Pacific Northwest
NATIONAL LABORATORY

PNNL-36693

Multi-scale Simulation, Calibration, and Optimization of Calcium Carbonate Precipitation in Microbial Communities

September 2024

August George
Winston Anthony
Andrew McNaughton
Tesia Lin
Victor Torres-Rodriguez
Shant Mahserejian
Sabina Altus
Yuliya Farris
Ryan McClure
Jeremy Zucker

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-36693

Multi-scale Simulation, Calibration, and Optimization of Calcium Carbonate Precipitation in Microbial Communities

September 2024

August George
Winston Anthony
Andrew McNaughton
Tesia Lin
Victor Torres-Rodriguez
Shant Mahserejian
Sabina Altus
Yuliya Farris
Ryan McClure
Jeremy Zucker

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

Ensuring the efficient engineering of microbially induced calcium carbonate precipitation (MICP) is crucial for a variety of environmental and civil engineering applications, such as soil stabilization and carbon sequestration. Addressing this need, we present a comprehensive multi-scale workflow that begins with the isolation of calcium carbonate-producing microbes from soil samples, followed by metagenomic sequencing and metabolic reconstruction. We then characterize microbial growth phenotypes under diverse nutrient conditions and compare observed growth with metabolic model predictions. Furthermore, we analyze metabolite consumption and production, and develop a consumer-resource model that is calibrated using time-series measurements of metabolites.

The primary benefit of our approach lies in its ability to predict and control MICP outcomes, facilitated by a Bayesian methodology that incorporates priors on initial conditions and parameters. This allows us to compute posteriors by integrating experimental data, and to solve a risk optimization under uncertainty (ROUU) problem to identify nutrient conditions that maximize calcium carbonate production. In contrast to non-Bayesian methods, which fail to quantify uncertainty accurately, our approach provides a more reliable pathway to optimizing nutrient conditions, enhancing the likelihood of achieving desired MICP outcomes. This positions our method as a superior alternative in the quest to improve MICP through engineered microbial consortia.

Summary

Project Overview

The CarbStor project aims to optimize microbially induced calcium carbonate precipitation (MICP), enhancing its applications in environmental and civil engineering, such as soil stabilization and carbon sequestration. This is achieved through a comprehensive multi-scale workflow that incorporates state-of-the-art techniques in microbial isolation, genomic sequencing, metabolic reconstruction, and model refinement.

Key Processes and Methodologies

1. **Microbial Isolation and Filtering:**
    - Soil samples are collected and processed to isolate calcium carbonate-producing microbes.
    - Progressive filtration techniques are used to enrich the microbial community with effective calcium carbonate producers, assessed based on their metabolic profiles and enzymatic activities.
2. **Metagenomic Sequencing and Assembly:**
    - High-throughput sequencing using PacBio and Illumina platforms generates extensive genetic data.
    - Sequences are assembled into high-quality genome assemblies, which are then annotated to identify metabolic pathways.
3. **Metabolic Reconstruction and Refinement:**
    - Genome-scale metabolic models (GEMs) are constructed and refined using the Consistent Reproduction of Phenotype (CROP) algorithm.
    - Phenotypic data from Biolog microarrays validate and inform these models, ensuring they accurately reflect microbial capabilities.
4. **Consumer-Resource Model Development:**
    - Detailed metabolite analysis informs the development of consumer-resource models, simulating microbial interactions and dynamics.
    - Time-series data of metabolites are used to calibrate these models.
5. **Model calibration using Bayesian Methodologies:**
    - Bayesian frameworks quantify uncertainty and integrate data.
    - This probabilistic framework is readily extendible to risk optimization for MICP.

Innovative Tools and Resources

- **Concerto:** A versatile tool for representing microbial community models at both genome-scale and consumer-resource scales, essential for detailed metabolic analysis and high-level community behavior simulation.

- **Organism-specific Pathway/Genome Databases:** Comprehensive genomic and pathway information for each microbe, facilitating accurate metabolic model construction.

Impact and Future Directions

The integration of these advanced methodologies and tools significantly enhances the ability to model and predict MICP outcomes. By providing a scientifically rigorous and practically applicable framework, the CarbStor project positions itself as a superior alternative to conventional methods, increasing the likelihood of achieving effective MICP for soil stabilization and carbon sequestration. Future work will apply Consistent Reproduction of Phenotype (CROP) to refine Genome-scale metabolic models (GEM)s by ensuring they consistently reproduce observed phenotypes, enhancing model accuracy and predictive power. We will also focus on integrating genome-scale, consumer-resource, and precipitation models, incorporating experimental data for calibration, and addressing numerical stability issues in solvers. This ongoing research will continue to refine and enhance MICP strategies, paving the way for more efficient and sustainable engineering and environmental solutions

Acknowledgments

This research was supported by the Predictive Phenomics Initiative, under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Acronyms and Abbreviations

CarbStor:	Calcium-Carbonate Storage
CROP:	Consistent Reproduction of Phenotype
PGDB:	Pathway-Genome Database
GEM:	Genome-Scale Metabolic Model
MEMOTE:	Metabolic Model Tests
MICP:	Microbial-Induced Calcium Carbonate Precipitation
ROUU:	Risk optimization under uncertainty
CaCO3:	Calcium carbonate

Contents

Figures

Tables

1.0 Introduction

1.1 Background on Calcium Carbonate Production by Microbial Communities

Calcium carbonate ($CaCO_3$) production by microbial communities, commonly referred to as microbially induced calcium carbonate precipitation (MICP), is a naturally occurring process dependent on environmental settings (Wang, et al. 2024). MICP occurs when microbes, through their metabolic activities in combination with the environment, create conditions favorable for the precipitation of calcium carbonate. These conditions can involve altering local pH, producing nucleation sites that facilitate $CaCO_3$ formation, or producing carbonate ions. The process is primarily driven by a variety of bacterial species, such as *Pseudomonas sp.* and *Bacillus cereus*, which can induce carbonate precipitation through ureolysis, denitrification, or sulfate reduction.

1.2 Significance and Applications of Microbial-Induced Calcium Carbonate Precipitation

The ability to harness and control MICP has significant implications for a wide range of engineering and environmental applications (Wang, et al. 2024). In civil engineering, MICP can be used to reinforce the mechanical properties and stability of soil. In environmental science, MICP presents a method for carbon sequestration, wherein CO_2 is permanently trapped in the form of stable calcium carbonate minerals, thus mitigating the effects of climate change. Furthermore, MICP can be used as a bio-consolidation technique to repair and preserve deteriorating stonework without the use of harmful chemicals.

1.3 Objectives and Scope of the Report

This report aims to detail a comprehensive workflow designed to engineer and optimize MICP using microbial communities. Our objectives include isolating and characterizing calcium carbonate-producing microbes, reconstructing and refining their metabolic models, and developing a predictive control strategy for optimizing MICP processes. (See **Figure 1**). Key steps in this workflow involve:

1. Isolating microbes from soil samples and filtering out non-contributing strains.
2. Using metagenomic sequencing to identify the genomes of retained microbes.
3. Reconstructing metabolic models from these genomes.
4. Characterizing microbial growth under various nutrient conditions using Biolog arrays.
5. Comparing observed growth phenotypes with model predictions and refining models using the CROP algorithm.

6. Measuring metabolite dynamics to understand consumption and exchange among microbes.

7. Developing and calibrating a consumer-resource model to simulate microbial interactions and growth.

8. Optimizing initial nutrient conditions to maximize calcium carbonate production.

Through this approach, we leverage Bayesian methodologies to quantify uncertainty and improve prediction accuracy, providing a robust framework for enhancing MICP. By addressing these objectives, the report aims to offer a scientifically rigorous pathway for advancing the applications of MICP in both engineering and environmental contexts.

