoptimal
* pHalpha
* count
* color
* state (active, inactive, dead)

#### subpopulations2subpopulations

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/subpopulations2subpopulations.tsv)

stores the transitions between intra-species subpopulations

* id
* subpopulation\_A
* subpopulation\_B
* hillFunc
* rate

#### subpopulations2feedingTerms

[template example](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/strainSummaries/dbsTemplateTables/subpopulations2feedingTerms.tsv)

stores the relation between subpopulations and feeding terms

* id
* subpopulation
* feedingTerm

## Loading parameters from a database and performing your own simulations

* First load the necessary packages

from pathlib import Path  
import os  
import sys  
  
#assuming the current path id /scripts/core import the main model classes  
from mainClasses import \*  
  
#add the database scripts to the sys.path and import them  
sys.path.append(os.path.join(Path(os.getcwd()).parents[0], 'db'))  
from readModelDB import \*  
  
#import plotly to make plots that are rendered in the browser  
import plotly.io as pio  
pio.renderers.default='browser'

* load the parameter database and file with pH and acid concentration to be used as an input to the linear model that predicts the pH

#pH file  
ipH\_path = os.path.join(Path(os.getcwd()).parents[1], 'files', 'strainSummaries', 'bhbtri\_ipH4.tsv')   
#databse file with parameters  
databaseName = 'modelDB\_bhbtri.sqlite3'  
databaseFolder = os.path.join(Path(os.getcwd()).parents[1], 'files', 'dbs')  
  
#Load the database  
db = get\_database(os.path.join(databaseFolder, databaseName))

* get the starting pH from the default WC metabolome

#getStarting pH  
wc = createMetabolome(db, 'wc')  
predictpH = getpH(wc.metabolites, ipH\_path) #this function will be used to predict the pH  
pH = predictpH(wc.get\_concentration()) #is the starting pH

* load the metabolomes and microbiome that will be used in the reactor and in the feed. Since we are not simulating migration, make them steril

wc\_f = createMetabolome(db, 'wc', pH, pHFunc=predictpH)  
wc\_r = createMetabolome(db, 'wc', pH, pHFunc=predictpH)  
  
#get the feed obj. Make it sterile  
bhbtri\_f = Microbiome({'bh':createBacteria(db, 'bh', 'wc'), 'bt':createBacteria(db, 'bt', 'wc'), 'ri':createBacteria(db, 'ri', 'wc')})  
bhbtri\_f.subpopD['xa'].count = 0  
bhbtri\_f.subpopD['xe'].count = 0  
bhbtri\_f.subpopD['xi'].count = 0  
  
#create the reactor obj, with starting populations  
bhbtri\_r = Microbiome({'bh':createBacteria(db, 'bh', 'wc'), 'bt':createBacteria(db, 'bt', 'wc'), 'ri':createBacteria(db, 'ri', 'wc')})

* create the culture regime (here, batch with 120 h).
* The regime is implemented by "pulses". The pulse class requires the following inputs:
  + metabolome
  + microbiome
  + t\_start
  + t\_end
  + n\_steps
  + vin (initial inflow)
  + vout (initial outflow)
  + qin (continuous inflow, volume in mL added per hour)
  + qout (continuous outflow, volume in mL removed per hour).

Different regimes and perturbations can be created by a list of "Pulse" objects.

batchA = Pulse(wc\_f, bhbtri\_f, 0, 120, 10000, 0, 0, 0,0)

* run a simulation and plot the results

r\_bhbtri = Reactor(bhbtri\_r, wc\_r,[batchA], 60)  
r\_bhbtri.simulate()  
r\_bhbtri.makePlots()

## Loading specific model parameters into a desired database file

It is not necessary to create a new database or to change the template file to use different parameter settings. This can be done automatically from a parameter file.

For example,

* consider the model parameters described [here](https://github.com/danielriosgarza/hungerGamesModel/wiki/State-equations)
* with "tsv" files structured as:
  + [bh](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/params/bh.tsv)
  + [bt](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/params/bt.tsv)
  + [ri](https://github.com/danielriosgarza/hungerGamesModel/blob/main/files/params/ri.tsv)
* we can load parameters into a database of choice and simulate the model.

from pathlib import Path  
import os  
import sys  
  
sys.path.append(os.path.join(Path(os.getcwd()).parents[0], 'db'))  
  
from readModelDB import \*  
from mainClasses import \*  
from loadParameters import \*  
  
import plotly.io as pio  
pio.renderers.default='browser'  
  
ipH\_path = os.path.join(Path(os.getcwd()).parents[1], 'files', 'strainSummaries', 'bhbtri\_ipH4.tsv')   
  
databaseName = 'modelDB\_bhbtri.sqlite3'  
  
databaseFolder = os.path.join(Path(os.getcwd()).parents[1], 'files', 'dbs')  
  
