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

Zachary B. Haiman[1], Daniel C. Zielinski[1], Yuko Koike[1,2], James T. Yurkovich[2], Bernhard O. Palsson[1,3*],

1 Department of Bioengineering, University of California San Diego, La Jolla, CA, United States of America
2 Institute for Systems Biology, Seattle, WA, United States of America
3 Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

* palsson@ucsd.edu (BOP)

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 COnstraint-Based Reconstruction and Analysis Python (COBRApy) package to provide an 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 case studies demonstrate how to use MASSpy: 1) to simulate dynamics of detailed mechanisms of enzyme regulation; 2) to generate an ensemble of kinetic models using Monte Carlo sampling to approximate missing numerical values of parameters and to quantify uncertainty, and 3) 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 challenge that arise in dynamic modeling of metabolic networks, both at a small and large scale.

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 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.

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]. COnstraint-Based Reconstruction and Analysis (COBRA) methods [2] have been shown to be a scalable framework that is invaluable for the contextualization and analysis of multi-omic data, as well as for understanding, predicting, and engineering metabolism [3–12]. While several methods have been developed that allow COBRA models to integrate certain data types to model long timescale dynamics [13–15], COBRA models are inherently limited by the flux-balance assumption.

Kinetic modeling methods use detailed mechanistic information to model dynamic states of a network [16]. 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 [17, 18].

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 [17–20]. One such methodology is the Mass Action Stoichiometric Simulation (MASS) approach, in which mass action kinetics are used to construct condition-specific dynamic models [20–23]. The MASS modeling approach provides an algorithmic, data-driven workflow for generating *in vivo* kinetic models in a scalable fashion [24]. 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 [21]. The MASS modeling framework has been implemented in the MASS Toolbox [25], 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 COnstraint-Based Reconstruction and Analysis Python (COBRApy) [26] 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.

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 with a clean syntax that has become widely adopted in the scientific community due to its unique features (e.g., a flexible interface to compiled languages such as C++ [27]). The open-source nature of Python avoids the inherent limitations associated with costly commercial software [28]. Consequently, developing in Python provides access to a growing variety of open-source scientific software libraries [29,30], several of which are integrated into the MASSpy package and utilized for various purposes (Table 1).

Table 1. Overview of external dependencies and their relevance to MASSpy functionality.

Package	Version	MASSpy Relevance	Reference
COBRApy	0.15.0	Reconstruction and simulation of genome-scale flux states	[26]
Escher	1.7.2	Visualization of pathway and node maps	[31]
libRoadRunner*	1.5.0 1	Dynamic simulation and steady state determination through NLEQ methods	[32]
libSBML*	5.18.0 1	A Python interface for reading and writing models in SBML	[33]
Matplotlib*	3.2.0	Visualization of simulation results	[34]
Numpy	1.13.0	Fundamental package for numerical computation in Python. Provides efficient array/matrix data types and operations.	[35]
OptLang	1.4.2	Formulation of optimization problems using symbolic expressions and native Python algebra syntax. Provides a common interface for various optimization solver backends.	[36]
Pandas	0.17.0	High-performance data structures and analysis tools for data science	[37]
SciPy*	1.2.0	A collection of scientific algorithms. Primarily used for interpolation of dynamic simulation results and linear algebra operations.	[38]
SymPy	1.0.0	Generation and manipulation of symbolic mathematical expressions, including ordinary differential equations, rate laws, and optimization problems.	[39]
swiglpk	1.4.3	A Python interface to the GNU Linear Programming Kit used for optimization. Utilized by OptLang to provide LP and MILP support.	[40]
cplex**	12.8.8.0	A Python interface to the CPLEX Optimizer used for optimization. Utilized by OptLang to provide LP, MILP, and QP support.	IBM, Armonk, NY
gurobi**	5.0.2	A Python interface to the Gurobi Optimizer used for optimization. Utilized by OptLang to provide LP, MILP, and QP support.	Gurobi Optimization, Houston, TX

* Additional packages required to enable all MASSpy features; all other packages are strict or optional dependencies of COBRApy.

** Commercial optimization solvers with Python APIs with free academic licenses. Abbreviations: Linear Programming (LP); Mixed Integer Linear Programming (MILP); Quadratic Programming (QP); Systems Biology Markup Language (SBML);

Building on the COBRApy framework

To facilitate the integration of constraint-based and dynamic modeling frameworks, MASSpy utilizes the COBRApy package [26] 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). 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 [41]; therefore, preserving COBRApy 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 [17]. MASSpy utilizes SymPy [39] to represent reaction rates and differential equations as symbolic expressions. All MassReaction objects automatically generate their own rate laws using mass action kinetics, unless a suitable rate law is available from literature and assigned to the reaction. 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 [32], a high-performance Systems Biology Markup Language (SBML) [42] 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 [43], and dynamic simulation via integration of ODEs through deterministic integrators, including CVODE solver from the Sundials suite [44]. 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 [45] along with the FBC [46] and Groups [47] 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 [42], SBML is a widely used model format specifically among members of the COBRA modeling community [41].

