

RESEARCH ARTICLE

MASSpy: Building, simulating, and visualizing dynamic biological models in Python using mass action kinetics

Zachary B. Haiman¹, Daniel C. Zielinski¹, Yuko Koike^{1,2}, James T. Yurkovich^{1,2}, Bernhard O. Palsson^{1,3*}

1 Department of Bioengineering, University of California San Diego, La Jolla, California, United States of America, **2** Institute for Systems Biology, Seattle, Washington, United States of America, **3** Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, Kongens Lyngby, Denmark

* palsson@ucsd.edu



OPEN ACCESS

Citation: Haiman ZB, Zielinski DC, Koike Y, Yurkovich JT, Palsson BO (2021) MASSpy: Building, simulating, and visualizing dynamic biological models in Python using mass action kinetics. *PLoS Comput Biol* 17(1): e1008208. <https://doi.org/10.1371/journal.pcbi.1008208>

Editor: Pedro Mendes, University of Connecticut School of Medicine, UNITED STATES

Received: July 27, 2020

Accepted: December 21, 2020

Published: January 28, 2021

Copyright: © 2021 Haiman et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: The source code for MASSpy is available online (<https://github.com/SBRG/MASSpy>) and in the [S1 File](#) under an MIT license. MASSpy is also hosted as a Python package on the Python Package Index (<https://pypi.org/project/masspy/>). All required external dependencies integrated and utilized by MASSpy are also available on the Python Package Index (<https://pypi.org/>) and are licensed under their respective licensing terms. Both the Gurobi Optimizer (Gurobi Optimization, Houston, TX) and the IBM CPLEX Optimizer (IBM, Armonk, NY) are

Abstract

Mathematical models of metabolic networks utilize simulation to study system-level mechanisms and functions. Various approaches have been used to model the steady state behavior of metabolic networks using genome-scale reconstructions, but formulating dynamic models from such reconstructions continues to be a key challenge. Here, we present the Mass Action Stoichiometric Simulation Python (MASSpy) package, an open-source computational framework for dynamic modeling of metabolism. MASSpy utilizes mass action kinetics and detailed chemical mechanisms to build dynamic models of complex biological processes. MASSpy adds dynamic modeling tools to the COBRAPy package to provide a unified framework for constraint-based and kinetic modeling of metabolic networks. MASSpy supports high-performance dynamic simulation through its implementation of libRoadRunner: the Systems Biology Markup Language (SBML) simulation engine. Three examples are provided to demonstrate how to use MASSpy: (1) a validation of the MASSpy modeling tool through dynamic simulation of detailed mechanisms of enzyme regulation; (2) a feature demonstration using a workflow for generating ensemble of kinetic models using Monte Carlo sampling to approximate missing numerical values of parameters and to quantify biological uncertainty, and (3) a case study in which MASSpy is utilized to overcome issues that arise when integrating experimental data with the computation of functional states of detailed biological mechanisms. MASSpy represents a powerful tool to address challenges that arise in dynamic modeling of metabolic networks, both at small and large scales.

Author summary

Genome-scale reconstructions of metabolism appeared shortly after the first genome sequences became available. Constraint-based models are widely used to compute steady state properties of such reconstructions, but the attainment of dynamic models has remained elusive. We thus developed the MASSpy software package, a framework that

freely available for academic use, with solvers and installation instructions found at their respective websites. Images for Docker containers that include the MASSpy software can be downloaded online from the DockerHub Registry (<https://hub.docker.com/r/sbrg/masspy>), or they can be built locally to include the licensed commercial optimization solvers. Instructions for MASSpy installation, including instructions for using MASSpy with Docker, are found in the repository README or in the documentation (<https://masspy.readthedocs.io> and [S2 File](#)). The data, scripts, and instructions needed to reproduce results of the presented examples are also available on GitHub (<https://github.com/SBRG/MASSpy-publication>) and in the supplement ([S3 File](#)).

Funding: Funding for this work and support for ZBH, DZ, and BOP was provided by the Novo Nordisk Foundation through the Center for Biosustainability (<https://www.biosustain.dtu.dk>) at the Technical University of Denmark [NNF10CC1016517]. YK and JTY were supported by the Institute for Systems Biology's Translational Research Fellows Program (<https://isbscience.org/training/trp/translational-research-fellows-program/>). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

enables the construction, simulation, and visualization of dynamic metabolic models. MASSpy is based on the mass action kinetics for each elementary step in an enzymatic reaction mechanism. MASSpy seamlessly unites existing software packages within its framework to provide the user with various modeling tools in one package. MASSpy integrates community standards to facilitate the exchange of models, giving modelers the freedom to use the software for different aspects of their own modeling workflows. Furthermore, MASSpy contains methods for generating and simulating ensembles of models, and for explicitly accounting for biological uncertainty. MASSpy has already demonstrated success in a classroom setting. We anticipate that the suite of modeling tools incorporated into MASSpy will enhance the ability of the modeling community to construct and interrogate complex dynamic models of metabolism.

This is a *PLOS Computational Biology* Software paper.

Introduction

The availability of genome sequences and omic data sets has led to significant advances in metabolic modeling at the genome scale, resulting in the rapid expansion of available genome-scale metabolic reconstructions [1]. The attainability of new data has also led to the generation of new metabolic modeling software tools that can process these data [2–4]. One of the most broadly used metabolic modeling software suites, COntstraint-Based Reconstruction and Analysis (COBRA) [5], provides a scalable framework that is invaluable for the contextualization and analysis of multi-omic data, as well as for understanding, predicting, and engineering metabolism [6–16]. While several methods have been developed that allow COBRA models to integrate certain data types to model long timescale dynamics [17–19], COBRA models are inherently limited by the flux-balance assumption.

Kinetic modeling methods use detailed mechanistic information to model dynamic states of a network [20]. The inclusion of multiple detailed enzymatic mechanisms presents challenges in formulating and parameterizing stable large-scale kinetic models. Further, additional issues arise when integrating incomplete experimental data into metabolic reconstructions, necessitating the need for approximation methods to gap fill missing values that satisfy the thermodynamic constraints imposed by the system [21, 22].

Various efforts have been made to bridge the gap between constraint-based and kinetic modeling methods in order to address the challenges associated with dynamic modeling [21–24]. One such methodology is the Mass Action Stoichiometric Simulation (MASS) approach, in which mass action kinetics are used to construct condition-specific dynamic models [24–28]. The MASS modeling approach provides an algorithmic, data-driven workflow for generating *in vivo* kinetic models in a scalable fashion [29]. The MASS methodology can be used in tandem with COBRA methods for both steady-state and dynamic analyses of a metabolic reconstruction in a single workflow. MASS models can incorporate the stoichiometric description of enzyme kinetic mechanisms and have been used to explicitly compute fractional states of enzymes, providing insight into regulation mechanisms at a network-level [26, 30]. The MASS modeling framework has been implemented in the MASS Toolbox [31], but is limited by its reliance on a commercial software platform (Mathematica).

Here, we detail the Mass Action Stoichiometric Simulation Python (MASSpy) package, a versatile computational framework for dynamic modeling of metabolism. MASSpy expands the modeling framework of the COBRApy [32] package by integrating dynamic simulation and analysis tools to facilitate dynamic modeling. Further, MASSpy contains various algorithms designed to address and overcome the issues that arise when incorporating experimental data and biological variation into dynamic models with detailed mechanistic information. By addressing the issues associated with integrating physiological measurements and biological mechanisms in dynamic modeling approaches, we anticipate that MASSpy will become a powerful modeling tool for modeling dynamic behavior in metabolic networks. The source code for MASSpy is freely available on GitHub (<https://github.com/SBRG/MASSpy>), and the documentation—along with tutorials—can be found at ReadTheDocs. (<https://masspy.readthedocs.io>).

Design and implementation

Developing in Python

The MASSpy software package (S1 File) is written entirely in Python 3, an interpreted object-oriented high-level programming language that has become widely adopted in the scientific community due to its clean syntax with straightforward semantics that make it intuitive to learn [33] and its unique features for high-level scientific computing (e.g., a flexible interface to compiled languages such as C++ [34]). The open-source nature of Python avoids the inherent limitations associated with costly commercial software [35]. Consequently, developing in Python provides access to a growing variety of open-source scientific software libraries, including key data science packages of the SciPy ecosystem [33, 36]. MASSpy relies extensively on these packages for numerical computation (NumPy [37]), symbolic mathematics (SymPy [38]), high-performance data structures that easily transfer and export content across packages (Pandas [39]), and data visualization (Matplotlib [40]). IPython (Jupyter) notebooks [41, 42] provide an interactive Python environment for developing and executing various applications. Furthermore, containerization of applications through Docker [43] facilitates their exchange and deployment using portable containers bundled with standardized computational environments. See Table 1 for a list of scientific packages integrated into the MASSpy package and how they are utilized for various purposes within the MASSpy framework.