Methods

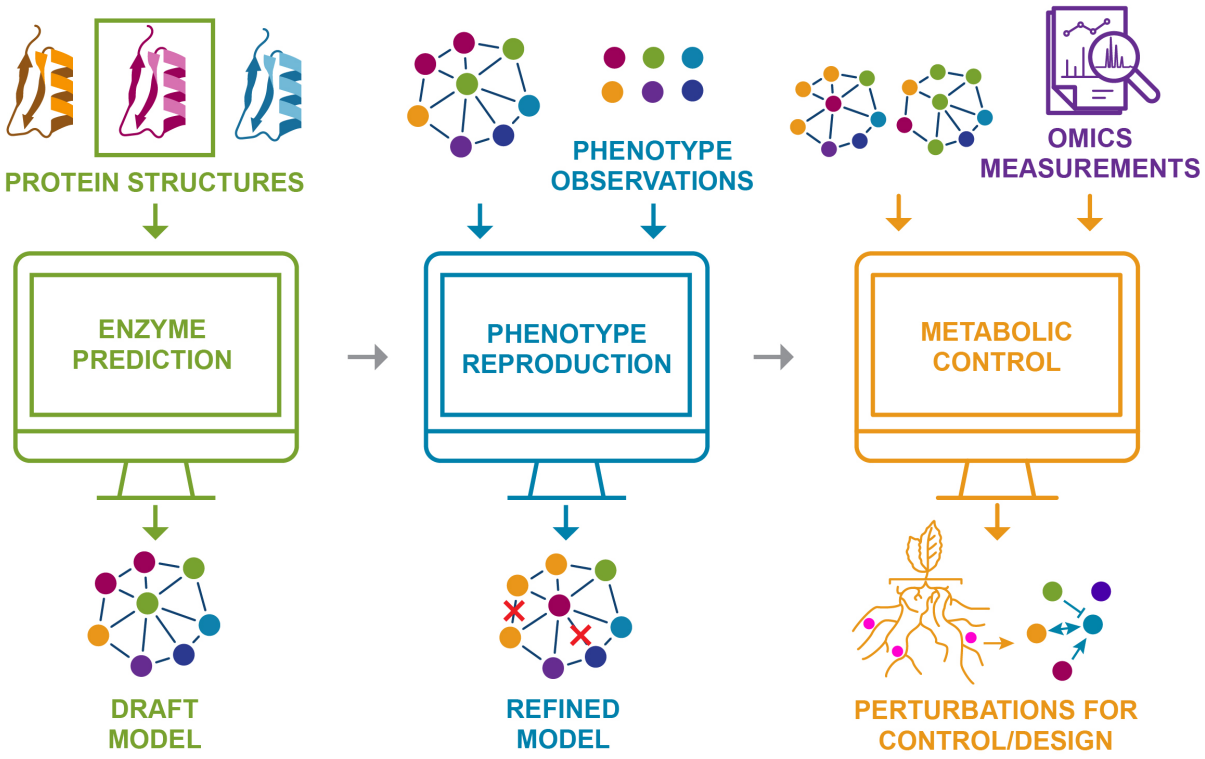

Figure 1: Concerto CarbStor Workflow. Protein structure and sequence is used to predict enzyme function for a draft metabolic reconstruction. Phenotype arrays are used as a training set to refine the model. Multi-omics measurements in time series are used to calibrate the model and design nutrient media to improve calcium carbonate production.

1.4 Soil Sample Preparation and Microbial Filtering

The initial phase involves collecting soil samples with communities of microbes known for their potentials to precipitate calcium carbonate. These samples undergo serial dilutions to simplify the input community, and sequential passaging to enrich for microbes that significantly contribute to calcium carbonate production. Enrichment begins with broad-spectrum microbial cultivation techniques followed by more selective culturing conditions that favor maximizing calcium carbonate production. Microbes were further assessed and selected for experimental testing based on criteria such as their metabolic profiles, growth rates under different pH conditions, presence of enzymes like urease that facilitate calcium carbonate precipitation, and stability within a calcium carbonate-producing community of microbes (Zelezniak, et al. 2015).

1.5 Metagenomic Sequencing and Assembly

To identify and assemble the genomes of the contributing microbes, we performed metagenomic sequencing on the microbial community retained from the filtration process. We used the PacBio High-throughput sequencing platforms to generate long-read sequencing data and Illumina to generate short read shotgun whole genome sequence data (Li and Durbin 2008) (Eid, et al. 2009). Initial steps involved quality control of raw reads using fastqc to filter low quality short reads, filtlong to filter low quality long reads, and bbmap to filter sequences to an even sequence depth. Clean reads were then assembled into contiguous sequences (contigs) using software like SPAdes (for short read only) or Unicycler (for hybrid long read/short read assemblies) (Wick, et al. 2017) (Bankevich, et al. 2012). This provided high quality genome assemblies ready for annotation by Prokka (Seemann 2014). Genome taxonomic identification was conducted using GTDB-TK, and assembly quality was assessed using checkM (Chaumeil, et al. 2022) (Parks, et al. 2015).

1.6 Metabolic Reconstruction

The assembled genomes were used to generate draft genome metabolic models (GEMs) using CARVEME, a specialized tool for metabolic reconstruction (Machado, et al. 2018). CARVEME integrates genomic data with biochemical databases to generate detailed models that map out the reactions and pathways specific to each microbe's metabolism. Initial quality was assessed using the MEMOTE validator (Lieven, et al. 2018). Using a custom python script, we utilized the COBRApy package to block reactions that produce unrealistic amounts of energy molecules, ensure that each model can produce a growth rate in a simulated LB media environment, annotate metabolites, genes, and reactions, fix unbalanced hydrogen charges in reactions, and using FASTCC to produce a consistent GEM network by removing blocked reactions and erroneous energy-generating cycles. This produces the most accurate network that results in the fewest similar optima for multiple states. These models were then checked against the MEMOTE validator a second time to ensure that they all generated an average MEMOTE score of over 90% (a similar level to the E. coli reference GEM). These models are characterized by their detailed representation of core metabolic reactions, essential growth pathways, and reactions potentially involved in calcium carbonate production.

1.7 Phenotypic Characterization Using Biolog Arrays

Empirical data on microbial growth under various nutrient conditions are gathered using Biolog microarrays (Bochner, Gadzinkski and Panomitros 2001). Each microbe species is subjected to multiple nutrient sources, allowed to grow for 96 hours, and optical density measurements taken as a proxy for growth phenotypes observed. This phenotypic characterization provides critical empirical data that validate and inform the metabolic models. The gathered data include growth rates, metabolic activity indicators, and substrate utilization profiles.

1.8 Comparison of Observed and Predicted Growth Phenotypes

We compare the observed growth data from Biolog arrays with the predicted growth phenotypes derived from the metabolic models to identify discrepancies. This comparison is crucial for understanding the accuracy and limitations of the initial models. Discrepancies guide subsequent refining efforts to enhance the fidelity of the models in predicting actual microbial behavior.

1.9 Model Refinement Using the CROP Algorithm

We apply the Consistent Reproduction of Phenotype (CROP) algorithm to refine the metabolic models (Dreyfuss, et al. 2013). The CROP algorithm adjusts the GEMs by adding or removing metabolic reactions, thus improving the consistency between predicted and observed growth phenotypes. This iterative refinement process ensures that the metabolic models accurately reflect the real metabolic capabilities and interactions within the microbial community.

1.10 Metabolite Measurement and Analysis

Changes in metabolite concentrations for each organism both individually and within the community are measured to determine metabolite consumption, production, and exchange. Techniques like mass spectrometry or nuclear magnetic resonance (NMR) spectroscopy are employed for precise measurement. This detailed analysis of metabolite dynamics is crucial for understanding microbial interactions and informs the development of accurate consumer-resource models.

1.11 Consumer-Resource Model Development

Using the insights gained from metabolite analysis, we develop a consumer-resource model focusing on metabolite exchange among organisms, growth rates, and enzyme-catalyzed reactions. SMETANA is used to assess the metabolic interactions within the community, identifying potential metabolic exchanges and dependencies (Zelezniak, et al. 2015). Microbially

induced calcium carbonate precipitation (MICP) reactions were obtained from the literature (Wang, et al. 2024). The kinetic model is constructed in a human-readable format using Antimony (Smith 2024). This model serves to simulate microbial interactions and predict calcium carbonate production dynamics.