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 [48]. The MASSpy JSON schema is designed for interoperability with Escher [31], a pathway visualization tool designed to visualize various -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 [22]. 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. [22]. Given the number and complexity of possible enzymatic mechanisms [49], 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 [18]. Ensemble modeling refers to the assembly of dynamic models that span the feasible kinetic solution space and is useful 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 [50, 51]. 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 [52] and another with multiprocessing support [53]. To sample metabolite concentrations, MASSpy employs a ConcSolver object to populate the optimization solver with constraints for thermodynamically feasible concentration ranges [23, 54, 55] 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 conducted three different case studies that exemplified how MASSpy features combined to facilitate dynamic modeling of metabolism (Fig 1). In Case Study 1, we validated MASSpy as a modeling tool by describing mechanisms of enzyme regulation using enzymes modules [20]. We demonstrated the utility of the software in Case Study 2, generating an ensemble of stable kinetic models through MCMC sampling to examine biological variability while satisfying thermodynamic constraints imposed by the network. In Case Study 3, 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.

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.

Case Study 1: 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. [20]. The authors used the MASS Toolbox [25] to elucidate the systems-level effects of allosteric regulation. We used MASSpy to reconstruct enzyme modules for hexokinase, phosphofructokinase, and pyruvate kinase in RBC glycolysis using the same mechanisms as previously described [20]. We provided several in-depth tutorials for constructing MASS models and enzyme modules in MASSpy, which can be found in the documentation (S2 File).

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) 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 [20] (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 [20]. See S3 File for all data and scripts associated with this case study, including kinetic parameters for all three enzyme modules.

Case Study 2: Ensemble sampling, assembly, and modeling

Many ensemble modeling approaches utilize sampling methods to approximate missing values and quantify uncertainty in metabolic models [17, 18, 23, 51, 56, 57]. To demonstrate the 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 [20] we used MCMC sampling to generate 25 candidate flux

Table 2. Comparison of explicitly supported features for dynamic modeling tools.

Software		**MASSpy**	**MASS Toolbox**	**Tellurium**	**PySB**	**PySCeS (CMBPy)**	**COPASI**
Version		0.1.0	1.2.0	2.1.5	1.11.0	0.9.7 (0.7.25)	4.27.217
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
Model construction	Model merging	+	+	+	+		+
	Automated rate law construction	(+) only mass action	(+) only mass action		(+) only mass action		+
	Enzyme modules	+	+		(+) monomers		
	Symbolic expression manipulation	+	+				
	GPR handling	+	+	+		+	
Sampling and estimation	Flux sampling	+	+				
	Concentration sampling	+					
	Parameter estimation	(+) PERCs	(+) PERCs				+
Simulation	ODE	+	+	+	+	+	+
	Stochastic			+	+		+
Analysis	Steady state	+	+	+	(+) via simulation	+	+
	Stoichiometric	+	+	+		+	+
	Sensitivity			+	+	+	+
	Metabolic control			+		+	+
	Stability	+	+	+		+	+
	Gene knockouts	+	+			+	
	Flux balance / flux variability	+	+			+	
Visualization	Time profiles	+	+	+		+	+
	Phase portraits	+	+	+		+	+
	Pathway maps	(+) via Escher	+	(+) via Graphviz	(+) via Graphviz	(+) via FAME	+
Quality control and assurance	Elemental balancing	+	+			+	
	Thermodynamic feasibility	+	+				
	Assistance w/ undefined values	+	+				
Standards	SBML	+	+	+	+	+	+
	SED-ML			+		(+) export	(+) export
	COMBINE			+		(+) export	+

"+" denotes explicit support for the feature. "(+)" denotes explicit support via an interface, or with some limitations. Explicit support is determined based on whether clear methods and documentation demonstrating feature support are available from the listed software. For example, although MASSpy could utilize the stochastic simulation capabilities of libRoadRunner, no MASSpy methods or documentation providing explicit support for stochastic simulation in MASSpy exist currently.

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 [31]. 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. [20]. (C) The different signals of the enzyme module can be observed to provide enzyme-level resolution of the regulatory response.

states and 25 candidate thermodynamically feasible concentration states, allowing for variables to deviate from their reference state by up to 80 percent. This procedure resulted in 625 models that represent all possible combinations of flux and concentration states. We calculated pseudo-elementary rate constants (PERCs) and steady-state for each model. We simulated a 50% increase in ATP utilization to mimic a physiologically relevant disturbance, such as increased shear stress due to arterial constriction [58]. Out of 625 models, 15 were discarded due to an inability to reach a steady state.

Fig 3. A workflow for ensemble creation and modeling using MCMC sampling. The general process for generating and assembling an ensemble of models for dynamic simulation and analysis. (A) The solution spaces for fluxes and concentrations are sampled using MCMC sampling to generate data for ensemble creation. Rate constants are obtained through parameter fitting for elementary rate constants and computation of PERCs in addition to MCMC sampling. (B) Sampling data is integrated into models, producing an assembly of models with variations in flux and concentration states. After models are created, ensembles of models can be studied through (C) dynamic simulation and (D) analysis.