Building on the COBRApy framework

To facilitate the integration of constraint-based and dynamic modeling frameworks, MASSpy utilizes the COBRApy package [32] as a foundation to build upon and extend in order to support dynamic simulation and analysis capabilities. MASSpy derives several benefits from building on the COBRApy framework, including exploiting the direct inclusion of various COBRA methods already implemented in Python. The inclusion of COBRA methods is made simple using Python inheritance behavior; the three core COBRApy classes (Metabolite, Reaction, and Model) serve as the base classes for three core MASSpy classes (MassMetabolite, MassReaction, and MassModel) as described in the MASSpy documentation (<https://masspy.readthedocs.io> and S2 File). Inheritance behavior maintains the functionality of the COBRA methods for gene deletion studies by preserving the Boolean representation of Gene-Protein-Reaction (GPR) rules that indicate whether the genes essential for an enzyme-catalyzed reaction are expressed in the organism [49]. Consequently, all methods for COBRApy objects readily accept the analogous MASSpy objects as valid input, preserving the commands and conventions familiar to current COBRApy users. COBRApy is a popular software platform preferred by many in the COBRA community [5]; therefore, preserving COBRApy

Table 1. Overview of select scientific software with relevance to MASSpy functionality.

| Package | Relevance to MASSpy functionality | Reference |
|--------------------------------|--|----------------------------------|
| COBRAPy | Python package for reconstruction and simulation of genome-scale flux states. Utilized as a framework for MASSpy and enables a variety of COBRA methods. | [32] |
| Escher | A Python interface to the Escher visualization package. Used to build, share, and embed visualizations of metabolic pathway and node maps. | [44] |
| libRoadRunner | High-performance and portable SBML simulation engine for systems biology. Provides dynamic simulation and NLEQ steady state determination methods for SBML compatible models. | [45] |
| libSBML | A Python interface for reading and writing models in SBML. | [46] |
| Matplotlib | Fundamental package for visualization in Python. Provides comprehensive plotting tools for visualizing observations, including dynamic simulation results and phase portraits | [40] |
| NumPy | Fundamental Python package for numerical computation and data science. Provides efficient array/matrix data types, operations, and random number generation when sampling. | [37] |
| Optlang | Formulation of optimization problems using symbolic expressions and native Python algebra syntax. Provides a common Python interface for various optimization solver backends. | [47] |
| Pandas | High-performance data structures and analysis tools for data science in Python. | [39] |
| SciPy | A collection of scientific algorithms in Python. Primarily used for interpolation of dynamic simulation results and linear algebra operations | [36] |
| SymPy | Generation and manipulation of symbolic mathematical expressions in Python. Primarily used to formulate ordinary differential equations, rate laws, and optimization problems. | [38] |
| swiglpk | A Python interface to the GNU Linear Programming Kit used for optimization. Utilized by Optlang to provide LP and MILP support. | [48] |
| cplex ^a | A Python interface to the IBM CPLEX Optimizer used for optimization. Utilized by Optlang to provide LP, MILP, and QP support. | IBM, Armonk, NY |
| gurobipy ^a | A Python interface to the Gurobi Optimizer used for optimization. Utilized by Optlang to provide LP, MILP, and QP support. | Gurobi Optimization, Houston, TX |
| iPython (Jupyter) ^b | Interactive computational environment for the development and exchange of scientific workflows. Utilized for interactive computation, interactive visualization, and documentation. | [41, 42] |
| Docker ^b | Containerize and deploy applications in standardized, portable environments. Facilitates the reliable creation, exchange and deployment of containers packaged with MASSpy and other related applications. | Docker Inc., Palo Alto, CA [43] |

Abbreviations: Linear Programming (LP); Mixed Integer Linear Programming (MILP); Quadratic Programming (QP); Systems Biology Markup Language (SBML).

^aCommercial optimization solvers with Python APIs and free academic licenses.

^bUtilized for code development and distribution purposes.

<https://doi.org/10.1371/journal.pcbi.1008208.t001>

conventions aids in the adoption of MASSpy among those users. Inheriting from the COBRAPy classes additionally allows for easy conversion between COBRAPy and MASSpy objects without loss of relevant biochemical and numerical information. These two features of Python inheritance are critical in maintaining functionality for COBRAPy implementations of various flux-balance analysis (FBA) algorithms in MASSpy.

Adding dynamic simulation capabilities

The creation and simulation of dynamic models requires deriving a set of ordinary differential equations (ODEs) from the stoichiometry of a reconstructed network and assigning kinetic rate laws to each reaction in the network [21]. MASSpy utilizes SymPy [38] to represent reaction rates and differential equations as symbolic expressions. All MassReaction objects use mass action kinetics to automatically generate a default rate law; however, user-defined rate laws can be assigned to replace mass action rate laws with alternative equations, such as those derived using Michaelis-Menten kinetics. All MassMetabolite objects generate their associated differential equation by combining the rates of reactions in which they participate and contain the initial conditions necessary to solve the system of ODEs.

To solve the system of ODEs, the MASSpy Simulation class employs libRoadRunner [45], a high-performance Systems Biology Markup Language (SBML) [50] simulation engine that is capable of supporting most SBML Level 3 specifications. The libRoadRunner utilizes a Just-In-Time (JIT) compiler with an LLVM JIT compiler framework to compile SBML-specified models into machine code, making the libRoadRunner simulation engine appropriate for solving large models effectively. Although libRoadRunner has a large suite of capabilities, it is currently used for two purposes in MASSpy: the steady-state determination via NLEQ1 and NLEQ2 global newton methods [51], and dynamic simulation via integration of ODEs through deterministic integrators, including CVODE solver from the Sundials suite [52]. Because libRoadRunner requires models to be in SBML format, the Simulation object exports models into SBML format before compiling them into machine code via libRoadRunner.

Model import, export, and network visualization

MASSpy utilizes two primary formats for the import and export of models: SBML format and JavaScript Object Notation (JSON). MASSpy currently supports SBML L3 core specifications [53] along with the FBC [54] and Groups [55] packages, providing support for both constraint-based and dynamic modeling formats. Although SBML is necessary to utilize libRoadRunner, there are a number of additional benefits obtained by supporting SBML. In addition to being a standard format among the general systems biology community [50], SBML is a widely used model format specifically among members of the COBRA modeling community [5].

MASSpy also provides support for importing and exporting models via JSON, a text-based syntax that is useful for exchanging structured data between programming languages [56]. The MASSpy JSON schema is designed for interoperability with Escher [44], a pathway visualization tool designed to visualize various multi-omic data sets mapped onto COBRA models. The interoperability with Escher is exploited by MASSpy to provide various pathway and node map visualization capabilities.

Mechanistic modeling of enzyme regulation

The reconstruction of all microscopic steps performed by an enzyme (an “enzyme module”) represents the full stoichiometric description of an enzyme using mass action kinetics [27]. MASSpy facilitates the construction of enzyme modules through the EnzymeModule, EnzymeModuleForm, and EnzymeModuleReaction classes, which inherit from the MassModel, MassMetabolite, and MassReaction classes, respectively. The EnzymeModule class contains methods and attribute fields to aid in the construction of EnzymeModules based on the steps outlined for constructing enzyme modules in Du et al. [27]. Given the number and complexity of possible enzymatic mechanisms [57], MASSpy also provides the ability to group relevant objects into different user-defined categories, such as active/inactive states and different enzyme complexes. The EnzymeModuleDict objects are used to represent enzyme modules once merged into a larger model, preserving user-defined categories and other information relevant to the construction of the EnzymeModule, such as total enzyme concentration. More details can be found in the MASSpy documentation (S2 File).

Ensemble sampling, assembly, and modeling

Ensemble approaches are used to address various issues concerning parameter uncertainty and experimental error in metabolic models [22]. Ensemble modeling refers to the assembly of dynamic models with similar mechanistic formulations that span the feasible kinetic solution space. Ensemble modeling allows for the range of possible network phenotypes to be explored,

proving to be a valuable tool when parameterization is incomplete or unknown, as is often the case with kinetic models. MASSpy enables ensemble modeling approaches through the use of Markov chain Monte Carlo (MCMC) sampling of fluxes and concentrations [58, 59]. The flux sampling capabilities can be derived from the COBRAPy package and employ two different hit-and-run sampling methods: one with a low memory footprint [60] and another with multiprocessing support [61]. To sample metabolite concentrations, MASSpy employs a ConcSolver object to populate the optimization solver with constraints for thermodynamically feasible concentration ranges [28, 62, 63] and two hit-and-run sampling methods for concentrations were implemented in MASSpy with algorithms analogous to those for flux sampling. MASSpy provides several built-in methods for ensemble generation from sampling data. Once generated, the ensemble of models can be loaded into the MASSpy Simulation object, simulated, and visualized using built-in ensemble visualization and analysis methods. Additional details can be found in the MASSpy documentation (S2 File).

Results

We have conducted a validation, a feature demonstration, and a case study that exemplify how MASSpy features combine to facilitate the dynamic modeling of metabolism (Fig 1). We validated MASSpy as a modeling tool by describing mechanisms of enzyme regulation using enzymes modules to replicate select results presented in Yurkovich et al. [24]. A variety of features are demonstrated through a workflow in which an ensemble of stable kinetic models is generated through MCMC sampling to examine biological variability while satisfying the thermodynamic constraints imposed by the network. In the case study, we integrated COBRA and MASS modeling methodologies to create a kinetic model of *E. coli* glycolysis from a metabolic reconstruction, providing novel insight into functional states of the proteome and activities of different isozymes. See Table 2 for a comparison of explicitly supported MASSpy features with those of other dynamic modeling tools.

Validation as a modeling tool through enzyme regulation in MASS models

Here, we demonstrated MASSpy as a modeling tool and the MASSpy implementation of enzyme modules by replicating the results produced by Yurkovich et al. [24]. The authors used the MASS Toolbox [31] to elucidate the systems-level effects of allosteric regulation in the glycolytic pathway of red blood cells (RBCs). Enzyme modules for hexokinase, phosphofructokinase, and pyruvate kinase were reconstructed using the mechanistic formulations of ligand-binding events [26, 27] and merged with a MASS model of RBC metabolism comprised of the glycolytic pathway, the Rapoport-Luebering (RL) shunt, and hemoglobin binding events for oxygen and 2,3-diphosphoglycerate [29]. Enzyme modules were merged into the glycolytic pathway in different combinations, and dynamic simulations were subsequently performed to characterize the interactions of the three allosterically regulated kinases when responding to disturbances in ATP utilization.