1.12 Time-Series Measurement and Model Calibration

Time-series measurements of microbial growth rates, pH, and calcium carbonate precipitation within the community are conducted. These measurements are used to calibrate the consumer-resource model, applying Michaelis-Menten enzyme kinetics, and mass-balance for spontaneous reactions. This calibration ensures that the model accurately predicts outcomes based on the observed data.

1.13 Optimization of Nutrient Conditions

To maximize calcium carbonate production, we optimize the initial conditions and nutrient compositions using risk optimization frameworks under uncertainty. This step involves developing strategies that incorporate probabilistic analyses and optimization algorithms to identify nutrient conditions that enhance microbial-induced calcium carbonate precipitation. The final optimized conditions are validated experimentally to confirm their efficacy. (See **Figure 6**).

By systematically executing and connecting these steps, this workflow ensures a scientifically rigorous and practically applicable pathway to optimizing microbial-induced calcium carbonate precipitation. Each step is critical in building a comprehensive understanding of the metabolic interactions within microbial communities, ultimately facilitating effective MICP optimization for various engineering and environmental applications.

2.0 Results

2.1 Development of specialized tools

To support the complex tasks involved in modeling and optimizing microbial-induced calcium carbonate precipitation (MICP), we developed specialized tools: Concerto, Organism-specific Pathway/Genome Databases, the Consistent Reproduction of Phenotype (CROP) algorithms, and the Memote continuous validation environment.

2.1.1 Concerto

- **Purpose and Functionality**: Concerto is a versatile tool designed to represent microbial community models at both the genome-scale and consumer-resource scale. It integrates and harmonizes diverse data types to construct comprehensive models that can simulate microbial interactions and metabolic exchanges within a community. It allows for the quick creation of MEMOTE repositories for version control of new genome-scale models (Lieven, et al. 2018).

- **Application**: By using Concerto, we can efficiently map out genome-scale metabolic pathways and transition smoothly to consumer-resource models that reflect metabolic exchanges and community dynamics. This tool is key for both detailed metabolic analysis and high-level community behavior simulation.

- **Availability**: The tool is open-source and available for the research community at the Concerto GitHub Repository.

2.1.2 Organism-specific Pathway/Genome Databases (PGDBs)

- **Purpose and Functionality**: These databases provide detailed genome and pathway information specific to each microbe within the studied community. They serve as a crucial resource for model construction, allowing us to annotate genomes accurately and identify pertinent metabolic pathways without programming knowledge. They are further useful for integrating data with prior biological knowledge.

- **Application**: These databases facilitate the accurate reconstruction of genome-scale metabolic models by providing detailed information on the metabolic capabilities and pathways of each organism. More specifically, pathway/genome databases (PGDBs) integrate information using the Pathway Tools Pathologic metabolic reconstruction software in a schema consistent with other databases as hosted by Biocyc (Karp 2001). Additionally, they support the refinement and validation processes by offering a reliable reference for genomic and pathway data.

- **Availability:** The organism-specific PGDBs are accessible for public use through PNNL's Instance of Biocyc (see individual "PGDBs" hyperlinks in **Error! Reference source not found.**).

2.1.3 Consistent Reproduction of Phenotype (CROP)

- **Purpose and Functionality:** CROP is an innovative tool designed to refine metabolic models by ensuring that they consistently reproduce observed phenotypes. CROP achieves this by systematically adding or removing reactions from the models.

- **Application:** By using CROP, we enhance the consistency between observed growth phenotypes and model predictions. This iterative refinement process ensures that our models are robust and have high predictive accuracy for microbial behavior under various conditions. CROP has been tested and validated on small models but has not yet been successfully applied to improve the accuracy of the CarbStor community models.

- **Availability:** The CROP tool is open-source and available at CROP GitHub Repository. We also implemented the fast gap-filling algorithm in CARVEME to improve its ability to efficiently add reactions to a model to make its predicted growth phenotypes consistent with observed growth phenotypes

2.1.4 Continuous validation with MEMOTE

A critical aspect of ensuring the reliability and accuracy of our genome-scale metabolic models (GEMs) in the CarbStor community involved the application of the MEMOTE continuous validation environment. MEMOTE an automated testing framework, was employed to systematically track and document all curations and modifications made to the GEMs of the five key microbial strains in the study. By integrating MEMOTE into our workflow, we were able to perform high-throughput, reproducible quality checks each time a change was made to the models. This not only streamlined the curation process but also provided immediate feedback on whether these changes enhanced the models' predictive accuracy. After each curation, MEMOTE ran a comprehensive suite of tests to assess model performance, including the ability to predict microbial growth phenotypes under varying conditions. This continuous validation ensured that each modification led to a tangible improvement in model accuracy, thereby fostering a cycle of refinement and validation that significantly enhanced our confidence in the predictive capabilities of the GEMs. (See **Table 1** below for links to the MEMOTE repositories).

Table 1. Summary of content included in organism-specific pathway-genome databases (PGDBs) and genome-scale metabolic models (GEM) repositories for each of the five microbes. PGDBs were generated using the Pathway Tools Pathologic metabolic reconstruction software, and the models in GEM repositories were generating using the Concerto modeling framework and further refined using the CROP tool as described in Section 2.1.

	Curtobacterium	*Microbacterium*	*Rhodococcus*	*Paenibacillus*	*Bacillus*
Genes:	3,650	3,807	6,015	6,584	5,948
Pathways:	299	260	408	354	374
Enzymatic Reactions:	1,705	1,461	2,436	2,073	2,186
Transport Reactions:	118	101	132	153	166
Polypeptides:	3,562	3,742	5,955	6,504	5,849
Protein Complexes:	51	124	85	84	71
Enzymes:	895	837	1,546	1,404	1,293

Transporters:	257	339	363	722	484
Compounds:	1,587	1,428	2,148	1,744	1,869
Transcription Units:	2,451	2,369	4,080	4,809	4,436
tRNAs:	77	47	56	72	81
GO Terms:	168	347	222	472	281
PGDBs:	CSC009 Database	CSC031 Database	CSC040 Database	CSC043 Database	CSC052 Database
MEMOTE Repositories:	CSC009 GEM	CSC031 GEM	CSC040 GEM	CSC043 GEM	CSC052 GEM

2.2 Calibration and Simulation of Consumer-Resource Models

In addition to developing specialized tools, we performed model calibration and simulation of consumer-resource models. This multi-step process was essential for accurately capturing the interactions and metabolic exchanges within the microbial community and predicting conditions for maximum calcium carbonate production.

2.2.1 Microbially-induced calcium carbonate precipitation model

The model was created using Antimony (Smith 2024), with the graphical petri-net and chemical reaction representations shown below. The Antimony file is available in the Appendix. This human-readable format is interoperable with the systems biology markup language (SBML) (Keating 2020) that allows for time-series simulation, calibration, and optimization.

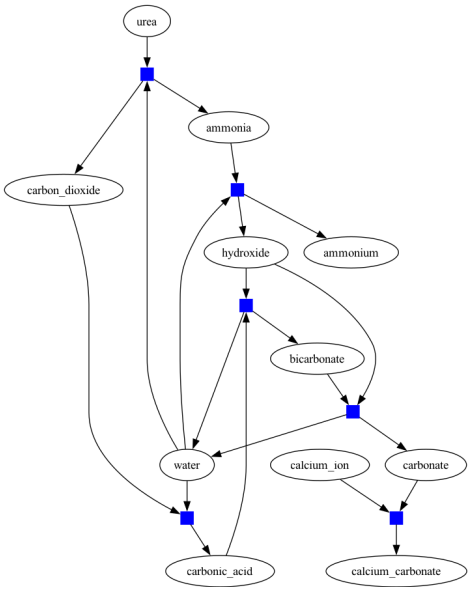

Figure 2: Petri net representation of the microbially induced calcium carbonate precipitation model. Blue rectangles represent transitions and ovals represent states.