We then reconstructed enzyme modules for pyruvate kinase [20] for the remaining 610 models. Numerical values of rate constants for each enzyme were determined using the SciPy implementation of a trust-region method for nonlinear convex optimization [59]. 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 filter out infeasible models based on whether they could reach a stable steady state. Out of 610 models, 446 could not reach a steady state and 92 could not reach a steady state with the perturbation, leaving 72 stable models for ensemble 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 40% from its baseline value (Fig 3C). Steady state analysis of the pyruvate kinase enzyme modules after the disturbance revealed a strong preference for the enzyme to remain in an active state, with a median value of approximately 77%. The differences in candidate flux and concentration states resulted in an interquartile range between 62-86% of total pyruvate kinase, with nearly all variations of pyruvate kinase in the ensemble maintaining a steady state value of at least 30% active. Furthermore, examination of the relative flux load through the $\mathrm{R}_{\mathrm{i,AP}}$ forms showed that most of the flux load was carried by the $\mathrm{R}_{2,\mathrm{AP}}$ and $\mathrm{R}_{3,\mathrm{AP}}$ reaction steps while a miniscule fraction was carried by the $\mathrm{R}_{0,\mathrm{AP}}$ reaction step, regardless of variation. However, the variations had an effect on whether $\mathrm{R}_{2,\mathrm{AP}}$ carried more flux than $\mathrm{R}_{3,\mathrm{AP}}$, and whether the remaining flux was predominantly distributed through the $\mathrm{R}_{1,\mathrm{AP}}$ or the $R_{4,AP}$ reaction step. Through this case study, 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 this case study.

Case Study 3: 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 Luca et. al [60] into the iML1515 genome-scale metabolic reconstruction of *E. coli* K-12 MG1655 [61]. 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 [62].

Once the flux state was obtained for each carbon source, the glycolytic subnetwork was extracted from iML1515. Flux units were converted into molar units using volumetric measurements of *E. coli* obtained from Volkmer and Heinemann [63], and equilibrium constants obtained from eQuilibrator [63] through component contribution [64] were set for each reaction. Concentration growth data from Luca et. al [60] 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.

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.

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 phosphoglucoisomerase (PGI), 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

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)

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 S4 File. 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 this 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, [31]), and exchanging models in SBML format (libSBML [33]). 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 [20–24], a formal implementation of the MASS framework has only existed on a commercial software platform (Mathematica). MASSpy's seamless integration with COBRApy offers a vast array of constraint-based and dynamic modeling tools within a single open-source framework. 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. 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.

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 [65] could be utilized to perform stochastic simulations. Users who often analyze sensitivity may prefer Tellurium and its implementation of libRoadRunner [32, 66] 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 [67]) or a rule-based modeling approach (PySB [29]): 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 [45].

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

MASSpy version 0.1.0 is available as a Python package hosted on the Python Package Index (https://pypi.org/project/masspy/), licensed under GNU General Public License, version 3.0 (GPL-3.0). All 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 CPLEX Optimizer (IBM, Armonk, NY) are freely available for academic use, with solvers and installation instructions found at their respective websites. The latest source code for MASSpy is currently available on GitHub (https://github.com/SBRG/MASSpy) and is compatible with Mac OS X, Linux, and Windows operating systems. Instructions for MASSpy installation can be found in the repository README or in the documentation (S2 File). The data, scripts, and instructions needed to reproduce the results of the case studies can be found in the S3 File.

Documentation

The documentation for MASSpy is available online (https://masspy.readthedocs.io/). 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 [41, 68]. 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 [24] 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 Case study 3.

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 Case Study 3.

S1 File. The source code for MASSpy version 0.1.0. The latest version of the MASSpy software can be found at https://github.com/SBRG/MASSpy. (ZIP)

S2 File. The documentation for MASSpy version 0.1.0 The latest version of the MASSpy documentation can be found at https://masspy.readthedocs.io. (ZIP).

S3 File. Data and Jupyter notebooks for case studies. All files necessary to repeat each case study. Each folder contains the relevant data, scripts, and Jupyter notebooks for that case study. Alternatively, these files can be found at https://github.com/SBRG/MASSpy-publication (ZIP)

S4 File. 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 Case Study 3. (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.

References

1. Jamshidi N, Palsson BØ. Formulating genome-scale kinetic models in the post-genome era. Mol Syst Biol. 2008;4:171.

2. 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.

3. 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.

4. Atala A. Re: Haem oxygenase is synthetically lethal with the tumour suppressor fumarate hydratase. J Urol. 2012;187(4):1506.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. Tran LM, Rizk ML, Liao JC. Ensemble Modeling of Metabolic Networks; 2008.

19. 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.

20. 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.

21. Jamshidi N, Palsson BØ. Mass Action Stoichiometric Simulation Models: Incorporating Kinetics and Regulation into Stoichiometric Models; 2010.

22. Du B, Zielinski DC, Kavvas 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.

23. 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.

24. Palsson BØ. Systems Biology: Simulation of Dynamic Network States. Cambridge University Press; 2011.

25. Sastry A, Sonnenschein N. opencobra/MASS-Toolbox; 2017.

26. Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst Biol. 2013;7:74.

27. Hinsen K. High-Level Scientific Programming with Python; 2002.

28. 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.

29. Lopez CF, Muhlich JL, Bachman JA, Sorger PK. Programming biological models in Python using PySB. Mol Syst Biol. 2013;9:646.

30. Ekmekci B, McAnany CE, Mura C. An Introduction to Programming for Bioscientists: A Python-Based Primer. PLoS Comput Biol. 2016;12(6):e1004867.

31. 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; 2015.

32. 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.