To validate MASSpy as a modeling tool, we used MASSpy to replicate key observations produced by the MASS Toolbox as presented in Yurkovich et al. [24]. Using MASSpy, we reconstructed the RBC metabolic model and the enzyme modules for the three key regulatory kinases as previously described. We integrated the reconstructed enzyme modules into the glycolytic model to introduce varying levels of regulation. Because enzyme modules were constructed and parameterized for the steady-state conditions of the MASS model, addition of an enzyme module to a MASS model was a straightforward and scalable process. The overall reaction representation for the enzyme in the MASS model was removed and replaced with the set of reactions that comprise the microscopic steps of the enzyme module (Fig 2).

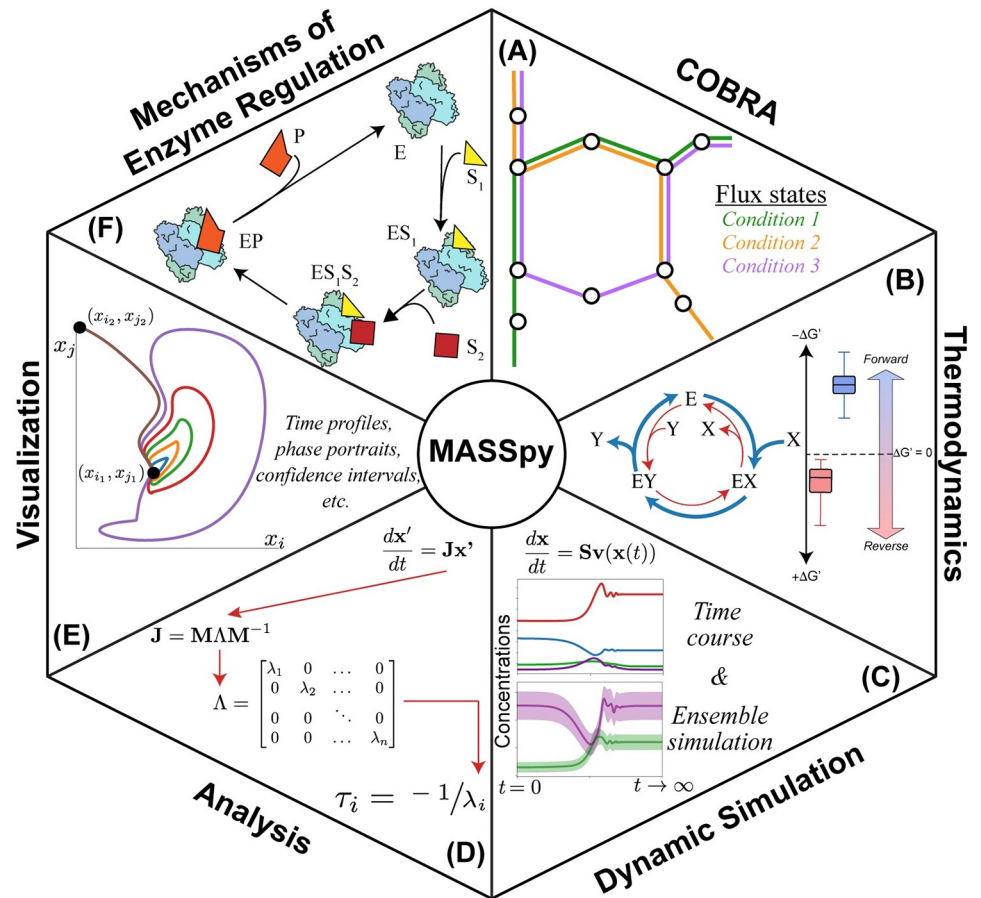


Fig 1. Overview of MASSpy features. (A) MASSpy expands COBRApy to provide constraint-based methods for obtaining flux states. (B) Thermodynamic principles are utilized by MASSpy to sample concentration solution spaces and to evaluate how thermodynamic driving forces shift under different metabolic conditions. (C) MASSpy enables dynamic simulation of models to characterize transient dynamic behavior and contains ensemble modeling methods to represent biological uncertainty. (D) Network properties such as relevant timescales and system stability are characterized by MASSpy using various linear algebra and analytical methods. (E) MASSpy contains built-in functions that enable the visualization of dynamic simulation results. (F) Mechanisms of enzymatic regulation are explicitly modeled in MASSpy through enzyme modules, enabling computation of catalytic activities and functional states of enzymes.

<https://doi.org/10.1371/journal.pcbi.1008208.g001>

We performed dynamic simulations, subject to physiologically relevant perturbations, to provide a fine-grained view of the concentration and flux solution profiles for individual enzyme signals and qualitatively represent the systemic effects of additional regulatory mechanisms. We then used MASSpy visualization methods to replicate key results [24] (S3 File). Through this case study, we have demonstrated how enzyme modules were constructed from enzymatic mechanisms in MASSpy, and we validated MASSpy as a dynamic modeling tool by exploring previously reported systems-level effects of regulation [24]. See S3 File for all data and scripts associated with the validation, including kinetic parameters for all three enzyme modules.

Demonstration of features through ensemble sampling, assembly, and modeling

Many ensemble modeling approaches utilize sampling methods to approximate missing values and quantify uncertainty in metabolic models [21, 22, 28, 59, 64, 65]. To demonstrate the

Table 2. Comparison of explicit helper methods for selected dynamic modeling tools.

| Software | | MASSpy (COBRApy) | MASS Toolbox | Tellurium | PySB | PySCeS (CBMPy) | COPASI |
|-------------------------------------|-----------------------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---|
| Version | | 0.1.1 (0.19.0) | 1.2.0 | 2.1.6 | 1.11.0 | 0.9.7 (0.7.25) | 4.29.228 |
| Environment | | Python 3.6+ | Mathematica 9.0+ | Python 2.7, 3.4+ | Python 2.7, 3.6+ | Python 2.7, 3.5+ | Bindings: C#, Java, Python 2.7, 3.6+ |
| Model Construction | Model merging | yes | yes | yes | yes | | yes |
| | Automated rate law construction | yes | yes | yes | yes | | yes |
| | Enzyme modules | yes | yes | | yes | | |
| | Symbolic expression manipulation | yes | yes | yes | yes | | |
| | Boolean GPR associations | yes | yes | | | yes | |
| Scanning and Sampling (MCMC) | Flux sampling | yes | yes | | | | yes |
| | Concentration sampling | yes | | | | | yes |
| | Parameter sampling | yes | | | | | yes |
| | Parameter scan | | | yes | yes | yes | yes |
| Simulation | ODE | yes | yes | yes | yes | yes | yes |
| | Stochastic | | | yes | yes | | yes |
| | Optimization | yes | yes | | | yes | yes |
| | Ensemble | yes | | yes | | yes | yes |
| Analysis | Steady state | yes | yes | yes | yes | yes | yes |
| | Stoichiometric | yes | yes | yes | | yes | yes |
| | Thermodynamic | yes | | | | | |
| | Sensitivity | | | yes | yes | yes | yes |
| | Metabolic Control | | | yes | | yes | yes |
| | Stability | yes | yes | yes | | yes | yes |
| | Flux balance / flux variability | yes | yes | | | yes | |
| | Gene essentiality / gene deletion | yes | yes | | | yes | |
| | | | | | | | |
| Visualization | Time profiles | yes | yes | yes | | yes | yes |
| | Phase portraits | yes | yes | | | yes | yes |
| | Pathway maps | | yes | yes | yes | | yes |
| QC/QA | Elemental balancing | yes | yes | | | | |
| | Thermodynamic feasibility | yes | yes | | | | |
| | Units | yes | yes | | | | yes |
| Standards | SBML | yes | yes | yes | yes | yes | yes |
| | SED-ML | | | yes | | yes | yes |
| | COMBINE | | | yes | | yes | yes |

Explicitly supported features (via “helper methods” or “helper classes”) are those that are directly provided by the tool or software package itself and do not include those that could be piped in using third-party packages or tools (i.e., directly calling a function from the package/tool as opposed to using other Python packages to achieve the goal).