Reaction 1: Ureolysis (irreversible)	$CO(NH_2)_2$ (urea) $+ H_2O \longrightarrow 2NH_3 + CO_2$
Reaction 2: Protonation of Ammonia	$2NH_3 + 2H_2O \rightleftharpoons 2NH_4^+ + 2OH^-$
Reaction 3: CO_2 Hydrolysis (irreversible)	$CO_2 + H_2O \longrightarrow H_2CO_3$
Reaction 4: Dissociation of Carbonic Acid	$H_2CO_3 + OH^- \rightleftharpoons HCO_3^- + H_2O$
Reaction 5: Dissociation of Bicarbonate	$HCO_3^- + OH^- \rightleftharpoons CO_3^{2-} + H_2O$
Reaction 6: $CaCO_3$ Precipitation/Dissolution	$CO_3^{2-} + Ca^{2+} \rightleftharpoons CaCO_3$

Figure 3: Microbially induced calcium carbonate precipitation chemical equations.

2.2.2 Consumer-resource model

The model was created using Antimony, with the graphical petri-net and Antimony file available in the Appendix. This human-readable format is interoperable with the systems biology markup language (SBML) that allows for time-series simulation, calibration, and optimization.

2.2.3 Calibration

- **Methodology:** Simulated data from time-series measurements of microbial metabolites and calcium carbonate precipitation were used to demonstrate our workflow to calibrate the consumer-resource models. Calibration involved adjusting model parameters to match observed data, ensuring that the models reflect the dynamics of the microbial community**.**

- **Outcome:** Calibrated models allowed for more precise predictions of community behavior under than uncalibrated models, providing a solid foundation for subsequent simulations and optimizations.

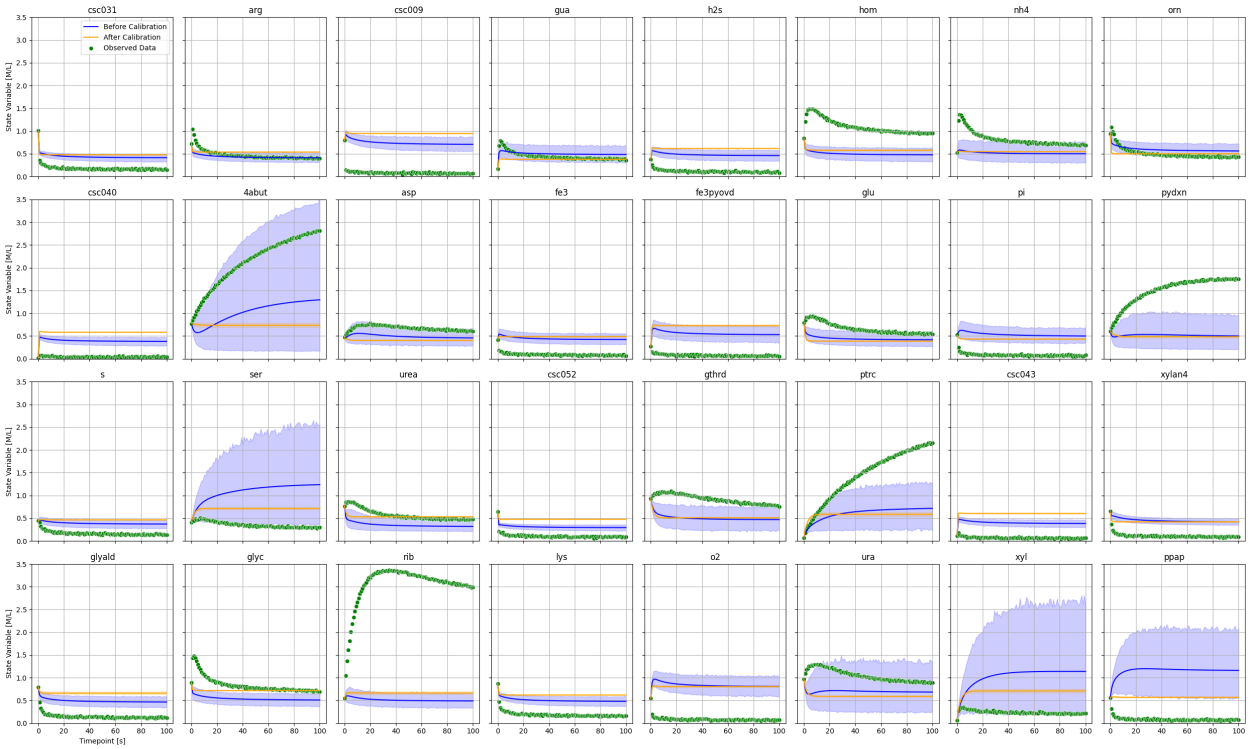

Figure 4: Calibration results for the consumer resource model. The before (blue) and after (orange) calibration outputs as well as the observed data (green) are shown for the components of the consumer resource model. The calibration significantly reduces the variance in predicted outputs. Due to numerical solver instability for the *in silico* parameter values chosen there is some discrepancy between the observed (simulated) data and the before and after calibration predications.

3.0 Discussion and Future Work

3.1 Impact on MICP Optimization

The development and utilization of these tools, along with the calibration and simulation of consumer-resource models, have significantly advanced our ability to model and predict MICP outcomes. These efforts have contributed to a deeper understanding of microbial interactions and metabolism, paving the way for more effective and optimized strategies for MICP. The integration of these tools and methodologies into our workflow provides a robust foundation for ongoing and future research efforts in this area, ensuring continued progress towards achieving optimal microbial-induced calcium carbonate precipitation.

3.2 Future directions

We plan to use risk optimization frameworks under uncertainty to identify initial conditions and nutrient compositions that maximize calcium carbonate production. This involves exploring a range of conditions and using optimization algorithms to find the best set of parameters. The optimized conditions will lead to significant improvements in calcium carbonate production, demonstrating the effectiveness of our approach.

The calibrated parameters obtained from consumer-resource models will be used to downscale to genome-scale community metabolic models. This downscaling process integrates high-level community interactions with detailed genome-scale metabolic pathways, ensuring that the GEMs accurately reflected the calibrated and optimized conditions observed at the community level. This seamless transition from consumer-resource models to GEMs will provide a comprehensive understanding of microbial metabolism and interactions, further enhancing our ability to predict and optimize calcium carbonate production. The current implementation uses separate consumer resource and calcium carbonate models, but future work will integrate genome scale, consumer resource, and precipitation models – as shown in the Appendix.

Due to limited available experimental data, simulated data was used to demonstrate the model calibration, simulation, and optimization workflow for CarbStor. Future work will incorporate experimental data as it becomes available. Furthermore, the models calibrated on experimental data will be used to simulate different scenarios, predicting the behavior of the microbial community under various nutrient conditions and environmental parameters. These simulations will help identify potential bottlenecks and interactions critical for calcium carbonate production. Due to numerical stability issues in the current solver used, there is a discrepancy between the observed data and the simulated data which will be addressed in future work. Finally, the simulation results will provide valuable insights into the metabolic dynamics and interactions within the community, guiding the optimization process towards achieving maximum efficiency and output.

4.0 Bibliography

Bochner, Barry R, Peter Gadzinkski, and Eugenia Panomitros. 2001. "Phenotype Microarrays for High-Throughput Phenotypic Testing and Assay of Gene Function." *Genome Research* 11 (7): 1246-55.

Dreyfuss, Jonathan M., Jeremy D. Zucker, Heather M. Hood, Linda R. Ocasio, Matthew S. Sachs, and James E. Galagan. 2013. "Reconstruction and Validation of a Genome-Scale Metabolic Model for the Filamentous Fungus Neurospora Crassa Using FARM." *PLoS Computational Biology* 9 (7): e1003126.

Karp, Peter D. 2001. "Pathway Databases: A Case Study in Computational Symbolic Theories." *Science* 293 (5537): 2040-44.