#update database with parameters from a file  
##########################################  
  
#create a database connection  
conn = create\_connection(os.path.join(databaseFolder, databaseName))  
  
#load the parameter file (parameter files are located at "/files/params" )  
bh\_params = getPramsFromFile('bh', os.path.join(Path(os.getcwd()).parents[1], 'files', 'params', 'bh.tsv'))  
  
bt\_params = getPramsFromFile('bt', os.path.join(Path(os.getcwd()).parents[1], 'files', 'params', 'bt.tsv'))  
  
ri\_params = getPramsFromFile('ri', os.path.join(Path(os.getcwd()).parents[1], 'files', 'params', 'ri.tsv'))  
  
  
#assign these parameters (depending on the strain, use the specific function)  
assignBhParams(bh\_params, conn)  
  
assignBtParams(bt\_params, conn)  
  
assignRiParams(ri\_params, conn)  
  
#Load database  
db = get\_database(os.path.join(databaseFolder, databaseName))

This script loaded the parameter file and altered the database values. Now the simulation may proceed as described above with the new parameter values. This quick parameter assigment allows us to fit the parameters to experimental data. ([see](https://github.com/danielriosgarza/hungerGamesModel/wiki/Fitting-models-to-experimental-data))

## Finding model parameters

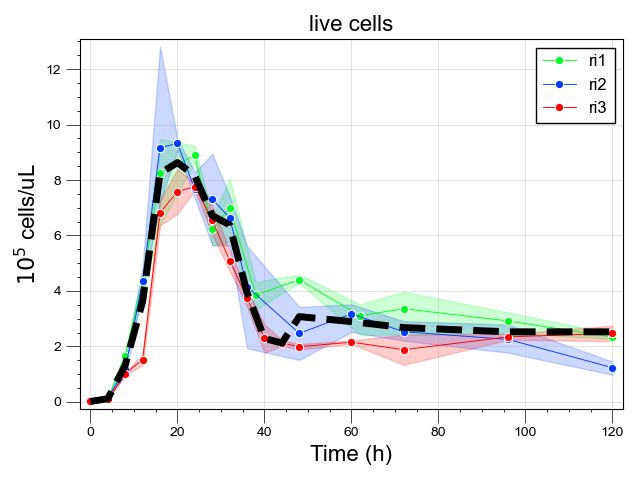
We used batch monoculture experiments to estimate the parameters for the state equations in our model. The codes that execute the complete procedure described below for each of the strains are available [here](https://github.com/danielriosgarza/hungerGamesModel/tree/main/scripts/parameterFIt).

### Moving average

We carried out three sets of monoculture measurements for cell counts, metabolite concentrations, and pH, with each set including at least two replicates. These experiments, conducted under nearly identical conditions, showed consistent kinetics across different days, despite some variability (e.g. lactate). We averaged measurements within regular intervals to establish a consensus kinetic curve for each state, which was used to estimate the model's kinetic parameters. Below, we provide a python code example of how we derived the moving average from three independent experiments, where we measured the concentration of live *Roseburia intestinalis* cells:

import os  
import sys  
from pathlib import Path  
import numpy as np  
import matplotlib.pyplot as plt  
  
sys.path.append(os.path.join(Path(os.getcwd()).parents[0], 'compare2experiments'))  
from general import \*  
from parseTable import \*  
  
#chose a species  
species = 'ri'  
  
#chose replicates to summarize  
experiments = ['bhri', 'btri', 'bhbtri']  
  
#location of the experiment data folder  
strainSummaryFolder = os.path.join(Path(os.getcwd()).parents[1], 'files', 'strainSummaries', species)  
  
#chose the state ('live', 'dead', 'glucose', 'acetate', 'pH', etc.)  
measuredState = 'live'  
  
#chose the state type for the plot  
stateType = 'cells'  
  
#chose the regular interval  
intervals = 4  
  
#get the data  
stFile = parseTable(os.path.join(strainSummaryFolder, measuredState + '.tsv'))  
df\_state = getDFdict(stFile, measuredState, False)  
  
labels = ['ri1', 'ri2', 'ri3']  
colors = ['#00ff26', '#003eff', '#ff0000']  
  
s = summarizeExperiments(df\_state, 'ri\_live', experiments, interval = intervals)  
  
makeExperimentPlot(species, measuredState, stateType, stFile, labels, colors)  
plt.plot(s, lw = 5, ls = '--', color = 'k')  
plt.savefig(os.path.join(Path(os.getcwd()).parents[1], 'files', 'Figures', 'movingAverage.png'), dpi = 100)