<https://doi.org/10.1371/journal.pcbi.1008208.t002>

sampling and ensemble handling capabilities of MASSpy, we utilized MCMC sampling with an ensemble modeling approach to assess the dynamics for a range of pyruvate kinase enzyme modules (Fig 3). Using RBC glycolysis and hemoglobin as the reference model [24], we used MCMC sampling to generate 15 candidate flux states and 15 candidate thermodynamically

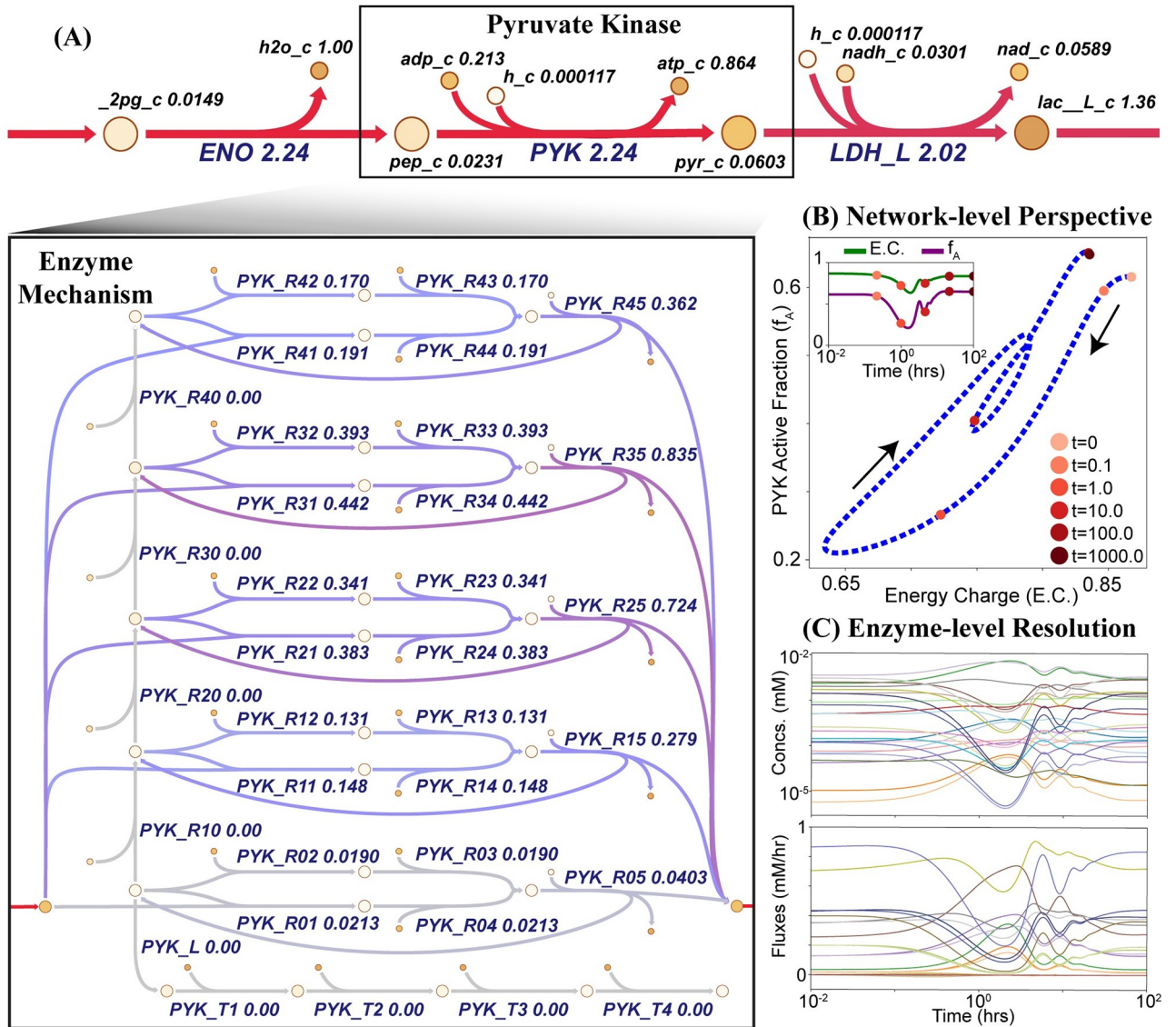


Fig 2. Enzyme modules are explicit representations of enzymatic regulatory mechanisms. (A) The reaction catalyzed by pyruvate kinase is replaced with the stoichiometric description of the enzymatic mechanism. The steady state values obtained after simulating a 50% increase of ATP utilization are mapped onto a metabolic pathway map drawn using Escher [44]. The colors represent flux values and range from red to purple to gray, with red indicating higher flux values and gray indicating lower flux values. (B) Enzyme modules provide a network-level perspective of regulation mechanisms by plotting systemic quantities against fractional states of enzymes as described in Yurkovich et al. [24]. (C) The different signals of the enzyme module can be observed to provide enzyme-level resolution of the regulatory response.

<https://doi.org/10.1371/journal.pcbi.1008208.g002>

feasible concentration states, allowing for variables to deviate from their reference state by up to 80 percent. This procedure resulted in 225 candidate models that represent all possible combinations of flux and concentration states. We calculated pseudo-elementary rate constants (PERCs) and simulated models to steady state. We simulated a 50% increase in ATP utilization to mimic a physiologically relevant disturbance, such as increased shear stress due to arterial constriction [66]. Out of the 225 models, 10 models were discarded due to instability, determined by an inability to reach a steady state through simulation.

We then reconstructed enzyme modules for pyruvate kinase [24] for the remaining 215 models. Numerical values of rate constants for each enzyme were determined using the SciPy

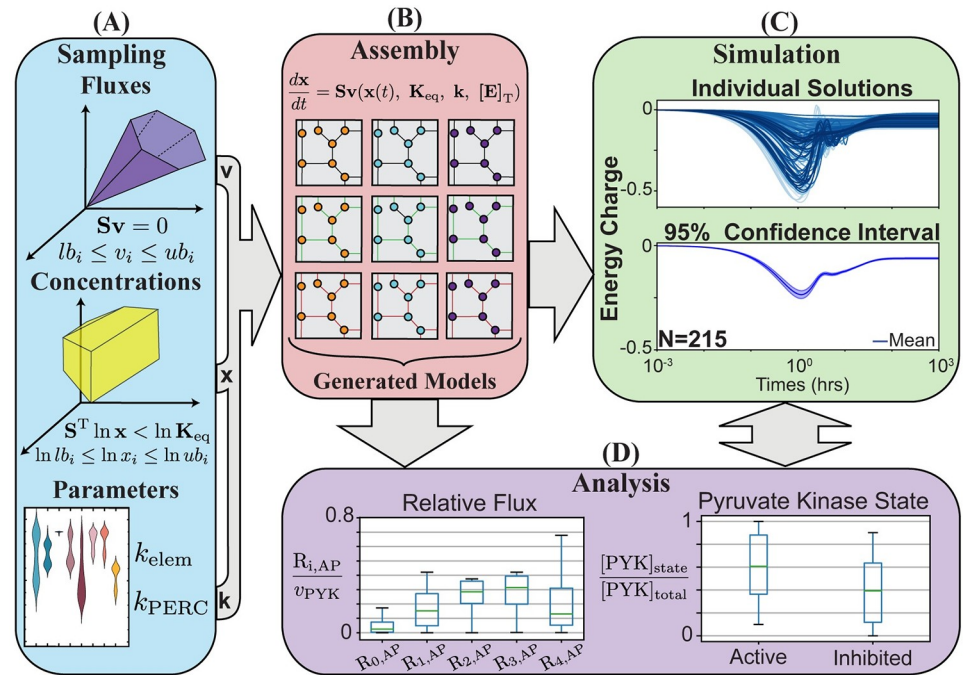


Fig 3. A MASSpy workflow for ensemble creation and modeling using MCMC sampling. A typical ensemble modeling workflow using MASSpy to generate and assemble an ensemble of stable kinetic models for dynamic simulation and analysis. (A) The solution spaces for fluxes and concentrations are sampled using MCMC sampling to generate data for candidate model states. Rate constants are obtained through parameter fitting for elementary rate constants and computation of PERCs. (B) Sampling data is integrated into the candidate models, and models are subsequently filtered based on their stability to assemble the ensemble of stable dynamic models. Once assembled, the ensemble is used to study biological variability in the network of interest through (C) dynamic simulation and (D) analysis.

<https://doi.org/10.1371/journal.pcbi.1008208.g003>

implementation of a trust-region method for nonlinear convex optimization [67]. Without knowledge of physiological constraints, the numerical solutions for rate constants produced by optimization routine were not guaranteed to be physiologically feasible. Therefore, we integrated these enzyme modules into their MASS models and simulated with and without the ATP utilization increase to ensure they had stable steady states before and after the perturbation. All remaining 215 models were able to successfully reach stable steady states and subsequently assembled into the ensemble for dynamic simulation and analysis.

The time-course results for the ensemble energy charge deviation were plotted with a 95% confidence interval. From these results, it can be seen that the mean energy charge decreased at most about 25% from its baseline value (Fig 3C). Steady state analysis of the pyruvate kinase enzyme modules after the disturbance revealed a tendency for the enzyme to remain in an active state, with a median value of approximately 61%. The differences in candidate flux and concentration states resulted in an interquartile range between 36-88% of total pyruvate kinase, with all variations of pyruvate kinase in the ensemble maintaining at least 10% activity. Furthermore, examination of the relative flux load through the $R_{i,AP}$ forms showed that most of the flux load was carried by the $R_{2,AP}$ and $R_{3,AP}$ reaction steps while a minuscule fraction was carried by the $R_{0,AP}$ reaction step, regardless of variation. However, the variations had an effect on whether $R_{2,AP}$ carried more flux than $R_{3,AP}$, and whether the remaining flux was predominantly distributed through the $R_{1,AP}$ or the $R_{4,AP}$ reaction step. Through this demonstration, we have demonstrated how MASSpy sampling facilitated the assembly and simulation of

an ensemble to characterize the dynamic response of a key regulatory enzyme and quantify its functional states after a physiologically relevant disturbance. See [S3 File](#) for all data and scripts associated with the ensemble modeling demonstration.

Case study: Computing functional states of the *E. coli* proteome

Here, we illustrated unique features of MASSpy in a workflow to compute the functional states of the proteome, providing insight into distribution of catalytic activities of enzymes for different metabolic states. We utilized COBRA and MASS modeling methodologies to incorporate omics data into a metabolic reconstruction of *E. coli*, formulating a kinetic model containing all microscopic steps for each enzymatic reaction mechanism of the glycolytic subnetwork. Once formulated, we interrogated the model to examine the shift in thermodynamic driving force for *E. coli* on different carbon sources and to compare the activities of different isozymes, exemplifying the utility of MASSpy.