33. Bornstein BJ, Keating SM, Jouraku A, Hucka M. LibSBML: an API library for SBML. Bioinformatics. 2008;24(6):880–881.

34. Hunter JD. Matplotlib: A 2D Graphics Environment. Comput Sci Eng. 2007;9(3):90–95.

35. 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.

36. Jensen K, G R Cardoso J, Sonnenschein N. Optlang: An algebraic modeling language for mathematical optimization. JOSS. 2017;2(9):139.

37. The pandas development team. pandas-dev/pandas: Pandas; 2020.

38. 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.

39. Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, et al.. SymPy: symbolic computing in Python; 2017.

40. Makhorin AO. GNU Linear Programming Kit; 2018.

41. Carey MA, Dräger A, Papin JA, Yurkovich JT. Community standards to facilitate development and address challenges in metabolic modeling; 2019.

42. 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.

43. Nowak U, Weimann L. A Family of Newton Codes for Systems of Highly Nonlinear Equations. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1991; 1991.

44. 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.

45. 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.

46. Brett G Olivier FTB. SBML Level 3 Package: Flux Balance Constraints version 2. J Integr Bioinform. 2018;15(1).

47. Michael Hucka LPS. SBML Level 3 package: Groups, Version 1 Release 1. J Integr Bioinform. 2016;13(3):290.

48. EMCA International. Standard ECMA-404; 2017.

49. Ulusu NN. Evolution of Enzyme Kinetic Mechanisms; 2015.

50. Schellenberger J, Palsson BØ. Use of randomized sampling for analysis of metabolic networks. J Biol Chem. 2009;284(9):5457–5461.

51. Khodayari A, Zomorrodi 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.

52. Kaufman DE, Smith RL. Direction Choice for Accelerated Convergence in Hit-and-Run Sampling; 1998.

53. 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.

54. 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.

55. 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.

56. Tan Y, Liao JC. Metabolic ensemble modeling for strain engineers. Biotechnol J. 2012;7(3):343–353.

57. Bordbar A, Lewis NE, Schellenberger J, Palsson BØ, Jamshidi N. Insight into human alveolar macrophage and M. tuberculosis interactions via metabolic reconstructions; 2010.

58. 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.

59. Conn AR, Gould NIM, Toint PL. Trust Region Methods; 2000.

60. 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.

61. 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.

62. Schmidt A, Kochanowski K, Vedelaar S, Ahrné E, Volkmer B, Callipo L, et al. The quantitative and condition-dependent Escherichia coli proteome. Nat Biotechnol. 2016;34(1):104–110.

63. 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.

64. Noor E, Haraldsdóttir HS, Milo R, Fleming RMT. Consistent estimation of Gibbs energy using component contributions. PLoS Comput Biol. 2013;9(7):e1003098.

65. Olivier BG, Rohwer JM, Hofmeyr JHS. Modelling cellular systems with PySCeS. Bioinformatics. 2005;21(4):560–561.

66. 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.

67. 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.

68. Prlić A, Procter JB. Ten simple rules for the open development of scientific software. PLoS Comput Biol. 2012;8(12):e1002802.