The black traced line shows the moving average:



### Cubic splines

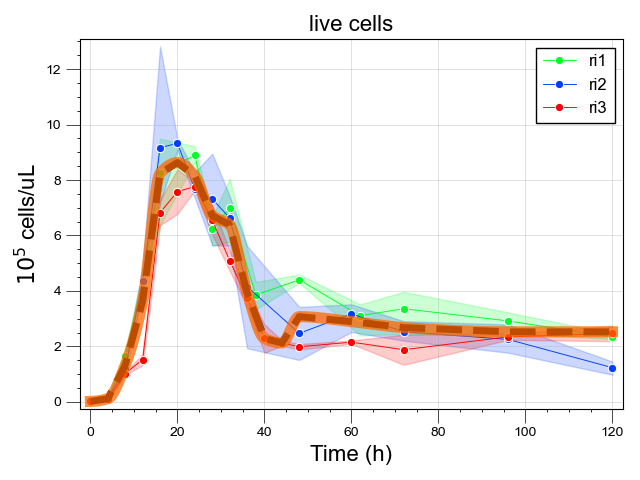
Moving averages were interpolated by cubic-splines, a method for smoothing data. Cubic spline interpolation is a mathematical technique often employed in data analysis to create a smooth curve that intersects a given set of points (in this case, the moving averages). This technique involves constructing a series of piecewise continuous polynomials of degree three that fit the data.

Cubic splines provide the advantage of calculating state values at any given point in time, not just at the sampled intervals. This means we can evaluate the state at any arbitrary point within the range of the measured time-points, not merely at specified time intervals. Using cubic splines, we can directly compare the smoothed values from the data (the spline's function values) with the solutions of our model's differential equations. This comparison is accomplished by using the simulation times from the model as inputs for the cubic spline functions.

Below, we offer a code example that demonstrates the computation of the cubic spline from the moving average illustrated above:

#get spline  
sp = get\_spline('live', strainSummaryFolder, labels, df\_state = s)  
  
#define an arbitrary time interval  
t = np.linspace(0, 120, 10000)  
  
#get spline points  
live = sp(t)  
  
#add to plot  
plt.plot(t, live, lw = 8, color='#FF6700', alpha = 0.75)  
  
plt.savefig(os.path.join(Path(os.getcwd()).parents[1], 'files', 'Figures', 'spline.png'), dpi = 100)

The orange line is the spline:



### Optimization

Finally, we searched for the parameter set that minimizes the difference between the [states](https://github.com/danielriosgarza/hungerGamesModel/wiki/State-equations#states) in our model simulations and their corresponding smoothed moving averages. To compute the discrepancy between the model simulations and the experimental data, we used the Hubber loss function, which is a function that is robust against eventual outliers. We then minimized this loss function using Powell's method, an approach that approximates the function using a quadratic model. The method sequentially minimizes this approximation along one direction at a time, generating a set of mutually conjugate search directions to navigate towards the minimum. The parameters that minimize this loss were subsequently used in all downstream simulations of our model.

In summary, the process of fitting our model to sets of experimental data involved the following steps:

1. Acquiring the moving average from experimental data that represent our model states
2. Fitting cubic splines to these averages to enable a smooth estimation of their values within their measured time frames;
3. Adjusting model parameters by minimizing the Huber loss function between model simulations and the smoothed moving averages (simulated by the cubic splines), using Powell's optimization method.

**Instructions to reproduce the manuscript’s plots and simulations**

Before reproducing the Figures, obtain a local installation of the GitHub repository and the necessary modules.