To construct a kinetic model of the glycolytic subnetwork, we integrated steady-state data for growth on glucose and pyruvate carbon sources from Gerosa et al. [68] into the iML1515 genome-scale metabolic reconstruction of *E. coli* K-12 MG1655 [69]. For each carbon source, we fixed the growth rate for iML1515 and performed FBA using a quadratic programming objective to compute a flux state that minimized the error between known and computed fluxes. For the irreversible enzyme pairs of phosphofructokinase/fructose 1,6-bisphosphatase (PFK/FBP) and pyruvate kinase/phosphoenolpyruvate synthase (PYK/PPS), individual flux measurements were each increased by 10% of the net flux for the enzyme pair without changing the overall net flux value to ensure presence of the enzyme as seen in proteomic data [70].

Once flux states were obtained for each carbon source, the glycolytic subnetwork was extracted from iML1515. Flux units were converted from grams of cellular dry weight into molar units using volumetric measurements of *E. coli* obtained from Volkmer and Heinemann [71], and equilibrium constants obtained from eQuilibrator [72] through component contribution [73] were set for each reaction. Concentration growth data from Gerosa et al. [68] was integrated into the model and minimally adjusted for thermodynamic feasibility; for metabolites missing concentration data, an initial value of 0.001 M was provided before adjustments through sampling. Concentrations were sampled within an order of magnitude of their current value to produce an ensemble of 100 candidate models for each growth condition. Metabolite sinks were added to the model to account for metabolite exchanges between the modeled subnetwork and the global metabolic network outside of the scope of the model. The relevant growth data for each carbon source, volumetric measurements, and equilibrium constants are found in the [S1 Data](#).

For each model in the ensemble, enzyme modules were constructed for each reaction using a nonlinear parameter fitting package (<https://github.com/opencobra/MASSEf>) and kinetic data extracted from the literature. Additional isozymes of phosphofructokinase (PFK), fructose 1,6-bisphosphatase (FBP), fructose 1,6-bisphosphate aldolase (FBA), phosphoglycerate mutase (PGM), and pyruvate kinase (PYK) were also constructed, bringing the total amount of enzyme modules to 17. Fluxes through individual isozymes were set by splitting the steady state flux between the major and minor isozyme forms. After integrating all enzyme modules into the network, each model was simulated to steady state and exported for analysis.

The Gibbs free energy for each enzyme-catalyzed reaction and fractional abundance of each enzyme form were calculated. Sensitivity analysis of the flux split between the isozyme forms revealed that the Gibbs free energy and fractional abundance of enzyme forms did not show significant variation for either carbon source ([S1 Fig](#)); therefore, remaining analyses were done with 75% and 25% of the flux assigned to major and minor isozyme forms, respectively.

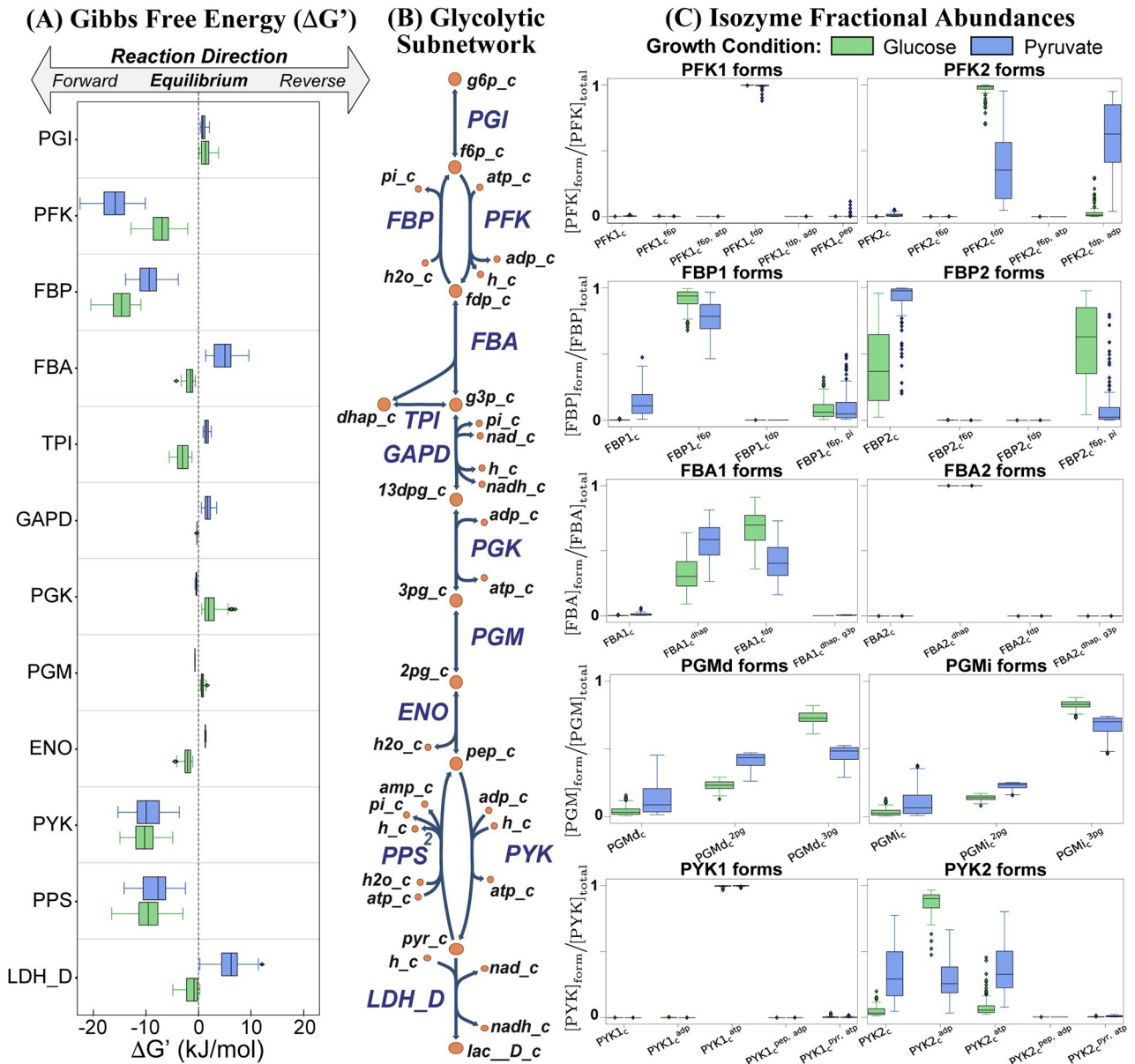


Fig 4. Comparison of free energy and isozyme fractional abundances for carbon sources. (A) The Gibbs free energy represents the thermodynamic driving force, shifting the metabolic state depending on the carbon source. (B) The glycolytic subnetwork extracted from *E. coli* iML1515 consists of 12 reactions represented by the 17 enzyme modules. (C) The fractional abundance for each enzyme form can be computed and compared for the different isozyme pairs, providing insight into how the catalytic activity is distributed across the isozymes in glucose and pyruvate growth conditions. The fractional abundances for all enzymes can be found in the supplement (S2 Fig).

<https://doi.org/10.1371/journal.pcbi.1008208.g004>

Comparison of the glucose and pyruvate growth conditions revealed that the free energy of the reversible reactions remained close to equilibrium, changing from one metabolic state to another as the thermodynamic driving force shifts according to the carbon source, as seen in reversible reactions triose phosphate isomerase (TPI), glucose 6-phosphate dehydrogenase (GAPD), phosphoglycerate kinase (PGK), phosphoglycerate mutase (PGM), and enolase (ENO) (Fig 4A). The reaction pairs, PFK/FBP and PYK/PPS, maintained their flux directions to form a futile cycle across conditions.

Steady state analysis of the isozyme fractional abundances elucidated a preference for a specific enzyme state conserved among growth conditions for PFK1, FBA2, and PYK1 (Fig 4C). Steady state analysis also revealed that the isozyme pairs of PFK, FBP, and PYK primarily existed in their product-bound form, a reflection of the metabolite concentration levels found in S2 Data. Specifically, the relatively high concentrations of fructose 1,6-diphosphate (FDP) observed for glucose growth conditions and of adenosine triphosphate (ATP) observed for pyruvate growth conditions contributed to significant differences between enzyme product and reactant concentration levels, creating the conditions favorable to the product-bound enzyme forms. Both the major and minor PGM isozymes have a similar distribution in their enzyme forms. The fractional abundance of all enzyme states in the glycolytic subnetwork can be found in S2 Fig. Through this case study, we have demonstrated that MASSpy can be used to gain insight into the distribution of functional states for the glycolytic proteome in *E. coli* without prior knowledge of enzyme concentrations. See S3 File for all data and scripts associated with the case study, including microscopic steps and kinetic parameters for all enzyme modules.

Discussion

We describe MASSpy, a free and open-source software implementation for dynamic modeling of biological systems. MASSpy expands the COBRApy framework, leveraging existing methods familiar to COBRA users combined with kinetic modeling methods to form a single, intuitive framework for constructing and interrogating dynamic models. In addition to enabling dynamic simulation, MASSpy contains tools for facilitating the reconstruction and analysis of enzyme modules, MCMC sampling and ensemble modeling capabilities, interfacing with packages for pathway visualization (Escher [44]), and exchanging models in SBML format (libSBML [46]). Taken together, the presentation of the MASSpy software package has several important implications for practitioners of dynamic metabolic simulation.