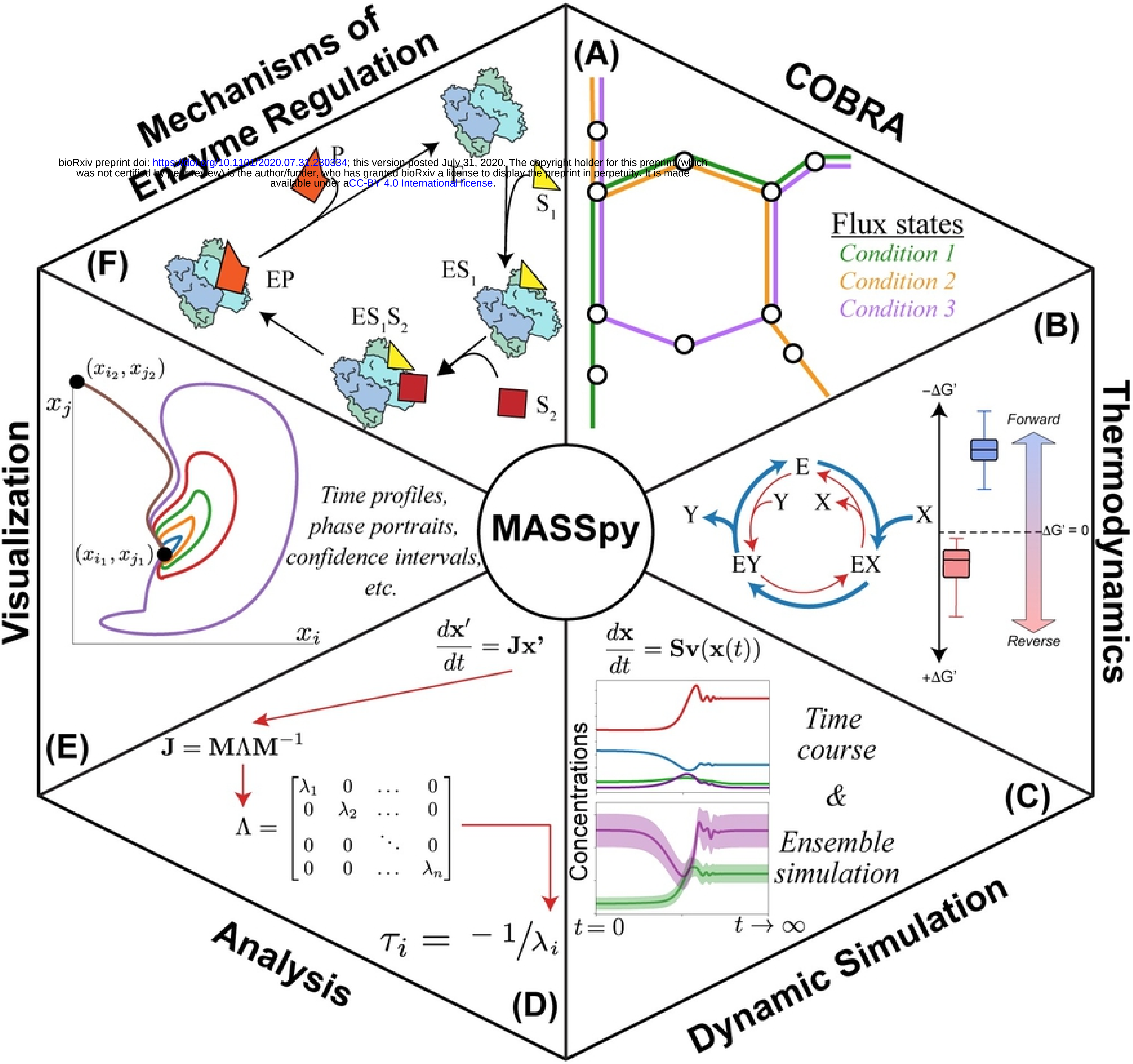
Mechanisms of Enzyme Regulation
(A)
COBRA
P
E
S$_1$
(F)
EP
ES$_1$
ES$_1$S$_2$
S$_2$
Flux states
Condition 1
Condition 2
Condition 3
(B)
Visualization
(x_{i_2}, x_{j_2})
x_j
(x_{i_1}, x_{j_1})
x_i
Time profiles, phase portraits, confidence intervals, etc.
MASSpy
E
Y
X
Y
X
EY
EX
−ΔG'
Forward
ΔG' = 0
Reverse
+ΔG'
Thermodynamics
$\frac{d\mathbf{x}'}{dt} = \mathbf{J}\mathbf{x}'$
$\frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v}(\mathbf{x}(t))$
(E)
$\mathbf{J} = \mathbf{M}\Lambda\mathbf{M}^{-1}$
$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$
$\tau_i = -1/\lambda_i$
Concentrations
Concentrations
Time course
&
Ensemble simulation
$t = 0$
$t \to \infty$
(C)
(D)
Analysis
Dynamic Simulation