Keating, Sarah M, Dagmar Waltemath, Matthias König, Fengkai Zhang, Andreas Dräger, Claudine Chaouiya, Frank T Bergmann. 2020. "SBML Level 3: An Extensible Format for the Exchange and Reuse of Biological Models." *Molecular Systems Biology* (Molecular Systems) 16 (8): e9110. https://doi.org/10.15252/msb.20199110. .

Lieven, Christian, Moritz E. Beber, Brett G. Olivier, Frank T. Bergmann, Meric Ataman, Parizad Babaei, Jennifer A. Bartell, and et al. 2018. "Memote: A Community-Driven Effort towards a Standardized Genome-Scale Metabolic Model Test Suite." *BioRxiv.*

Machado, Daniel, Sergej Andrejev, Melanie Tramontano, and Kiran R. Patil. 2018. "Fast Automated Reconstruction of Genome-Scale Metabolic Models for Microbial Species and Communities." *BioRxiv.*

Seemann, Torsten. 2014. "Prokka: Rapid Prokaryotic Genome Annotation." *Bioinformatics* 30 (14): 2068-69.

Smith, Lucian, and Herbert M Sauro. 2024. "An Update to the SBML Human-Readable Antimony Language." *Arxiv* https://arxiv.org/abs/2405.15109.

Wang, Jianye, Helen Mitrani, Anil Wipat, Polly Moreland, Jamie Haystead, Meng Zhang, and Martyn D. Robertson. 2024. "A Numerical Bio-Geotechnical Model of Pressure-Responsive Microbially Induced Calcium Carbonate Precipitation." *Applied Sciences* 14 (7): 2854.

Zelezniak, Aleksej, Sergej Andrejev, Olga Ponomarova, Daniel R. Mende, Peer Bork, and Kiran R. Patil. 2015. "Metabolic Dependencies Drive Species Co-Occurrence in Diverse Microbial Communities." *Proceedings of the National Academy of Sciences of the United States of America* 112 (20): 6449-54.

Appendix A – Antimony Representations

A.1 Antimony representation of microbially-induced calcium carbonate precipitation:

```
antimony_model_str = """
model MICP

  const water;

  // Reaction 1: Ureolysis (irreversible)
  reaction_ureolysis: urea + $water => 2 ammonia + carbon_dioxide;
(V_max_urease*urea)/(K_m_urease + urea);  // Michaelis-Menten (water in excess and
fixed);

  // Reaction 2: Protonation of Ammonia
  reaction_protonation_NH3: 2 ammonia + 2 $water -> 2 ammonium + 2 hydroxide; k2f *
ammonia^2 * water^2 - k2r * ammonium^2 * hydroxide^2;

  // Reaction 3: CO2 Hydrolysis (irreversible)
  reaction_CO2_hydrolysis: carbon_dioxide + $water => carbonic_acid;
(K_cat_CA*CA_conc*carbon_dioxide)/(K_m_CA + carbon_dioxide);   // Michaelis-Menten;

  // Reaction 4: Dissociation of Carbonic Acid
  reaction_dissociation_H2CO3: carbonic_acid + hydroxide -> bicarbonate + $water; k4f
* carbonic_acid * hydroxide - k4r * bicarbonate * water;

  // Reaction 5: Dissociation of Bicarbonate
  reaction_dissociation_HCO3: bicarbonate + hydroxide -> carbonate + $water; k5f *
bicarbonate * hydroxide - k5r * carbonate * water;

  // Reaction 6: CaCO3 Precipitation/Dissolution
  reaction_CaCO3_precipitation: carbonate + calcium_ion -> calcium_carbonate; k6f *
carbonate * calcium_ion - k6r * calcium_carbonate;

  // Set initial concentrations for species
  urea = 0.33;             // Initial concentration for urea
  water = 55.0;            // Initial concentration for water
  ammonia = 1e-5;          // Initial concentration for ammonia
  carbon_dioxide = 1e-5;   // Initial concentration for carbon dioxide (from
atmosphere)
  ammonium = 1e-5;         // Initial concentration for ammonium
  hydroxide = 1e-7;        // Initial concentration for hydroxide
  carbonic_acid = 1e-5;    // Initial concentration for carbonic acid
  bicarbonate = 1e-5;      // Initial concentration for bicarbonate
  carbonate = 1e-5;        // Initial concentration for carbonate
  calcium_ion = 0.136;     // Initial concentration for calcium ion
  calcium_carbonate = 0;   // Initial concentration for calcium carbonate
```

```
  // Define rate constants
  k2f = 1;  // Rate constant for reaction 2 forward
  k2r = 1;  // Rate constant for reaction 2 reverse
  k4f = 1;  // Rate constant for reaction 4 forward
  k4r = 1;  // Rate constant for reaction 4 reverse
  k5f = 1;  // Rate constant for reaction 5 forward
  k5r = 1;  // Rate constant for reaction 5 reverse
  k6f = 1;  // Rate constant for reaction 6 forward
  k6r = 1;  // Rate constant for reaction 6 reverse

  // enzymatic reaction parameters
  V_max_urease = 1.8e4; // M/L/s (1100 mM/mg / min)
  K_m_urease= 0.3; // M (300 mM)
  K_cat_CA = 530000; // 1/s;
  CA_conc = 3e-5; // M/L (30 uM)
  K_m_CA = 12.9; // M (12900 mM)

end
"""
```