MASSpy provides several benefits over existing modeling packages (Table 2). While MASS models provide an algorithmic approach for generating dynamic models that has already proven useful in several metabolic studies [24, 26–29], a formal implementation of the MASS framework has only existed on a commercial software platform (Mathematica). MASSpy primarily utilizes the MASS approach and therefore integrates a suite of tools into its framework for addressing issues specific to MASS modeling. Unlike other packages for traditional kinetic modeling, MASSpy incorporates both COBRA methods and MCMC sampling methods for estimating missing values for several data types. MASSpy's seamless integration with COBRApy offers a vast array of constraint-based and dynamic modeling tools within a single open-source framework. Although alternative python-based packages for constraint-based modeling exist (e.g., PyFBA [2]), the COBRApy package is the most widely used Python package for constraint-based modeling by the COBRA community [5]. Furthermore, MASSpy contains unique capabilities to facilitate the construction and analysis of detailed enzyme modules (i.e., microscopic steps), which allow for the dynamics of transient responses to be observed in situations in which the quasi-steady state and quasi-equilibrium assumptions cannot be applied. By directly expanding the COBRApy framework for MASSpy, current COBRApy users will find that MASSpy provides procedures and protocols that they may be familiar with, and allows members of the COBRA community to directly integrate new tools into their existing workflows.

The open source nature of Python also enables users to utilize their preferred tools while integrating MASSpy capabilities, creating workflows suited to their specific needs. For example, users familiar with PyBioNetFit [74], PyDREAM [75], or PyMC3 [76] can implement

these packages for parameter fitting and optimization routines, then update their MASSpy models with the resulting parameter sets. Through manipulation of data via Pandas [39], a wide variety of additional interactive statistical visualizations are available through the Altair visualization package [77], while dynamic network visualizations in iPython notebooks [41, 42] can be accomplished through PyVIPR [78]. By capitalizing on the availability of other systems biology and data science tools in the Python ecosystem, users maintain the freedom to use their preferred tools and gain flexibility to determine how MASSpy is best utilized for their particular needs.

MASSpy is primarily built for deterministic simulations of a metabolic model and thus may face limitations for other uses. For example, a package like PySCeS [79] could be utilized to perform stochastic simulations. Users who often analyze sensitivity may prefer Tellurium and its implementation of libRoadRunner [45, 80] for explicit support of metabolic control analysis (MCA) workflows; however, MASSpy does contain similar MCA methods through its own implementation of libRoadRunner. Other dynamic modeling packages offer certain features not available in MASSpy, such as a graphical user interface (COPASI [81]) or a rule-based modeling approach (PySB [82]): see Table 2 for a comparison of MASSpy's software features with other existing dynamic modeling packages. MASSpy's use of SBML facilitates the transfer of models to other software environments, if desired [53].

Taken together, we have described MASSpy, a Python-based software package for the reconstruction, simulation, and visualization of dynamic metabolic models. MASSpy provides a suite of dynamic modeling tools while leveraging existing implementations of constraint-based modeling tools within a single, unified framework. The case studies presented here validate MASSpy as a modeling tool and demonstrate how the combination of constraint-based and kinetic modeling features support data-driven solutions for various dynamic modeling applications. We anticipate that the community will find MASSpy to be a useful tool for dynamic modeling of metabolism.

Availability and future directions

Software availability and requirements

The source code for MASSpy is available online (<https://github.com/SBRG/MASSpy>) and in the S1 File under an MIT license. MASSpy is also hosted as a Python package on the Python Package Index (<https://pypi.org/project/masspy/>). All required external dependencies integrated and utilized by MASSpy are also available on the Python Package Index (<https://pypi.org/>) and are licensed under their respective licensing terms. Both the Gurobi Optimizer (Gurobi Optimization, Houston, TX) and the IBM CPLEX Optimizer (IBM, Armonk, NY) are freely available for academic use, with solvers and installation instructions found at their respective websites. Images for Docker containers [43] that include the MASSpy software can be downloaded online from the DockerHub Registry (<https://hub.docker.com/r/sbrg/masspy>), or they can be built locally to include the licensed commercial optimization solvers. Instructions for MASSpy installation, including instructions for using MASSpy with Docker, are found in the repository README or in the documentation (<https://masspy.readthedocs.io> and S2 File). The data, scripts, and instructions needed to reproduce results of the presented examples are also available on GitHub (<https://github.com/SBRG/MASSpy-publication>) and in the supplement (S3 File).

Documentation

The documentation for MASSpy is available online (<https://masspy.readthedocs.io>) and in the S2 File. Good documentation is vital to the adoption and success of a software package; it

should teach new users how to get started while showing more experienced users how to fully capitalize on the software's features [5, 83]. For new users, MASSpy provides several simple tutorials demonstrating the usage of MASSpy's features and its capabilities. The MASSpy documentation also contains a growing collection of examples that demonstrate the use of MASSpy, including examples of workflows, advanced visualization tutorials, and in-depth textbook [29] examples that teach the fundamentals for dynamic modeling of mass action kinetics (S2 File).

Improvements and community outreach

The MASSpy package is designed to provide various dynamic modeling tools for the openCOBRA community; therefore a substantial portion of future development for MASSpy will be tailored toward fulfilling the needs of the COBRA community based on user feedback and feature requests. New MASSpy releases will utilize GitHub for version control and adhere to Semantic Versioning guidelines (<https://semver.org>) in order to inform the community about the compatibility and scope of improvements. Examples of potential improvements for future releases of MASSpy include bug fixes, additional SBML compatibility, new import/export formats, support for additional modeling standards, explicit support for additional libRoadRunner simulation capabilities, and implementation of additional algorithms relevant to MASS modeling approaches. As the systems biology field continues to address challenges in dynamic models of metabolism, MASSpy will continue to expand its collection of modeling tools to support data-driven reconstruction and analysis of mechanistic models.

Supporting information

S1 Fig. Sensitivity analysis on flux split through isozymes in the *E. coli* glycolytic subnetwork. The Gibbs free energy of enzyme-catalyzed reactions and the fractional abundance for isozyme states for all isozyme pairs for (A) glucose growth conditions and (B) pyruvate growth conditions when computing the functional states of the *E. coli* proteome in the case study.

(TIF)

S2 Fig. Fractional abundance of all enzyme states in the *E. coli* glycolytic subnetwork. The fractional abundance for all enzymes states of all enzyme modules when computing the functional states of the *E. coli* proteome in the case study.

(TIF)

S1 File. The source code for MASSpy version 0.1.1. The latest version of the MASSpy software can be found at <https://github.com/SBRG/MASSpy>.

(ZIP)

S2 File. The documentation for MASSpy version 0.1.1. The latest version of the MASSpy documentation can be found at <https://masspy.readthedocs.io>.

(ZIP)

S3 File. Data, scripts, and instructions to produce presented results. All files necessary to reproduce the results presented in the validation, ensemble modeling feature demonstration, and case study. Alternatively, these files can be found at <https://github.com/SBRG/MASSpy-publication>.

(ZIP)

S1 Data. Data used to construct the *E. coli* glycolytic subnetwork. Includes growth data and volume measurements observed for *E. coli* growth on glucose and pyruvate media, and

equilibrium constants from eQuilibrator used in the case study to construct the *E. coli* glycolytic subnetwork.

(XLSX)

S2 Data. Steady state concentration data in the *E. coli* glycolytic subnetwork. Includes the steady state concentration data for all metabolites and enzymes in the *E. coli* glycolytic subnetwork in the case study.

(XLSX)

Acknowledgments

The authors gratefully acknowledge: Patrick Phaneuf and Laurence Yang for discussions about software design considerations during the development of the MASSpy software, Zak King for help concerning Escher interoperability and general software development, Bin Du for discussions about the implementation of enzyme modules and thermodynamic feasibility features, Colton Lloyd for discussions about COBRA methods and expanding the COBRAPy framework, and the users who provided feedback during the development process for the initial MASSpy release.

Author Contributions

Conceptualization: James T. Yurkovich, Bernhard O. Palsson.

Data curation: Zachary B. Haiman, Daniel C. Zielinski.

Formal analysis: Zachary B. Haiman, Daniel C. Zielinski.

Funding acquisition: Bernhard O. Palsson.

Investigation: Zachary B. Haiman, Daniel C. Zielinski, Yuko Koike, James T. Yurkovich, Bernhard O. Palsson.

Methodology: Zachary B. Haiman, Daniel C. Zielinski, Yuko Koike, James T. Yurkovich, Bernhard O. Palsson.

Project administration: Bernhard O. Palsson.

Resources: Bernhard O. Palsson.

Software: Zachary B. Haiman, Yuko Koike.

Supervision: Daniel C. Zielinski, James T. Yurkovich, Bernhard O. Palsson.

Validation: Zachary B. Haiman.

Visualization: Zachary B. Haiman, Daniel C. Zielinski, James T. Yurkovich.

Writing – original draft: Zachary B. Haiman, Daniel C. Zielinski, Yuko Koike, James T. Yurkovich, Bernhard O. Palsson.

Writing – review & editing: Zachary B. Haiman, Daniel C. Zielinski, Yuko Koike, James T. Yurkovich, Bernhard O. Palsson.

References

1. Jamshidi N, Palsson BØ. Formulating genome-scale kinetic models in the post-genome era. *Mol Syst Biol.* 2008; 4:171. <https://doi.org/10.1038/msb.2008.8>
2. Cuevas DA, Edirisinghe J, Henry CS, Overbeek R, O'Connell TG, Edwards RA. From DNA to FBA: How to Build Your Own Genome-Scale Metabolic Model. *Frontiers in Microbiology.* 2016; 7:907.