Figure 1

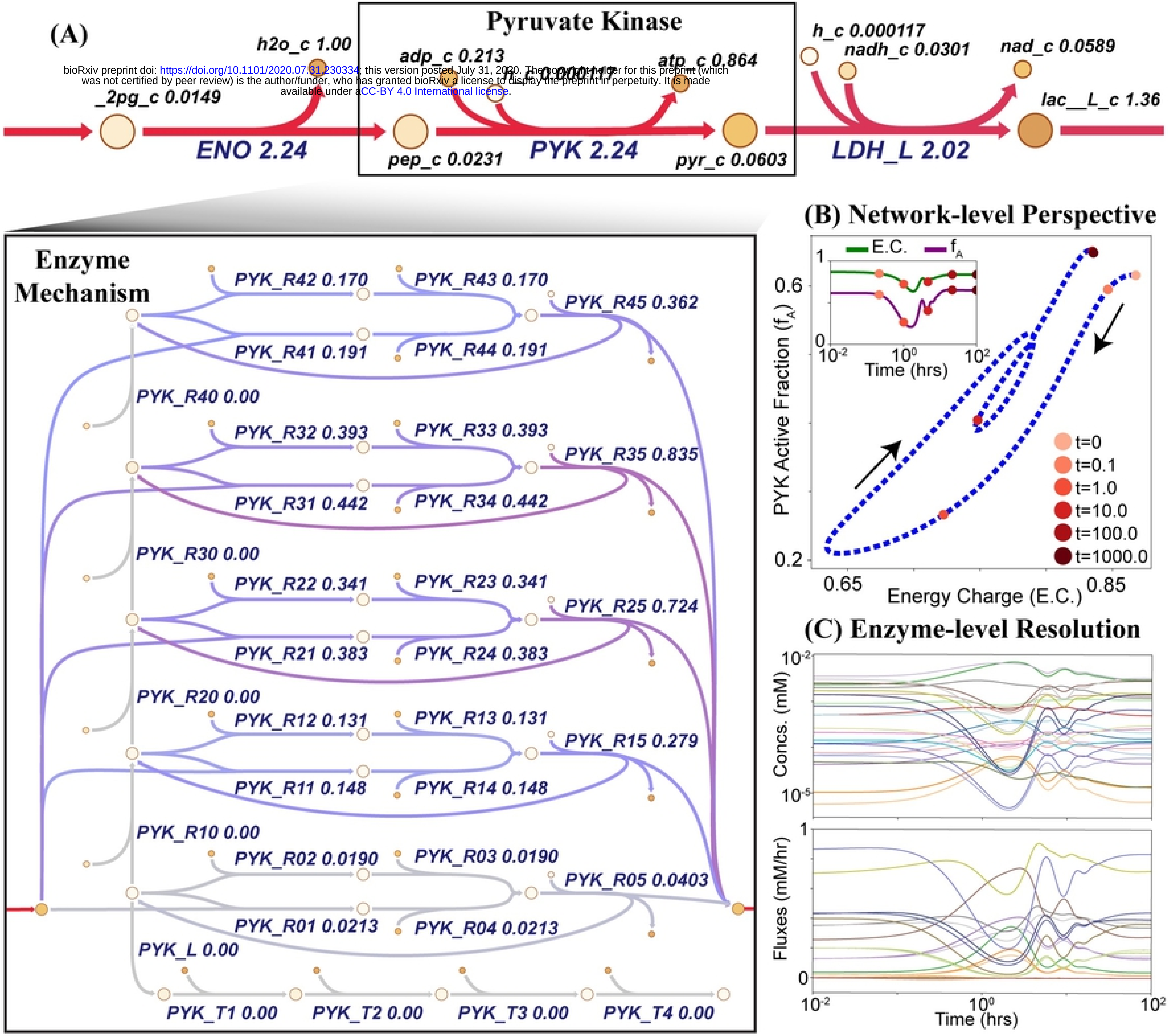
(A)
Pyruvate Kinase
h2o_c 1.00
_2pg_c 0.0149
ENO 2.24
adp_c 0.213
h_c 0.000117
atp_c 0.864
pep_c 0.0231
PYK 2.24
pyr_c 0.0603
h_c 0.000117
nadh_c 0.0301
nad_c 0.0589
lac__L_c 1.36
LDH_L 2.02
Enzyme Mechanism
PYK_R42 0.170
PYK_R43 0.170
PYK_R45 0.362
PYK_R41 0.191
PYK_R44 0.191
PYK_R40 0.00
PYK_R32 0.393
PYK_R33 0.393
PYK_R35 0.835
PYK_R31 0.442
PYK_R34 0.442
PYK_R30 0.00
PYK_R22 0.341
PYK_R23 0.341
PYK_R25 0.724
PYK_R21 0.383
PYK_R24 0.383
PYK_R20 0.00
PYK_R12 0.131
PYK_R13 0.131
PYK_R15 0.279
PYK_R11 0.148
PYK_R14 0.148
PYK_R10 0.00
PYK_R02 0.0190
PYK_R03 0.0190
PYK_R05 0.0403
PYK_R01 0.0213
PYK_R04 0.0213
PYK_L 0.00
PYK_T1 0.00
PYK_T2 0.00
PYK_T3 0.00
PYK_T4 0.00
(B) Network-level Perspective
E.C.
f_A
1
0
10^{-2}
10^0
10^2
Time (hrs)
PYK Active Fraction (f_A)
0.6
0.2
0.65
0.85
Energy Charge (E.C.)
t=0
t=0.1
t=1.0
t=10.0
t=100.0
t=1000.0
(C) Enzyme-level Resolution
Concs. (mM)
10^{-2}
10^{-5}
Fluxes (mM/hr)
1
0
10^{-2}
10^0
10^2
Time (hrs)