A.2 Antimony representation of the consumer resource model:

```
antimony_str = """
model smetana
    updated_csc031 -> M_arg__L_e; k_d0*updated_csc031
    M_arg__L_e -> updated_csc009; k_r0*M_arg__L_e
    updated_csc031 -> M_gua_e; k_d1*updated_csc031
    M_gua_e -> updated_csc009; k_r1*M_gua_e
    updated_csc031 -> M_h2s_e; k_d2*updated_csc031
    M_h2s_e -> updated_csc009; k_r2*M_h2s_e
    updated_csc031 -> M_hom__L_e; k_d3*updated_csc031
    M_hom__L_e -> updated_csc009; k_r3*M_hom__L_e
    updated_csc031 -> M_nh4_e; k_d4*updated_csc031
    M_nh4_e -> updated_csc009; k_r4*M_nh4_e
    updated_csc031 -> M_orn_e; k_d5*updated_csc031
    M_orn_e -> updated_csc009; k_r5*M_orn_e
    updated_csc040 -> M_4abut_e; k_d6*updated_csc040
    M_4abut_e -> updated_csc009; k_r6*M_4abut_e
    updated_csc040 -> M_arg__L_e; k_d7*updated_csc040
    M_arg__L_e -> updated_csc009; k_r7*M_arg__L_e
    updated_csc040 -> M_asp__L_e; k_d8*updated_csc040
    M_asp__L_e -> updated_csc009; k_r8*M_asp__L_e
    updated_csc040 -> M_fe3_e; k_d9*updated_csc040
    M_fe3_e -> updated_csc009; k_r9*M_fe3_e
    updated_csc040 -> M_fe3pyovd_kt_e; k_d10*updated_csc040
    M_fe3pyovd_kt_e -> updated_csc009; k_r10*M_fe3pyovd_kt_e
```

```
updated_csc040 -> M_glu__L_e; k_d11*updated_csc040
M_glu__L_e -> updated_csc009; k_r11*M_glu__L_e
updated_csc040 -> M_gua_e; k_d12*updated_csc040
M_gua_e -> updated_csc009; k_r12*M_gua_e
updated_csc040 -> M_h2s_e; k_d13*updated_csc040
M_h2s_e -> updated_csc009; k_r13*M_h2s_e
updated_csc040 -> M_hom__L_e; k_d14*updated_csc040
M_hom__L_e -> updated_csc009; k_r14*M_hom__L_e
updated_csc040 -> M_nh4_e; k_d15*updated_csc040
M_nh4_e -> updated_csc009; k_r15*M_nh4_e
updated_csc040 -> M_orn_e; k_d16*updated_csc040
M_orn_e -> updated_csc009; k_r16*M_orn_e
updated_csc040 -> M_pi_e; k_d17*updated_csc040
M_pi_e -> updated_csc009; k_r17*M_pi_e
updated_csc040 -> M_pydxn_e; k_d18*updated_csc040
M_pydxn_e -> updated_csc009; k_r18*M_pydxn_e
updated_csc040 -> M_s_e; k_d19*updated_csc040
M_s_e -> updated_csc009; k_r19*M_s_e
updated_csc040 -> M_ser__L_e; k_d20*updated_csc040
M_ser__L_e -> updated_csc009; k_r20*M_ser__L_e
updated_csc040 -> M_urea_e; k_d21*updated_csc040
M_urea_e -> updated_csc009; k_r21*M_urea_e
updated_csc052 -> M_arg__L_e; k_d22*updated_csc052
M_arg__L_e -> updated_csc009; k_r22*M_arg__L_e
updated_csc052 -> M_fe3_e; k_d23*updated_csc052
M_fe3_e -> updated_csc009; k_r23*M_fe3_e
updated_csc052 -> M_glu__L_e; k_d24*updated_csc052
M_glu__L_e -> updated_csc009; k_r24*M_glu__L_e
updated_csc052 -> M_gthrd_e; k_d25*updated_csc052
M_gthrd_e -> updated_csc009; k_r25*M_gthrd_e
updated_csc052 -> M_h2s_e; k_d26*updated_csc052
M_h2s_e -> updated_csc009; k_r26*M_h2s_e
updated_csc052 -> M_nh4_e; k_d27*updated_csc052
M_nh4_e -> updated_csc009; k_r27*M_nh4_e
updated_csc052 -> M_orn_e; k_d28*updated_csc052
M_orn_e -> updated_csc009; k_r28*M_orn_e
updated_csc052 -> M_ptrc_e; k_d29*updated_csc052
M_ptrc_e -> updated_csc009; k_r29*M_ptrc_e
updated_csc052 -> M_s_e; k_d30*updated_csc052
M_s_e -> updated_csc009; k_r30*M_s_e
updated_csc052 -> M_urea_e; k_d31*updated_csc052
M_urea_e -> updated_csc009; k_r31*M_urea_e
csc043 -> M_arg__L_e; k_d32*csc043
M_arg__L_e -> updated_csc009; k_r32*M_arg__L_e
csc043 -> M_gua_e; k_d33*csc043
M_gua_e -> updated_csc009; k_r33*M_gua_e
csc043 -> M_h2s_e; k_d34*csc043
M_h2s_e -> updated_csc009; k_r34*M_h2s_e
```

```
csc043 -> M_nh4_e; k_d35*csc043
M_nh4_e -> updated_csc009; k_r35*M_nh4_e
csc043 -> M_s_e; k_d36*csc043
M_s_e -> updated_csc009; k_r36*M_s_e
csc043 -> M_xylan4_e; k_d37*csc043
M_xylan4_e -> updated_csc009; k_r37*M_xylan4_e
updated_csc009 -> M_fe3_e; k_d38*updated_csc009
M_fe3_e -> updated_csc031; k_r38*M_fe3_e
updated_csc009 -> M_fe3pyovd_kt_e; k_d39*updated_csc009
M_fe3pyovd_kt_e -> updated_csc031; k_r39*M_fe3pyovd_kt_e
updated_csc009 -> M_glyald_e; k_d40*updated_csc009
M_glyald_e -> updated_csc031; k_r40*M_glyald_e
updated_csc009 -> M_glyc_e; k_d41*updated_csc009
M_glyc_e -> updated_csc031; k_r41*M_glyc_e
updated_csc009 -> M_rib__D_e; k_d42*updated_csc009
M_rib__D_e -> updated_csc031; k_r42*M_rib__D_e
updated_csc040 -> M_fe3_e; k_d43*updated_csc040
M_fe3_e -> updated_csc031; k_r43*M_fe3_e
updated_csc040 -> M_fe3pyovd_kt_e; k_d44*updated_csc040
M_fe3pyovd_kt_e -> updated_csc031; k_r44*M_fe3pyovd_kt_e
updated_csc040 -> M_glyc_e; k_d45*updated_csc040
M_glyc_e -> updated_csc031; k_r45*M_glyc_e
updated_csc040 -> M_pi_e; k_d46*updated_csc040
M_pi_e -> updated_csc031; k_r46*M_pi_e
updated_csc052 -> M_fe3_e; k_d47*updated_csc052
M_fe3_e -> updated_csc031; k_r47*M_fe3_e
updated_csc052 -> M_glyald_e; k_d48*updated_csc052
M_glyald_e -> updated_csc031; k_r48*M_glyald_e
updated_csc052 -> M_glyc_e; k_d49*updated_csc052
M_glyc_e -> updated_csc031; k_r49*M_glyc_e
updated_csc052 -> M_rib__D_e; k_d50*updated_csc052
M_rib__D_e -> updated_csc031; k_r50*M_rib__D_e
csc043 -> M_glyald_e; k_d51*csc043
M_glyald_e -> updated_csc031; k_r51*M_glyald_e
csc043 -> M_glyc_e; k_d52*csc043
M_glyc_e -> updated_csc031; k_r52*M_glyc_e
csc043 -> M_rib__D_e; k_d53*csc043
M_rib__D_e -> updated_csc031; k_r53*M_rib__D_e
csc043 -> M_xylan4_e; k_d54*csc043
M_xylan4_e -> updated_csc031; k_r54*M_xylan4_e
updated_csc009 -> M_arg__L_e; k_d55*updated_csc009
M_arg__L_e -> updated_csc040; k_r55*M_arg__L_e
updated_csc009 -> M_fe3_e; k_d56*updated_csc009
M_fe3_e -> updated_csc040; k_r56*M_fe3_e
updated_csc009 -> M_fe3pyovd_kt_e; k_d57*updated_csc009
M_fe3pyovd_kt_e -> updated_csc040; k_r57*M_fe3pyovd_kt_e
updated_csc009 -> M_hom__L_e; k_d58*updated_csc009
M_hom__L_e -> updated_csc040; k_r58*M_hom__L_e
```

```
updated_csc009 -> M_lys__L_e; k_d59*updated_csc009
M_lys__L_e -> updated_csc040; k_r59*M_lys__L_e
updated_csc009 -> M_nh4_e; k_d60*updated_csc009
M_nh4_e -> updated_csc040; k_r60*M_nh4_e
updated_csc009 -> M_o2_e; k_d61*updated_csc009
M_o2_e -> updated_csc040; k_r61*M_o2_e
updated_csc009 -> M_orn_e; k_d62*updated_csc009
M_orn_e -> updated_csc040; k_r62*M_orn_e
updated_csc031 -> M_arg__L_e; k_d63*updated_csc031
M_arg__L_e -> updated_csc040; k_r63*M_arg__L_e
updated_csc031 -> M_gua_e; k_d64*updated_csc031
M_gua_e -> updated_csc040; k_r64*M_gua_e
updated_csc031 -> M_hom__L_e; k_d65*updated_csc031
M_hom__L_e -> updated_csc040; k_r65*M_hom__L_e
updated_csc031 -> M_lys__L_e; k_d66*updated_csc031
M_lys__L_e -> updated_csc040; k_r66*M_lys__L_e
updated_csc031 -> M_nh4_e; k_d67*updated_csc031
M_nh4_e -> updated_csc040; k_r67*M_nh4_e
updated_csc031 -> M_orn_e; k_d68*updated_csc031
M_orn_e -> updated_csc040; k_r68*M_orn_e
updated_csc052 -> M_arg__L_e; k_d69*updated_csc052
M_arg__L_e -> updated_csc040; k_r69*M_arg__L_e
updated_csc052 -> M_fe3_e; k_d70*updated_csc052
M_fe3_e -> updated_csc040; k_r70*M_fe3_e
updated_csc052 -> M_lys__L_e; k_d71*updated_csc052
M_lys__L_e -> updated_csc040; k_r71*M_lys__L_e
updated_csc052 -> M_nh4_e; k_d72*updated_csc052
M_nh4_e -> updated_csc040; k_r72*M_nh4_e
updated_csc052 -> M_orn_e; k_d73*updated_csc052
M_orn_e -> updated_csc040; k_r73*M_orn_e
csc043 -> M_arg__L_e; k_d74*csc043
M_arg__L_e -> updated_csc040; k_r74*M_arg__L_e
csc043 -> M_gua_e; k_d75*csc043
M_gua_e -> updated_csc040; k_r75*M_gua_e
csc043 -> M_lys__L_e; k_d76*csc043
M_lys__L_e -> updated_csc040; k_r76*M_lys__L_e
csc043 -> M_nh4_e; k_d77*csc043
M_nh4_e -> updated_csc040; k_r77*M_nh4_e
csc043 -> M_ura_e; k_d78*csc043
M_ura_e -> updated_csc040; k_r78*M_ura_e