3. Gevorgyan A, Bushell ME, Avignone-Rossa C, Kierzek AM. SurreyFBA: a command line tool and graphics user interface for constraint-based modeling of genome-scale metabolic reaction networks. *Bioinformatics*. 2011; 27(3):433–434. <https://doi.org/10.1093/bioinformatics/btq679>
4. Marmiesse L, Peyraud R, Cottret L. FlexFlux: combining metabolic flux and regulatory network analyses. *BMC Systems Biology*. 2015; 9(1):93. <https://doi.org/10.1186/s12918-015-0238-z>
5. Carey MA, Dräger A, Beber ME, Papin JA, Yurkovich JT. Community standards to facilitate development and address challenges in metabolic modeling. *Molecular Systems Biology*. 2020; 16(8):e9235.
6. Edwards JS, Palsson BO. The *Escherichia coli* MG1655 in silico metabolic genotype: its definition, characteristics, and capabilities. *Proc Natl Acad Sci U S A*. 2000; 97(10):5528–5533. <https://doi.org/10.1073/pnas.97.10.5528>
7. Atala A. Re: Haem oxygenase is synthetically lethal with the tumour suppressor fumarate hydratase. *J Urol*. 2012; 187(4):1506. <https://doi.org/10.1016/j.juro.2011.12.029>
8. Yim H, Haselbeck R, Niu W, Pujol-Baxley C, Burgard A, Boldt J, et al. Metabolic engineering of *Escherichia coli* for direct production of 1,4-butanediol. *Nat Chem Biol*. 2011; 7(7):445–452. <https://doi.org/10.1038/nchembio.580> PMID: 21602812
9. Nam H, Lewis NE, Lerman JA, Lee DH, Chang RL, Kim D, et al. Network context and selection in the evolution to enzyme specificity. *Science*. 2012; 337(6098):1101–1104. <https://doi.org/10.1126/science.1216861> PMID: 22936779
10. Bordbar A, Palsson BO. Using the reconstructed genome-scale human metabolic network to study physiology and pathology. *J Intern Med*. 2012; 271(2):131–141. <https://doi.org/10.1111/j.1365-2796.2011.02494.x>
11. Sonnenschein N, Golib Dzib JF, Lesne A, Eilebrecht S, Boulkroun S, Zennaro MC, et al. A network perspective on metabolic inconsistency. *BMC Syst Biol*. 2012; 6:41. <https://doi.org/10.1186/1752-0509-6-41> PMID: 22583819
12. McCloskey D, Palsson BØ, Feist AM. Basic and applied uses of genome-scale metabolic network reconstructions of *Escherichia coli*. *Mol Syst Biol*. 2013; 9:661. <https://doi.org/10.1038/msb.2013.18>
13. Chang RL, Andrews K, Kim D, Li Z, Godzik A, Palsson BO. Structural systems biology evaluation of metabolic thermotolerance in *Escherichia coli*. *Science*. 2013; 340(6137):1220–1223. <https://doi.org/10.1126/science.1234012>
14. Brynildsen MP, Winkler JA, Spina CS, MacDonald IC, Collins JJ. Potentiating antibacterial activity by predictably enhancing endogenous microbial ROS production. *Nat Biotechnol*. 2013; 31(2):160–165. <https://doi.org/10.1038/nbt.2458>
15. Lewis NE, Nagarajan H, Palsson BO. Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods. *Nat Rev Microbiol*. 2012; 10(4):291–305. <https://doi.org/10.1038/nrmicro2737>
16. Heirendt L, Arreckx S, Pfau T, Mendoza SN, Richelle A, Heinken A, et al. Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0. *Nat Protoc*. 2019; 14(3):639–702. <https://doi.org/10.1038/s41596-018-0098-2> PMID: 30787451
17. Varma A, Palsson BO. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *Escherichia coli* W3110. *Appl Environ Microbiol*. 1994; 60(10):3724–3731. <https://doi.org/10.1128/AEM.60.10.3724-3731.1994>
18. Bordbar A, Yurkovich JT, Paglia G, Rolfsson O, Sigurjónsson ÓE, Palsson BO. Elucidating dynamic metabolic physiology through network integration of quantitative time-course metabolomics. *Sci Rep*. 2017; 7:46249. <https://doi.org/10.1038/srep46249>
19. Kleessen S, Irgang S, Klie S, Giavalisco P, Nikoloski Z. Integration of transcriptomics and metabolomics data specifies the metabolic response of *Chlamydomonas* to rapamycin treatment. *Plant J*. 2015; 81(5):822–835. <https://doi.org/10.1111/tpj.12763>
20. Machado D, Costa RS, Ferreira EC, Rocha I, Tidor B. Exploring the gap between dynamic and constraint-based models of metabolism. *Metab Eng*. 2012; 14(2):112–119. <https://doi.org/10.1016/j.ymben.2012.01.003>
21. Stanford NJ, Lubitz T, Smallbone K, Klipp E, Mendes P, Liebermeister W. Systematic construction of kinetic models from genome-scale metabolic networks. *PLoS One*. 2013; 8(11):e79195. <https://doi.org/10.1371/journal.pone.0079195>
22. Tran LM, Rizk ML, Liao JC. Ensemble Modeling of Metabolic Networks. *Biophysical Journal*. 2008; 95(12):5606–5617. <https://doi.org/10.1529/biophysj.108.135442> PMID: 18820235
23. Liepe J, Barnes C, Cule E, Erguler K, Kirk P, Toni T, et al. ABC-SysBio—approximate Bayesian computation in Python with GPU support. *Bioinformatics*. 2010; 26(14):1797–1799. <https://doi.org/10.1093/bioinformatics/btq278> PMID: 20591907

24. Yurkovich JT, Alcantar MA, Haiman ZB, Palsson BO. Network-level allosteric effects are elucidated by detailing how ligand-binding events modulate utilization of catalytic potentials. *PLoS Comput Biol*. 2018; 14(8):e1006356. <https://doi.org/10.1371/journal.pcbi.1006356>
25. Chance B, Garfinkel D, Higgins J, Hess B, the technical assistance of E M Chance II W. Metabolic Control Mechanisms: V. A SOLUTION FOR THE EQUATIONS REPRESENTING INTERACTION BETWEEN GLYCOLYSIS AND RESPIRATION IN ASCITES TUMOR CELLS. *Journal of Biological Chemistry*. 1960; 235(8):2426–2439. [https://doi.org/10.1016/S0021-9258\(18\)64638-1](https://doi.org/10.1016/S0021-9258(18)64638-1)
26. Jamshidi N, Palsson BØ. Mass Action Stoichiometric Simulation Models: Incorporating Kinetics and Regulation into Stoichiometric Models. *Biophysical Journal*. 2010; 98(2):175–185. <https://doi.org/10.1016/j.bpj.2009.09.064>
27. Du B, Zielinski DC, Kavas ES, Dräger A, Tan J, Zhang Z, et al. Evaluation of rate law approximations in bottom-up kinetic models of metabolism. *BMC Syst Biol*. 2016; 10(1):40. <https://doi.org/10.1186/s12918-016-0283-2> PMID: 27266508
28. Bordbar A, McCloskey D, Zielinski DC, Sonnenschein N, Jamshidi N, Palsson BO. Personalized Whole-Cell Kinetic Models of Metabolism for Discovery in Genomics and Pharmacodynamics. *Cell Syst*. 2015; 1(4):283–292. <https://doi.org/10.1016/j.cels.2015.10.003>
29. Palsson BØ. *Systems Biology: Simulation of Dynamic Network States*. Cambridge University Press; 2011.
30. Soo V, Kulikowski C, Garfinkel D, Garfinkel L. Theory formation in postulating enzyme kinetic mechanisms: reasoning with constraints. *Computers and biomedical research, an international journal*. 1988; 21(4):381–403. [https://doi.org/10.1016/0010-4809\(88\)90052-3](https://doi.org/10.1016/0010-4809(88)90052-3)
31. Anand, Sonnenschein N. *opencobra/MASS-Toolbox: Updated Documentation*; 2017. Available from: <https://doi.org/10.5281/zenodo.803492>.
32. Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COConstraints-Based Reconstruction and Analysis for Python. *BMC Syst Biol*. 2013; 7:74. <https://doi.org/10.1186/1752-0509-7-74>
33. Ekmekci B, McAnany CE, Mura C. An Introduction to Programming for Bioscientists: A Python-Based Primer. *PLoS Comput Biol*. 2016; 12(6):e1004867. <https://doi.org/10.1371/journal.pcbi.1004867>
34. Hinsin K. High-Level Scientific Programming with Python. In: Sloot PMA, Hoekstra AG, Tan CJK, Dongarra JJ, editors. *Computational Science—ICCS 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002. p. 691–700.
35. Yurkovich JT, Yurkovich BJ, Dräger A, Palsson BO, King ZA. A Padawan Programmer's Guide to Developing Software Libraries. *Cell Syst*. 2017; 5(5):431–437. <https://doi.org/10.1016/j.cels.2017.08.003>
36. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*. 2020; 17(3):261–272. <https://doi.org/10.1038/s41592-019-0686-2> PMID: 32015543
37. van der Walt S, Colbert SC, Varoquaux G. The NumPy Array: A Structure for Efficient Numerical Computation. *Comput Sci Eng*. 2011; 13(2):22–30. <https://doi.org/10.1109/MCSE.2011.37>
38. Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*. 2017; 3:e103. <https://doi.org/10.7717/peerj-cs.103>
39. Reback J, McKinney W, jbrockmendel, den Bossche JV, Augspurger T, Cloud P, et al. *pandas-dev/pandas: Pandas 1.1.3*; 2020. Available from: <https://doi.org/10.5281/zenodo.4067057>.
40. Hunter JD. Matplotlib: A 2D Graphics Environment. *Comput Sci Eng*. 2007; 9(3):90–95. <https://doi.org/10.1109/MCSE.2007.55>
41. Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: *ELPUB*; 2016. p. 87–90.
42. Perez F, Granger BE. IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*. 2007; 9(3):21–29. <https://doi.org/10.1109/MCSE.2007.53>
43. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J*. 2014; 2014(239).
44. King ZA, Dräger A, Ebrahim A, Sonnenschein N, Lewis NE, Palsson BO. Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. *PLoS Computational Biology*. 2015; 11(8):1–13.
45. Somogyi ET, Bouteiller JM, Glazier JA, König M, Medley JK, Swat MH, et al. libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics*. 2015; 31(20):3315–3321. <https://doi.org/10.1093/bioinformatics/btv363> PMID: 26085503
46. Bornstein BJ, Keating SM, Jouraku A, Hucka M. LibSBML: an API library for SBML. *Bioinformatics*. 2008; 24(6):880–881. <https://doi.org/10.1093/bioinformatics/btn051>

47. Jensen K, G R Cardoso J, Sonnenschein N. Optlang: An algebraic modeling language for mathematical optimization. *JOSS*. 2017; 2(9):139. <https://doi.org/10.21105/joss.00139>
48. Makhorin AO. GNU Linear Programming Kit; 2018.
49. Schellenberger J, Que R, Fleming RMT, Thiele I, Orth JD, Feist AM, et al. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*. 2011; 6(9):1290–1307. <https://doi.org/10.1038/nprot.2011.308> PMID: 21886097
50. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*. 2003; 19(4):524–531. <https://doi.org/10.1093/bioinformatics/btg015> PMID: 12611808
51. Nowak U, Weimann L. A Family of Newton Codes for Systems of Highly Nonlinear Equations. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1991; 1991.
52. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, et al. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans Math Softw*. 2005; 31(3):363–396. <https://doi.org/10.1145/1089014.1089020>
53. Hucka M, Bergmann FT, Hoops S, Keating SM, Sahle S, Schaff JC, et al. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. *J Integr Bioinform*. 2015; 12(2):266. <https://doi.org/10.2390/biecoll-jib-2015-266> PMID: 26528564
54. Brett GOLivier FTB. SBML Level 3 Package: Flux Balance Constraints version 2. *J Integr Bioinform*. 2018; 15(1).
55. Michael Hucka LPS. SBML Level 3 package: Groups, Version 1 Release 1. *J Integr Bioinform*. 2016; 13(3):290.
56. EMCA International. Standard ECMA-404; 2017.
57. Ulusu NN. Evolution of Enzyme Kinetic Mechanisms. *Journal of Molecular Evolution*. 2015; 80(5):251–257.
58. Schellenberger J, Palsson BØ. Use of randomized sampling for analysis of metabolic networks. *J Biol Chem*. 2009; 284(9):5457–5461. <https://doi.org/10.1074/jbc.R800048200>
59. Khodayari A, Zomorodi AR, Liao JC, Maranas CD. A kinetic model of Escherichia coli core metabolism satisfying multiple sets of mutant flux data. *Metab Eng*. 2014; 25:50–62. <https://doi.org/10.1016/j.ymben.2014.05.014>
60. Kaufman DE, Smith RL. Direction Choice for Accelerated Convergence in Hit-and-Run Sampling. *Operations Research*. 1998; 46(1):84–95. <https://doi.org/10.1287/opre.46.1.84>
61. Megchelenbrink W, Huynen M, Marchiori E. optGpSampler: an improved tool for uniformly sampling the solution-space of genome-scale metabolic networks. *PLoS One*. 2014; 9(2):e86587. <https://doi.org/10.1371/journal.pone.0086587>
62. Kümmel A, Panke S, Heinemann M. Putative regulatory sites unraveled by network-embedded thermodynamic analysis of metabolome data. *Mol Syst Biol*. 2006; 2:2006.0034.
63. Noor E, Bar-Even A, Flamholz A, Reznik E, Liebermeister W, Milo R. Pathway Thermodynamics Highlights Kinetic Obstacles in Central Metabolism. *PLoS Comput Biol*. 2014; 10(2):e1003483. <https://doi.org/10.1371/journal.pcbi.1003483>
64. Tan Y, Liao JC. Metabolic ensemble modeling for strain engineers. *Biotechnol J*. 2012; 7(3):343–353. <https://doi.org/10.1002/biot.201100186>
65. Bordbar A, Lewis NE, Schellenberger J, Palsson BØ, Jamshidi N. Insight into human alveolar macrophage and M. tuberculosis interactions via metabolic reconstructions. *Molecular Systems Biology*. 2010; 6(1):422. <https://doi.org/10.1038/msb.2010.68>
66. Wan J, Ristenpart WD, Stone HA. Dynamics of shear-induced ATP release from red blood cells. *Proc Natl Acad Sci U S A*. 2008; 105(43):16432–16437. <https://doi.org/10.1073/pnas.0805779105>
67. Conn AR, Gould NIM, Toint PL. Trust Region Methods. Society for Industrial and Applied Mathematics; 2000. Available from: <https://epubs.siam.org/doi/abs/10.1137/1.9780898719857>.
68. Gerosa L, Haverkorn van Rijsewijk BRB, Christodoulou D, Kochanowski K, Schmidt TSB, Noor E, et al. Pseudo-transition Analysis Identifies the Key Regulators of Dynamic Metabolic Adaptations from Steady-State Data. *Cell Syst*. 2015; 1(4):270–282. <https://doi.org/10.1016/j.cels.2015.09.008> PMID: 27136056
69. Monk JM, Lloyd CJ, Brunk E, Mih N, Sastry A, King Z, et al. i ML1515, a knowledgebase that computes Escherichia coli traits. *Nat Biotechnol*. 2017; 35(10):904–908. <https://doi.org/10.1038/nbt.3956> PMID: 29020004
70. Schmidt A, Kochanowski K, Vedelaar S, Ahm e E, Volkmer B, Callipo L, et al. The quantitative and condition-dependent Escherichia coli proteome. *Nat Biotechnol*. 2016; 34(1):104–110. <https://doi.org/10.1038/nbt.3418> PMID: 26641532

71. Volkmer B, Heinemann M. Condition-dependent cell volume and concentration of *Escherichia coli* to facilitate data conversion for systems biology modeling. *PLoS One*. 2011; 6(7):e23126. <https://doi.org/10.1371/journal.pone.0023126>
72. Flamholz A, Noor E, Bar-Even A, Milo R. eQuilibrator—the biochemical thermodynamics calculator. *Nucleic Acids Research*. 2012; 40(D1):D770–D775. <https://doi.org/10.1093/nar/gkr874>
73. Noor E, Haraldsdóttir HS, Milo R, Fleming RMT. Consistent estimation of Gibbs energy using component contributions. *PLoS Comput Biol*. 2013; 9(7):e1003098. <https://doi.org/10.1371/journal.pcbi.1003098>
74. Mitra ED, Suderman R, Colvin J, Ionkov A, Hu A, Sauro HM, et al. PyBioNetFit and the Biological Property Specification Language. *iScience*. 2019; 19:1012–1036. <https://doi.org/10.1016/j.isci.2019.08.045> PMID: 31522114
75. Shockley EM, Vrugt JA, Lopez CF. PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics (Oxford, England)*. 2018; 34(4):695–697. <https://doi.org/10.1093/bioinformatics/btx626>
76. Salvatier J, Wiecki TV, Fonnesbeck C. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*. 2016; 2:e55. <https://doi.org/10.7717/peerj-cs.55>
77. VanderPlas J, Granger B, Heer J, Moritz D, Wongsuphasawat K, Satyanarayan A, et al. Altair: Interactive Statistical Visualizations for Python. *Journal of Open Source Software*. 2018; 3(32):1057. <https://doi.org/10.21105/joss.01057>
78. Ortega OO, Lopez CF. Interactive Multiresolution Visualization of Cellular Network Processes. *iScience*. 2020; 23(1). <https://doi.org/10.1016/j.isci.2019.100748> PMID: 31884165
79. Olivier BG, Rohwer JM, Hofmeyr JHS. Modelling cellular systems with PySCeS. *Bioinformatics*. 2005; 21(4):560–561. <https://doi.org/10.1093/bioinformatics/bti046>
80. Choi K, Medley JK, König M, Stocking K, Smith L, Gu S, et al. Tellurium: An extensible python-based modeling environment for systems and synthetic biology. *Biosystems*. 2018; 171:74–79. <https://doi.org/10.1016/j.biosystems.2018.07.006> PMID: 30053414
81. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI—a COmplex PATHway Simulator. *Bioinformatics*. 2006; 22(24):3067–3074. <https://doi.org/10.1093/bioinformatics/btl485> PMID: 17032683
82. Lopez CF, Muhlich JL, Bachman JA, Sorger PK. Programming biological models in Python using PySB. *Mol Syst Biol*. 2013; 9:646. <https://doi.org/10.1038/msb.2013.1>
83. Prlić A, Procter JB. Ten simple rules for the open development of scientific software. *PLoS Comput Biol*. 2012; 8(12):e1002802. <https://doi.org/10.1371/journal.pcbi.1002802>