Figure 2

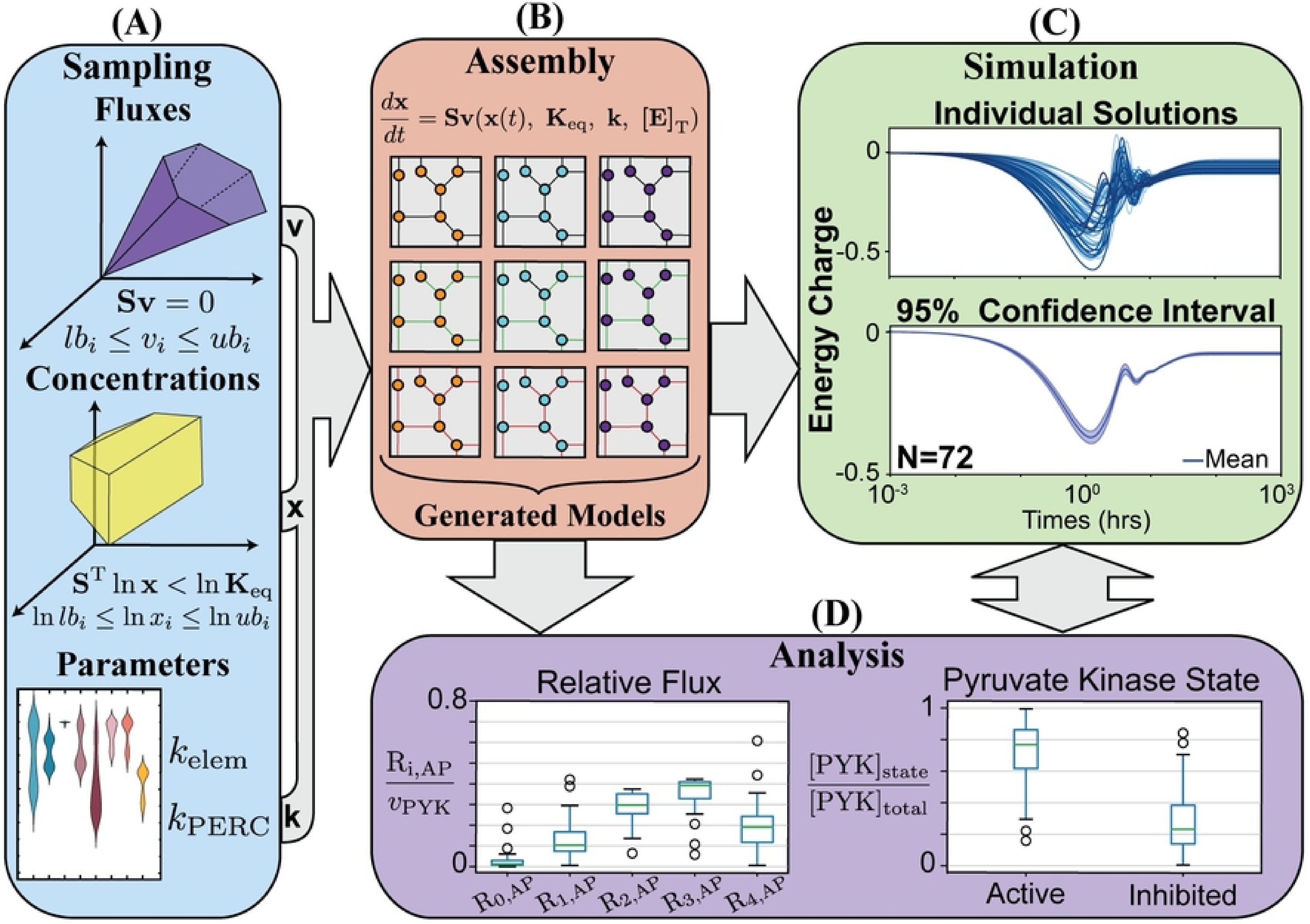
(A)
Sampling
Fluxes
$\mathbf{S}\mathbf{v} = 0$
$lb_i \leq v_i \leq ub_i$
Concentrations
$\mathbf{S}^{\mathrm{T}} \ln \mathbf{x} < \ln \mathbf{K}_{\mathrm{eq}}$
$\ln lb_i \leq \ln x_i \leq \ln ub_i$
Parameters
k_{elem}
k_{PERC}
v
x
k
(B)
Assembly
$\frac{d\mathbf{x}}{dt} = \mathbf{S}\mathbf{v}(\mathbf{x}(t),\ \mathbf{K}_{\mathrm{eq}},\ \mathbf{k},\ [\mathbf{E}]_{\mathrm{T}})$
Generated Models
(C)
Simulation
Individual Solutions
0
-0.5
95% Confidence Interval
0
-0.5
N=72
Mean
Energy Charge
10^{-3}
10^{0}
10^{3}
Times (hrs)
(D)
Analysis
Relative Flux
0.8
0
$\frac{\mathrm{R_{i,AP}}}{v_{\mathrm{PYK}}}$
$\mathrm{R_{0,AP}}$
$\mathrm{R_{1,AP}}$
$\mathrm{R_{2,AP}}$
$\mathrm{R_{3,AP}}$
$\mathrm{R_{4,AP}}$
Pyruvate Kinase State
1
0
$\frac{[\mathrm{PYK}]_{\mathrm{state}}}{[\mathrm{PYK}]_{\mathrm{total}}}$
Active
Inhibited

Figure 3

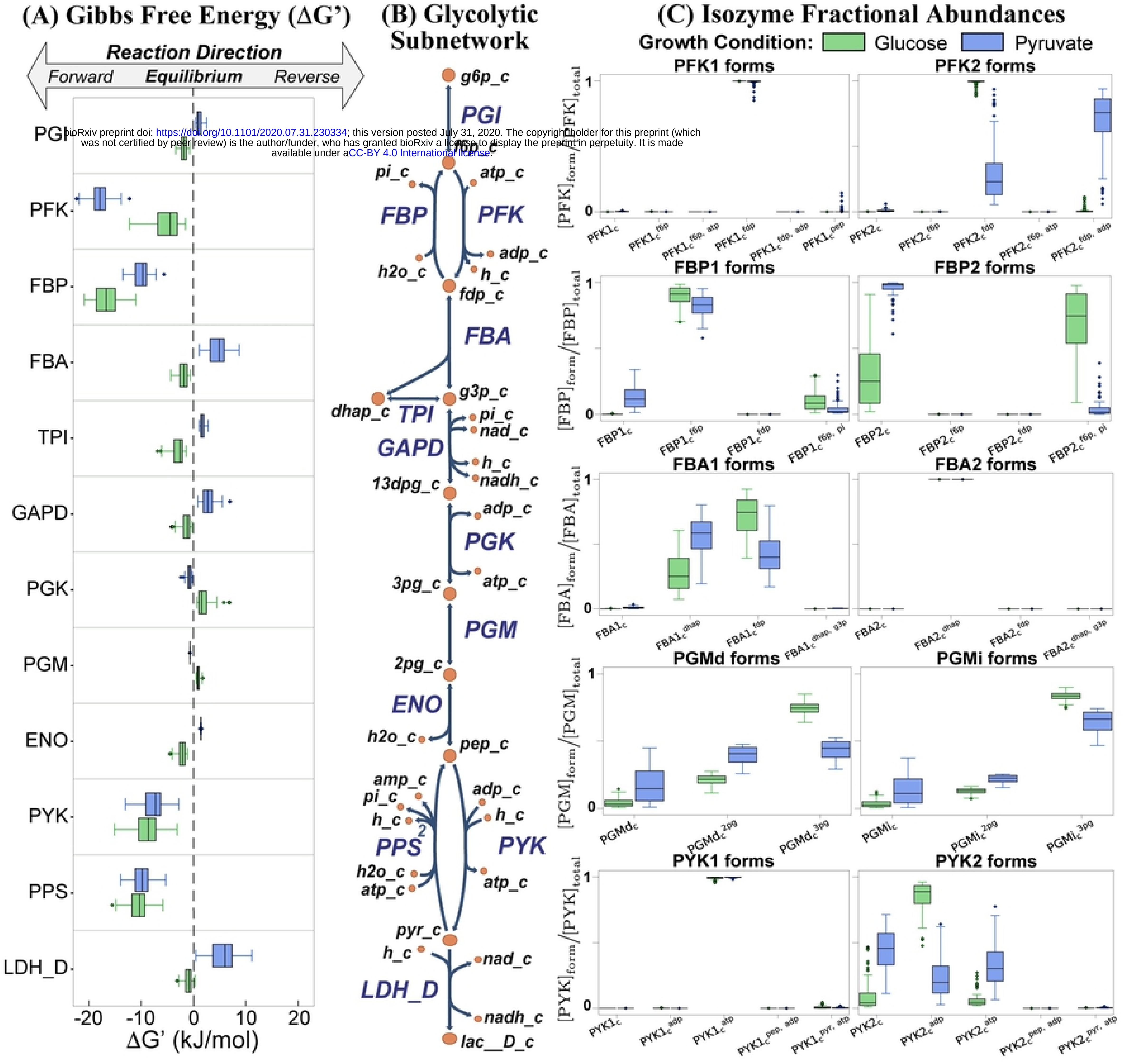
(A) Gibbs Free Energy (ΔG')
Reaction Direction
Forward
Equilibrium
Reverse
PGI
PFK
FBP
FBA
TPI
GAPD
PGK
PGM
ENO
PYK
PPS
LDH_D
-20
-10
0
10
20
ΔG' (kJ/mol)
(B) Glycolytic Subnetwork
g6p_c
PGI
f6p_c
pi_c
atp_c
FBP
PFK
h2o_c
adp_c
h_c
fdp_c
FBA
dhap_c
g3p_c
TPI
pi_c
nad_c
GAPD
h_c
nadh_c
13dpg_c
adp_c
PGK
3pg_c
atp_c
PGM
2pg_c
ENO
h2o_c
pep_c
amp_c
pi_c
h_c
2
adp_c
h_c
PPS
PYK
h2o_c
atp_c
atp_c
pyr_c
h_c
nad_c
LDH_D
nadh_c
lac__D_c
(C) Isozyme Fractional Abundances
Growth Condition: 🟩 Glucose 🟦 Pyruvate
PFK1 forms
PFK2 forms
$[PFK]_{form}/[PFK]_{total}$
1
0
$PFK1_c$
$PFK1_c^{f6p}$
$PFK1_c^{f6p, atp}$
$PFK1_c^{fdp}$
$PFK1_c^{fdp, adp}$
$PFK1_c^{pep}$
$PFK2_c$
$PFK2_c^{f6p}$
$PFK2_c^{fdp}$
$PFK2_c^{f6p, atp}$
$PFK2_c^{fdp, adp}$
FBP1 forms
FBP2 forms
$[FBP]_{form}/[FBP]_{total}$
1
0
$FBP1_c$
$FBP1_c^{f6p}$
$FBP1_c^{fdp}$
$FBP1_c^{f6p, pi}$
$FBP2_c$
$FBP2_c^{f6p}$
$FBP2_c^{fdp}$
$FBP2_c^{f6p, pi}$
FBA1 forms
FBA2 forms
$[FBA]_{form}/[FBA]_{total}$
1
0
$FBA1_c$
$FBA1_c^{dhap}$
$FBA1_c^{fdp}$
$FBA1_c^{dhap, g3p}$
$FBA2_c$
$FBA2_c^{dhap}$
$FBA2_c^{fdp}$
$FBA2_c^{dhap, g3p}$
PGMd forms
PGMi forms
$[PGM]_{form}/[PGM]_{total}$
1
0
$PGMd_c$
$PGMd_c^{2pg}$
$PGMd_c^{3pg}$
$PGMi_c$
$PGMi_c^{2pg}$
$PGMi_c^{3pg}$
PYK1 forms
PYK2 forms
$[PYK]_{form}/[PYK]_{total}$
1
0
$PYK1_c$
$PYK1_c^{adp}$
$PYK1_c^{atp}$
$PYK1_c^{pep, adp}$
$PYK1_c^{pyr, atp}$
$PYK2_c$
$PYK2_c^{adp}$
$PYK2_c^{atp}$
$PYK2_c^{pep, adp}$
$PYK2_c^{pyr, atp}$

Figure 4