updated_csc009 -> M_glyc_e; k_d79*updated_csc009
M_glyc_e -> updated_csc052; k_r79*M_glyc_e
updated_csc009 -> M_h2s_e; k_d80*updated_csc009
M_h2s_e -> updated_csc052; k_r80*M_h2s_e
updated_csc009 -> M_o2_e; k_d81*updated_csc009
M_o2_e -> updated_csc052; k_r81*M_o2_e
updated_csc031 -> M_glyc_e; k_d82*updated_csc031
M_glyc_e -> updated_csc052; k_r82*M_glyc_e
```

```
updated_csc031 -> M_h2s_e; k_d83*updated_csc031
M_h2s_e -> updated_csc052; k_r83*M_h2s_e
updated_csc031 -> M_xyl__D_e; k_d84*updated_csc031
M_xyl__D_e -> updated_csc052; k_r84*M_xyl__D_e
updated_csc040 -> M_glyc_e; k_d85*updated_csc040
M_glyc_e -> updated_csc052; k_r85*M_glyc_e
updated_csc040 -> M_h2s_e; k_d86*updated_csc040
M_h2s_e -> updated_csc052; k_r86*M_h2s_e
updated_csc040 -> M_pi_e; k_d87*updated_csc040
M_pi_e -> updated_csc052; k_r87*M_pi_e
updated_csc040 -> M_s_e; k_d88*updated_csc040
M_s_e -> updated_csc052; k_r88*M_s_e
csc043 -> M_glyc_e; k_d89*csc043
M_glyc_e -> updated_csc052; k_r89*M_glyc_e
csc043 -> M_h2s_e; k_d90*csc043
M_h2s_e -> updated_csc052; k_r90*M_h2s_e
csc043 -> M_s_e; k_d91*csc043
M_s_e -> updated_csc052; k_r91*M_s_e
csc043 -> M_xylan4_e; k_d92*csc043
M_xylan4_e -> updated_csc052; k_r92*M_xylan4_e
updated_csc009 -> M_arg__L_e; k_d93*updated_csc009
M_arg__L_e -> csc043; k_r93*M_arg__L_e
updated_csc009 -> M_h2s_e; k_d94*updated_csc009
M_h2s_e -> csc043; k_r94*M_h2s_e
updated_csc009 -> M_lys__L_e; k_d95*updated_csc009
M_lys__L_e -> csc043; k_r95*M_lys__L_e
updated_csc009 -> M_o2_e; k_d96*updated_csc009
M_o2_e -> csc043; k_r96*M_o2_e
updated_csc009 -> M_ppap_e; k_d97*updated_csc009
M_ppap_e -> csc043; k_r97*M_ppap_e
updated_csc031 -> M_arg__L_e; k_d98*updated_csc031
M_arg__L_e -> csc043; k_r98*M_arg__L_e
updated_csc031 -> M_h2s_e; k_d99*updated_csc031
M_h2s_e -> csc043; k_r99*M_h2s_e
updated_csc031 -> M_lys__L_e; k_d100*updated_csc031
M_lys__L_e -> csc043; k_r100*M_lys__L_e
updated_csc040 -> M_arg__L_e; k_d101*updated_csc040
M_arg__L_e -> csc043; k_r101*M_arg__L_e
updated_csc040 -> M_h2s_e; k_d102*updated_csc040
M_h2s_e -> csc043; k_r102*M_h2s_e
updated_csc040 -> M_lys__L_e; k_d103*updated_csc040
M_lys__L_e -> csc043; k_r103*M_lys__L_e
updated_csc040 -> M_s_e; k_d104*updated_csc040
M_s_e -> csc043; k_r104*M_s_e
updated_csc052 -> M_arg__L_e; k_d105*updated_csc052
M_arg__L_e -> csc043; k_r105*M_arg__L_e
updated_csc052 -> M_h2s_e; k_d106*updated_csc052
M_h2s_e -> csc043; k_r106*M_h2s_e
```

```
updated_csc052 -> M_lys__L_e; k_d107*updated_csc052
M_lys__L_e -> csc043; k_r107*M_lys__L_e
updated_csc052 -> M_s_e; k_d108*updated_csc052
M_s_e -> csc043; k_r108*M_s_e
k_d0=0.3333333333333333
k_r0=0.08
k_d1=0.3333333333333333
k_r1=0.1
k_d2=0.3333333333333333
k_r2=0.33
k_d3=0.3333333333333333
k_r3=0.07
k_d4=0.3333333333333333
k_r4=0.09
k_d5=0.3333333333333333
k_r5=0.03
k_d6=1.0
k_r6=0.01
k_d7=1.0
k_r7=0.08
k_d8=1.0
k_r8=0.06
k_d9=1.0
k_r9=0.39
k_d10=1.0
k_r10=0.61
k_d11=1.0
k_r11=0.06
k_d12=1.0
k_r12=0.1
k_d13=1.0
k_r13=0.33
k_d14=1.0
k_r14=0.07
k_d15=1.0
k_r15=0.09
k_d16=1.0
k_r16=0.03
k_d17=1.0
k_r17=0.49
k_d18=1.0
k_r18=0.02
k_d19=1.0
k_r19=0.33
k_d20=1.0
k_r20=0.12
k_d21=1.0
k_r21=0.07
```

```
k_d22=0.3333333333333333
k_r22=0.08
k_d23=0.3333333333333333
k_r23=0.39
k_d24=0.3333333333333333
k_r24=0.06
k_d25=0.3333333333333333
k_r25=0.04
k_d26=0.3333333333333333
k_r26=0.33
k_d27=0.3333333333333333
k_r27=0.09
k_d28=0.3333333333333333
k_r28=0.03
k_d29=0.3333333333333333
k_r29=0.01
k_d30=0.3333333333333333
k_r30=0.33
k_d31=0.3333333333333333
k_r31=0.07
k_d32=1.0
k_r32=0.08
k_d33=1.0
k_r33=0.1
k_d34=1.0
k_r34=0.33
k_d35=1.0
k_r35=0.09
k_d36=1.0
k_r36=0.33
k_d37=1.0
k_r37=1.0
k_d38=1.0
k_r38=1.0
k_d39=1.0
k_r39=0.89
k_d40=1.0
k_r40=0.48
k_d41=1.0
k_r41=0.06
k_d42=1.0
k_r42=0.02
k_d43=1.0
k_r43=1.0
k_d44=1.0
k_r44=0.89
k_d45=1.0
k_r45=0.06
```

```
k_d46=1.0
k_r46=1.0
k_d47=0.5
k_r47=1.0
k_d48=0.5
k_r48=0.48
k_d49=0.5
k_r49=0.06
k_d50=0.5
k_r50=0.02
k_d51=1.0
k_r51=0.48
k_d52=1.0
k_r52=0.06
k_d53=1.0
k_r53=0.02
k_d54=1.0
k_r54=0.44
k_d55=1.0
k_r55=0.06
k_d56=1.0
k_r56=0.15
k_d57=1.0
k_r57=0.79
k_d58=1.0
k_r58=0.04
k_d59=1.0
k_r59=0.16
k_d60=1.0
k_r60=0.05
k_d61=1.0
k_r61=0.93
k_d62=1.0
k_r62=0.18
k_d63=0.3333333333333333
k_r63=0.06
k_d64=0.3333333333333333
k_r64=0.19
k_d65=0.3333333333333333
k_r65=0.04
k_d66=0.3333333333333333
k_r66=0.16
k_d67=0.3333333333333333
k_r67=0.05
k_d68=0.3333333333333333
k_r68=0.18
k_d69=0.3333333333333333
k_r69=0.06
```

```
k_d70=0.3333333333333333
k_r70=0.15
k_d71=0.3333333333333333
k_r71=0.16
k_d72=0.3333333333333333
k_r72=0.05
k_d73=0.3333333333333333
k_r73=0.18
k_d74=1.0
k_r74=0.06
k_d75=1.0
k_r75=0.19
k_d76=1.0
k_r76=0.16
k_d77=1.0
k_r77=0.05
k_d78=1.0
k_r78=0.07
k_d79=1.0
k_r79=0.1
k_d80=1.0
k_r80=0.43
k_d81=1.0
k_r81=1.0
k_d82=0.5
k_r82=0.1
k_d83=0.5
k_r83=0.43
k_d84=0.5
k_r84=0.36
k_d85=1.0
k_r85=0.1
k_d86=1.0
k_r86=0.43
k_d87=1.0
k_r87=0.05
k_d88=1.0
k_r88=0.31
k_d89=1.0
k_r89=0.1
k_d90=1.0
k_r90=0.43
k_d91=1.0
k_r91=0.31
k_d92=1.0
k_r92=0.54
k_d93=1.0
k_r93=0.22
```

```
k_d94=1.0
k_r94=0.78
k_d95=1.0
k_r95=0.46
k_d96=1.0
k_r96=1.0
k_d97=1.0
k_r97=1.0
k_d98=0.3333333333333333
k_r98=0.22
k_d99=0.3333333333333333
k_r99=0.78
k_d100=0.3333333333333333
k_r100=0.46
k_d101=1.0
k_r101=0.22
k_d102=1.0
k_r102=0.78
k_d103=1.0
k_r103=0.46
k_d104=1.0
k_r104=0.22
k_d105=0.3333333333333333
k_r105=0.22
k_d106=0.3333333333333333
k_r106=0.78
k_d107=0.3333333333333333
k_r107=0.46
k_d108=0.3333333333333333
k_r108=0.22
M_hom__L_e=0.5488135039273248
M_lys__L_e=0.7151893663724195
M_4abut_e=0.6027633760716439
M_urea_e=0.5448831829968969
M_fe3_e=0.4236547993389047
M_rib__D_e=0.6458941130666561
M_ura_e=0.4375872112626925
M_gthrd_e=0.8917730007820798
updated_csc052=0.9636627605010293
M_glu__L_e=0.3834415188257777
M_xylan4_e=0.7917250380826646
M_glyc_e=0.5288949197529045
M_arg__L_e=0.5680445610939323
updated_csc040=0.925596638292661
csc043=0.07103605819788694
M_orn_e=0.08712929970154071
M_fe3pyovd_kt_e=0.02021839744032572
M_ptrc_e=0.832619845547938
```

```
    M_o2_e=0.7781567509498505
    M_h2s_e=0.8700121482468192
    updated_csc031=0.978618342232764
    M_gua_e=0.7991585642167236
    M_s_e=0.46147936225293185
    M_nh4_e=0.7805291762864555
    M_ppap_e=0.11827442586893322
    updated_csc009=0.6399210213275238
    M_pydxn_e=0.1433532874090464
    M_ser__L_e=0.9446689170495839
    M_asp__L_e=0.5218483217500717
    M_xyl__D_e=0.4146619399905236
    M_pi_e=0.26455561210462697
    M_glyald_e=0.7742336894342167
    M_hom__L_e is "M_hom__L_e"
    M_lys__L_e is "M_lys__L_e"
    M_4abut_e is "M_4abut_e"
    M_urea_e is "M_urea_e"
    M_fe3_e is "M_fe3_e"
    M_rib__D_e is "M_rib__D_e"
    M_ura_e is "M_ura_e"
    M_gthrd_e is "M_gthrd_e"
    updated_csc052 is "updated_csc052"
    M_glu__L_e is "M_glu__L_e"
    M_xylan4_e is "M_xylan4_e"
    M_glyc_e is "M_glyc_e"
    M_arg__L_e is "M_arg__L_e"
    updated_csc040 is "updated_csc040"
    csc043 is "csc043"
    M_orn_e is "M_orn_e"
    M_fe3pyovd_kt_e is "M_fe3pyovd_kt_e"
    M_ptrc_e is "M_ptrc_e"
    M_o2_e is "M_o2_e"
    M_h2s_e is "M_h2s_e"
    updated_csc031 is "updated_csc031"
    M_gua_e is "M_gua_e"
    M_s_e is "M_s_e"
    M_nh4_e is "M_nh4_e"
    M_ppap_e is "M_ppap_e"
    updated_csc009 is "updated_csc009"
    M_pydxn_e is "M_pydxn_e"
    M_ser__L_e is "M_ser__L_e"
    M_asp__L_e is "M_asp__L_e"
    M_xyl__D_e is "M_xyl__D_e"
    M_pi_e is "M_pi_e"
    M_glyald_e is "M_glyald_e"
end
"""
```