The repository address is: https://github.com/danielriosgarza/hungerGamesModel

This can be downloaded in a terminal with the command:

Git clone https://github.com/danielriosgarza/hungerGamesModel

Or Windows users can obtain the repository using GitHub desktop app: https://desktop.github.com/

Make sure to read our wiki: https://github.com/danielriosgarza/hungerGamesModel/wiki

Once a local repository is available, use your favorite Python interpreter to run the scripts listed below. Make sure that you have all the pre-requisites installed. Below is a list of them and the versions that were used in this manuscript.

|  |  |
| --- | --- |
| Package | Version |
| aquarel | 0.0.5 |
| cobra | 0.29.0 |
| gurobi | 11.0.0 |
| lmfit | 1.2.2 |
| matplotlib | 3.8.1 |
| numpy | 1.24.3 |
| pandas | 2.1.3 |
| plotly | 5.18.0 |
| pony | 0.7.17 |
| scikit-learn | 1.3.0 |
| scipy | 1.11.3 |
| sdeint | 0.3.0 |
| seaborn | 0.12.2 |
| sqlite | 3.41.2 |
| umap-learn | 0.5.4 |

**Figure 1: Experiments and simulations**

*Gene expression bar charts*

Generated by running the script:

hungerGamesModel\scripts\FiguresReproduce\Figure1\_barCharts.py

This script automatically gets the reactions from the genome-scale metabolic models, matches genes associated to the reactions in the core energy metabolism diagrams to their DeSeq2 differential expression analyses files, and plots their respective fold-changes as shown in Figure 1 A, B, and C.

*Experimental datasets and model simulations*

Generated by running the script:

hungerGamesModel\scripts\FiguresReproduce\Figure1\_dynamicPlots.py

For guidance, we will provide a detailed description of the portion of the script that generates *Blautia hydrogenotryphica*’s Figures. The rest of the script follows and identical procedure for the other two species.

First, import the modules that are needed to locate the files in our repository, parse data, and make plots:

import os

import sys

from pathlib import Path

import pandas as pd

import matplotlib.pyplot as plt

Next, append the folder “compare2experiments” to our path and import its scripts:

sys.path.append(os.path.join(Path(os.getcwd()).parents[0], 'compare2experiments'))

from parseTable import \*

from general import \*

Define the species we want plot (from the options “bh”, “bt”, or “ri”) and the experiment groups we want to plot and give them labels and colors. The experiment groups are reference to independent experiments with biological replicates that in Figure 1 are labeled as “experiment 1”, “experiment 2”, and “experiment 3”.

species = 'bh'

experiments = ['bhbt', 'bhri', 'bhbtri']

labels = ['bh1', 'bh2', 'bh3']

colors = ['#00ff26', '#003eff', '#ff0000']

Next, load model parameters for *Blautia hydrogenotrophica*, these parameters are summarized in the “parameters” table above. Load the model’s database files, which is explained in the database setup, create a database connection and load the parameters to the database.

params = getPramsFromFile('bh', os.path.join(Path(os.getcwd()).parents[1], 'files', 'params', 'bh.tsv'))

databaseName = 'modelDB\_bhbtri.sqlite3'

databaseFolder = os.path.join(Path(os.getcwd()).parents[1], 'files', 'dbs')

database = os.path.join(databaseFolder, databaseName)

conn = create\_connection(os.path.join(databaseFolder, databaseName))

assignBhParams(params, conn)

Select the measured states from which to extract initial conditions:

measuredStates = ['live',

'trehalose',

'pyruvate',

'glucose',

'lactate',

'acetate']

Perform simulations for each experiment using the “simulateExperiment” function that is already loaded in our environment. This will load the database that we set with parameters, load the initial values for the state variables, and perform simulations within the experiment times.

bh1 = simulateExperiment(group = 'bh',

experimentLabel = 'bhbt',

dbPath = database,

measuredStates = measuredStates,

)

bh2 = simulateExperiment(group = 'bh',

experimentLabel = 'bhri',

dbPath = database,

measuredStates = measuredStates

)

bh3 = simulateExperiment(group = 'bh',

experimentLabel = 'bhbtri',

dbPath = database,

measuredStates = measuredStates

)

Select states to make experiment/simulation Figures. Also define their types between the alternatives “cells”, “metabolite”, and “pH”.

states = ['live',

'pH',

'trehalose',

'pyruvate',

'glucose',

'acetate',

'lactate']

stTypes = ['cells',

'pH',

'metabolite',

'metabolite',

'metabolite',

'metabolite',

'metabolite']

Plot the kinetics for each state using the “makeExperimentPlot” function, which can use the simulation objects obtained above (bh1, bh2, and bh3) to get simulated data as well. If these are not provided, only the experimental points will be ploted.

figPath = os.path.join(Path(os.getcwd()).parents[1], 'files', 'Figures', species+'Experiments')

for i,v in enumerate(states):

makeExperimentPlot(species, v, stTypes[i], experiments, labels, colors, simulObj = [bh1, bh2, bh3], alpha=0.5, legend=False)

plt.savefig(os.path.join(figPath, v + 'noLegend\_model.png'), dpi = 300)

plt.show()