Appendix B – Alternative Diagram Representations

B.1 Petri-net representation of the consumer resource model:

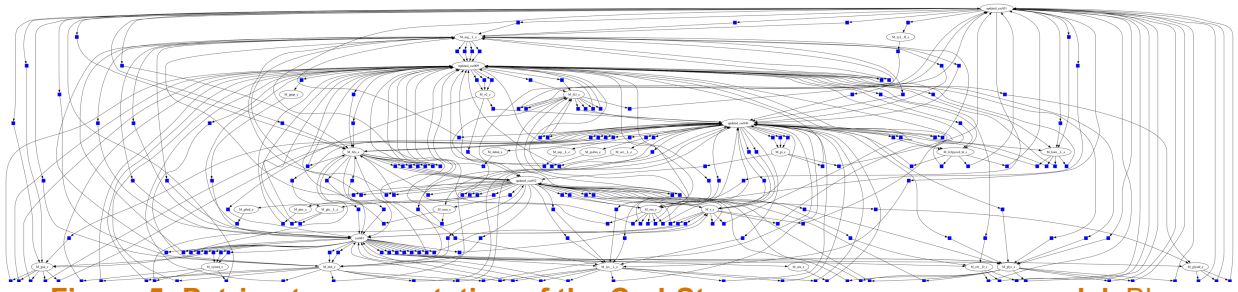

Figure 5: Petri-net representation of the CarbStor consumer resource model. Blue rectangles represent transitions and ovals represent states.

B.2 Graph representation of the integrated model and workflow:

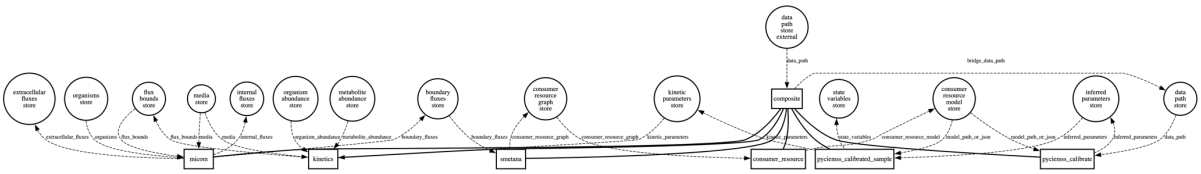

Figure 6: Graphical representation of a proposed integrated model for CarbStor in Vivarium. MICOM is used to simulate genome-scale community models. SMETANA is used to upscale genome-scale community models to consumer-resource models. Kinetic parameters of the consumer resource model are calibrated using metabolite and organism abundance data, and the calibrated kinetic parameters are fed back into the down-scaled community genome-scale metabolic models.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov