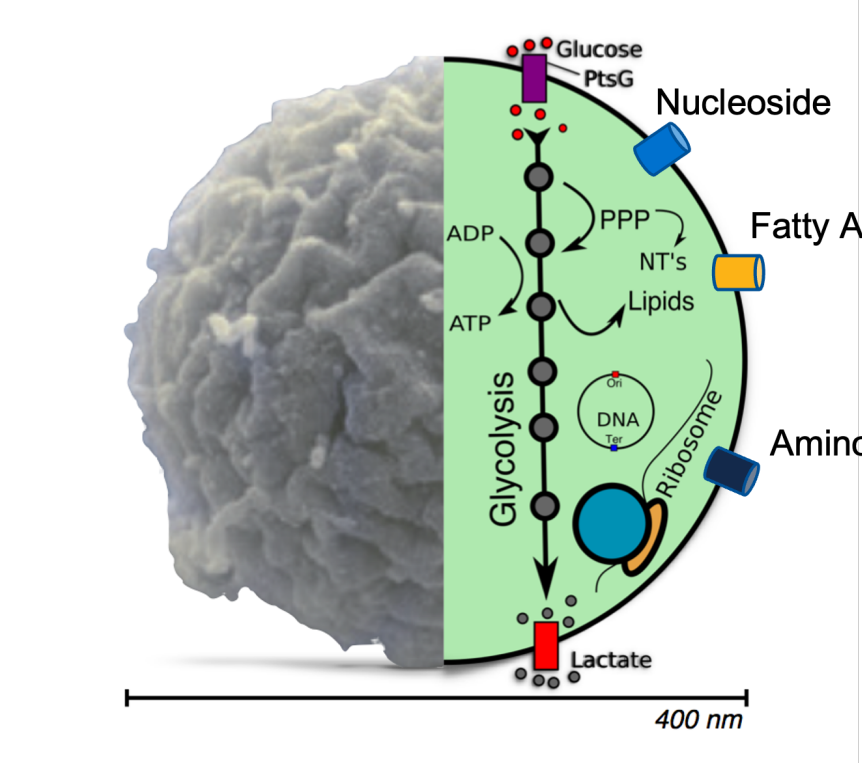
Coupled Genetic Information Processes and Metabolism in Minimal Cell, JCVI-syn3A



STC-QCB Summer School Tutorial
August 5th-9th, 2024

Enguang Fu and Zaida Luthey-Schulten

University of Illinois at Urbana-Champaign

Description

This Tutorial is written for STC-QCB Summer School describe how to use the software to perform and analyze hybrid stochastic/deterministic simulations of a minimal bacterium.
Lattice Microbes development is supported in part by NSF Science and Technology Center for Quantitative Cell Biology (NSF DBI-2243257) and NSF (MCB: 2221237 “Simulating a growing minimal cell: Integrating experiment and theory”).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Lattice Microbe Overview

Lattice microbe (LM) is a GPU accelerated stochastic simulation platform. The latest published version 2.4 can be found at Lattice Microbe 2.4 on Luthey-Schulten Lab GitHub. In this tutorial, we use an updated version 2.5 under development.
LM is designed to simulate stochastic processes in biological cells using the Chemical Master Equation (CME) and the Reaction Diffusion Master Equation (RDME). With hooking, one can also incorporate metabolic reactions using Ordinary Differential Equation (ODE) solver and chromosome dynamics with Brownian Dynamics (BD) code.
The architecture of LM is shown in Figure 1.1. pyLM and jLM are two python-based Problem Solving Environments (PSE) developed in 2013 [3] and 2018, respectively. The API of pyLM and jLM can be found at Lattice Microbes API Reference.
With pyLM and jLM, one can easily construct the system purely using Python or Jupyter Notebook. And the calculation is still performed in C++.

1.2 pyLM and jLM Overview

pyLM and jLM are a Problem Solving Environment (PSE) for biological simulations [3]. Written in Python, it wraps and extends Lattice Microbes. The PSE is comprised of a base set of functionality to set up, monitor and modify simulations, as well as a set of standard post-processing routines that interface to other Python packages, including NumPy, SciPy, H5py, iGraph to name a few.

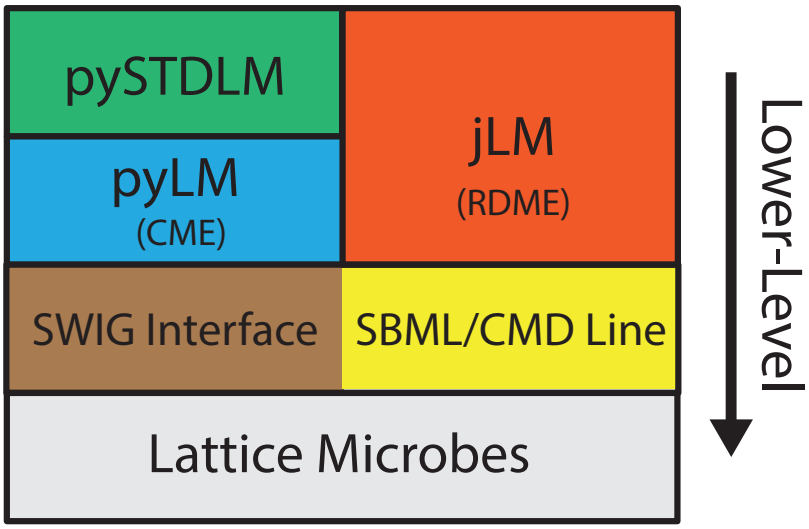


Figure 1.1: A schematic Architecture of the Lattice Microbes software.

The PSE is shown schematically in Figure 1.1. It sits on top of a SWIG interface that allows the C++ code to be accessible from the Python terminal. Using pyLM allows the user to set up, run and post-process simulations all within a single script. A general workflow is shown in Figure 1.2. For tutorials on using pyLM please see the "pyLM Reference pdf" in the Luthey-Schulten webpage under Software/Documentations.
The main difference between pyLM and jLM is that pyLM is used specifically for homogenous Chemical Master Equation (CME) simulation while jLM for heterogenous Reaction-Diffusion Master Equation (RDME) simulation. Designed with Jupyter Notebook interfaces, jLM offers a suite of functions to visualize the RDME simulation states like spatial regions, species and reactions. To make PSE more user friendly, we are working on integrating pyLM and jLM into one PSE. In this tutorial, you will only use pyLM since we are dealing with homogenous system.

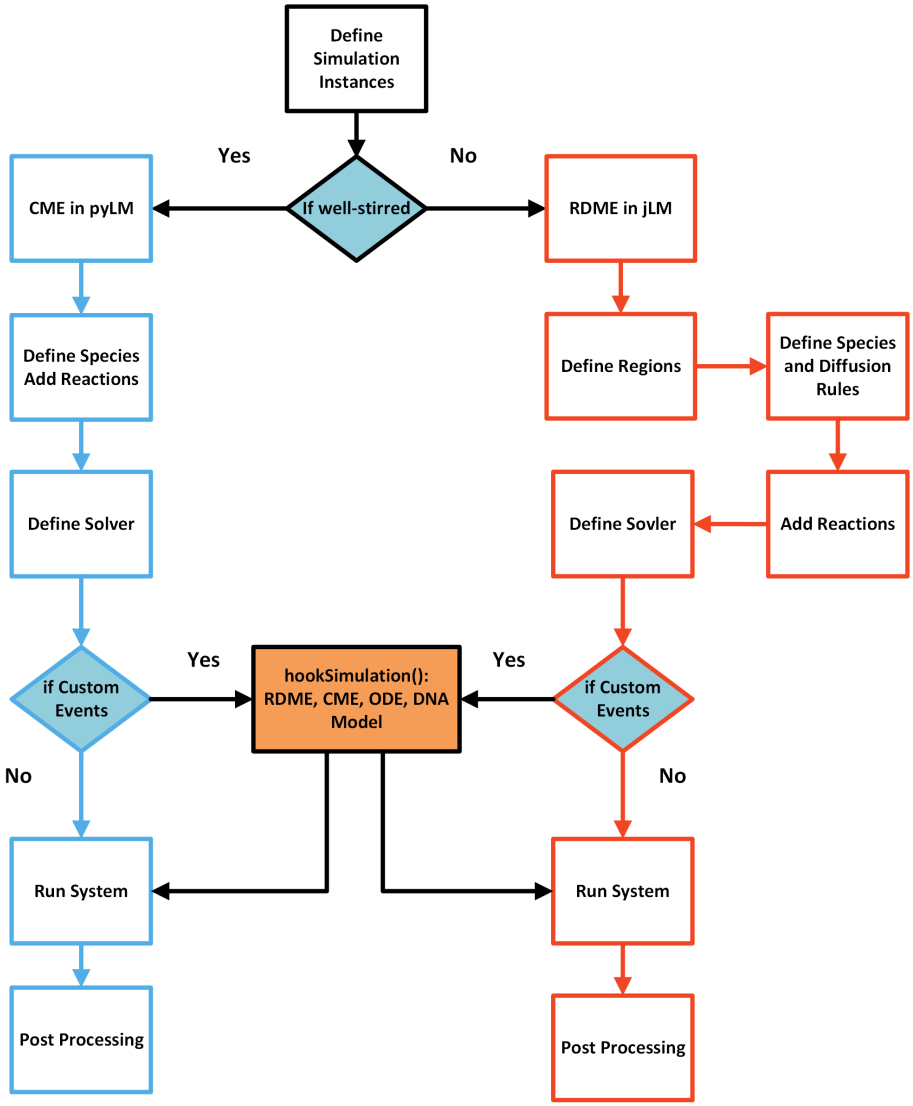


Figure 1.2: Workflow in pyLM and jLM. In jLM, regions and diffusion rules are defined. In the hookSimulation(), various algorithms can be incorporated with CME or RDME. In this tutorial, we will show hybrid CME/ODE algorithm implemented by Lattice Microbe

1.3 Stochastic Modeling

1.3.1 Cellular Modelling Algorithms and Hooking

Figure 1.3 shows the common cellular modelling algorithms. CME and RDME are the homogenous and heterogenous stochastic chemical reaction kinetics algorithms. Ordinary Differential Equations (ODE) is suitable for macroscopic and homogenous chemical kinetics that includes metabolic networks in the minimal cell. Brownian Dynamics (BD) is used in continuum polymer chromosome model.

Lattice Microbes inherently supports CME and RDME algorithms, and with hooking, we can implement hybrid CME/ODE or more complex BD/ODE/RDME algorithms.

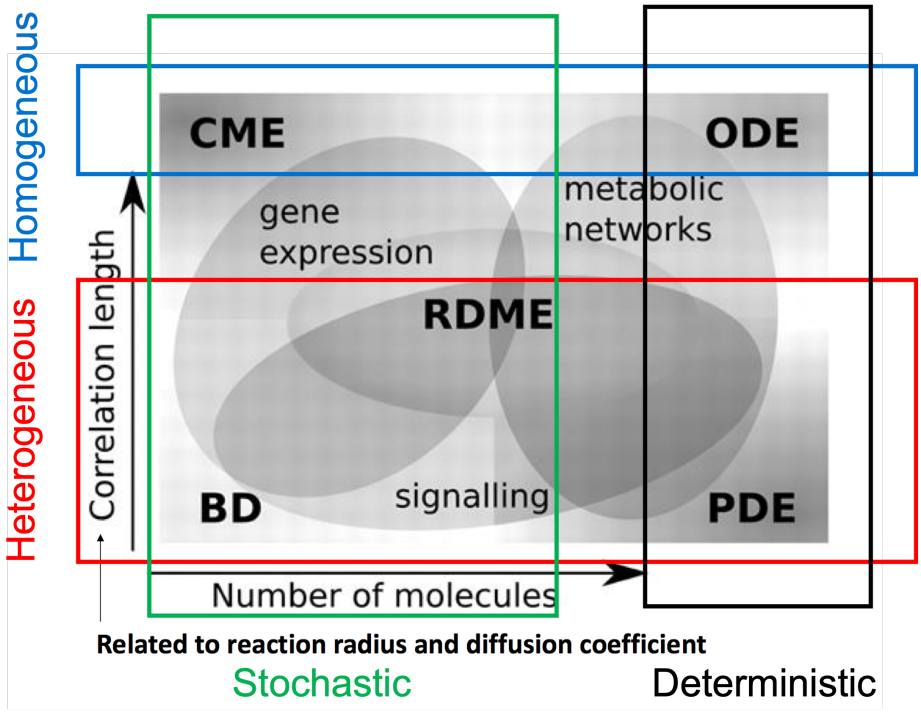


Figure 1.3: Lattice Microbe inherently supports stochastic CME and RDME algorithms. With hookSimulation, CME and RDME can communicate with ODE (for metabolism) and BD (for chromosome model).

1.3.2 Chemical Master Equation

General CME formula is:

$$\frac{dP(\mathbf{x},t)}{dt} = \sum_{r}^{R}[-a_r(\mathbf{x})P(\mathbf{x},t) + a_r(\mathbf{x}_\nu - \mathbf{S_r})P(\mathbf{x}-\mathbf{S_r},t)]$$

In CME, the state of the system $\mathbf{x}$ as a vector is the counts of all species. The transition between different states is the fire of single chemical reaction r out of all reactions R with stoichiometry $\mathbf{S_r}$. The probability for the fire of each single chemical reaction in next time step dt is $a_r(\mathbf{x})dt$, where we name $a_r(\mathbf{x})$ propensity of reaction r under system state $\mathbf{x}$. Propensities are calculated shown in in Table 1.1 where macroscopic ODE rate constants are first converted into stochastic ones and then calculate the propensity using mass action law.

Table 1.1: Zeroth, First and Second Order reactions in ODE and CME. Here, the stochastic rate constant should be computed from the macroscopic rate constant using the volume of the experiment, V, and Avogadro's number, N_A. n_A and n_B are the absolute numbers of A and B molecules.

Order	**Form**	**Parameters**	**Macroscopic Units**	**Stochastic Rate Constant (s^{-1})**	**Propensity (s^{-1})**
0th	$\emptyset \rightarrow A$	k_{0th}	$M \cdot s^{-1}$	$k_{0th} \cdot V \cdot N_A$	$k_{0th} \cdot V \cdot N_A$
1st	$A \rightarrow B$	k_{1st}	s^{-1}	k_{1st}	$k_{1st} n_A$
2nd	$A + B \rightarrow C$	k_{2nd}	$M^{-1} \cdot s^{-1}$	$\frac{k_{2nd}}{V \cdot N_A}$	$\frac{k_{2nd}}{V \cdot N_A} n_A n_B$

Thus, Chemical Master Equation states that the derivative of probability, $P(\mathbf{x}, t)$ to stay in state $\mathbf{x}$ with respect to time t equals the summation of in-flow, $\sum_r^R [a_r(\mathbf{x}_\nu - \mathbf{S_r})P(\mathbf{x} - \mathbf{S_r}, t)]$ minus the out-flow, $\sum_r^R [-a_r(\mathbf{x})P(\mathbf{x}, t)]$.

Similarly to ODE, CME for complex system is unsolvable. In most cases, we use Gillespie Algorithm (also known as Stochastic Simulation Algorithm or SSA) to sample multiple trajectories which represent the stochastic evolution of the system. We will not dive into Gillespie Algorithm in this tutorial but you are free to look more at Wiki_Gillespie and Stochastic Simulation of Chemical Kinetics from Daniel T. Gillespie afterwards.

RDME is used to simulate spatially heterogenous systems. Within each subvolume, the reactions are assumed to be well-stirred and described by CME.

$$\frac{dP(\mathbf{x}, t)}{dt} = \sum_{\nu}^{V} \sum_{r}^{R} [-a_r(\mathbf{x}_\nu) P(\mathbf{x}_\nu, t) + a_r(\mathbf{x}_\nu - \mathbf{S_r}) P(\mathbf{x}_\nu - \mathbf{S_r}, t)]$$
$$+ \sum_{\nu}^{V} \sum_{\xi}^{\pm\hat{i},\hat{j},\hat{k}} \sum_{\alpha}^{N} [-d^\alpha_{xi} x^\alpha_\nu P(\mathbf{x}, t) + d^\alpha_\xi (x^\alpha_{\nu+\xi} + 1^\alpha_\nu) P(\mathbf{x} + 1^\alpha_{\nu+\xi} - 1^\alpha_\nu, t)]$$

Chapter 2

Tutorials for CME in pyLM

2.1 Tutorial 1: Bimolecular Reaction Solved in ODE and CME

In the first tutorial, we will motivate stochastic modeling with a simple example of a bimolecular reaction. The counts of reactants are small enough that you will see the fluctuations and variance of the stochastic result compared to the deterministic one.
You will use **Tut1.1-ODEBimol.ipynb** to run the ODE of Bimolecular reaction solved with Scipy and **Tut1.2-CMEBimol.ipynb** to run the CME of the Bimolecular reaction with pyLM in Lattice Microbe.
We will simulate the association/dissociation reaction of hypothetical molecules:

$$\mathrm{A} + \mathrm{B} \underset{\mathrm{k_r}}{\overset{\mathrm{k_f}}{\rightleftharpoons}} \mathrm{C}$$

Let us start out with a low number of each particles: 100 of A and B and 0 of C. Let us imagine simulating that problem in a microbe sized volume of 1 fL. Also, let us start off with rates of $k_f = 1.07 \times 10^6 M^{-1}s^{-1}$ and $k_r = 0.351/s$.
Under ODE representation using concentrations of species, the equations are:

$$\begin{aligned}
\frac{d[A]}{dt} &= -k_f[A][B] + k_r[C] \\
\frac{d[B]}{dt} &= -k_f[A][B] + k_r[C] \\
\frac{d[C]}{dt} &= k_f[A][B] - k_r[C]
\end{aligned}$$

Since we are using counts rather than concentrations in CME, the forward rate constant (2nd order) is divided by Avogadro's number and the volume of the cell which takes us to units of molecules per second. Conversion to stochastic rate constant is shown in Table 1.1.

Now go to Jupyter Notebook file **Tut1.1-ODEBimol.ipynb** and **Tut1.2-CMEBimol.ipynb** (latter's code shown in List 2.1) on your Jupyter Notebook webpage.

Listing 2.1: `Tut1.2-CMEBimol` — Code used to solve the bimolecular reaction with the discrete/stochastic CME implementation using Lattice Microbes.

```python
# Import Standard Python Libraries
import os
import numpy as np
import matplotlib.pyplot as plt

# Import pyLM Libraries
from pyLM import *
from pyLM.units import *
from pySTDLM import *
from pySTDLM.PostProcessing import *

# Constants
V  = 1.0e-15       # L
NA = 6.022e23      # molecules/mole
kf_ODE = 1.07e6; # /M/s
kr_ODE = 0.351 # /s
kf = 1.07e6/(NA*V) # convert from 1.07e5 /M/s to /counts/s
kr = 0.351         # /s

### Define a CME Simulation

# Create our CME simulation object
sim = CME.CMESimulation(name='Bimolecule Reaction')

# Define our chemical species
species = ['A', 'B', 'C']
sim.defineSpecies(species)

# Add reactions to the simulation
sim.addReaction(reactant=('A','B'), product='C', rate=kf)
sim.addReaction(reactant='C', product=('A','B'), rate=kr)

# Set our initial species counts
sim.addParticles(species='A', count=50)
sim.addParticles(species='B', count=50)
sim.addParticles(species='C', count=0)

# Define simulation parameters: run for 30 seconds, saving data every 30 ms
sim.setWriteInterval(microsecond(30))
sim.setSimulationTime(30)
filename = "./T1.2-bimol.lm"
os.system("rm -rf %s"%(filename))
sim.save(filename)

# Run reps replicates using the Gillespie solver
reps = 10
sim.run(filename=filename, method="lm::cme::GillespieDSolver", replicates=reps
    )

### Post-Process Simulation
plotfolder = './plots_bimolecule/'

if not os.path.exists(plotfolder):
```

```
    os.mkdir(plotfolder)

# Plot the traces of one replicate to see the fluctuation
fieHandle = PostProcessing.openLMFile(filename)
rep = 1
picturepath = plotfolder + 'bimolecule_CME_trace_replicate{0}.png'.format(rep)
PostProcessing.plotTrace(fieHandle, species=['A','C'], replicate=1, filename=
    picturepath)
closeLMFile(fieHandle)

# Plot average and variance among all replicates
picturepath = plotfolder + 'bimolecule_CME_Avgs_Vars_{0}replicates.png'.format
    (reps)
PostProcessing.plotAvgVarFromFile(filename = filename, species = ['A','C'],
    outfile = picturepath)
```

As mentioned in the introduction part, pyLM Problem Solving Environment is intensively used to construct the CME system. To do so, we need to import `pyLM` module.

The line `sim=CME.CMESimulation()` creates an empty simulation object. The next lines define the chemical species; in pyLM species are named by python strings and must be registered with the simulation using the `defineSpecies` command. `addReaction` function adds reactions to the defined species.

In Lattice Microbes, you need to specify both the forward and back reactions separately. The first and second argument can be either a tuple of reactants or a string when only one reactant is specified. Lattice Microbes currently supports 0th, 1st and 2nd order reactions, and reaction rates must be specified in the "stochastic" format (see Table **??**). In the special case of a 0th order reaction, the empty string `''` should be passed as the reactant. In addition, annihilation reactions can be specified by passing the empty string `''` as the product parameter.

The following lines with `addParticles` define the initial species counts.

Next the simulation parameters are specified, time steps will be written out every 30 microseconds and the total simulation will run for 30 seconds. The next line is of particular importance; the simulation must be saved to a file before running the simulation.

Finally, we call the `run(...)` command on the simulation object giving it the name of the simulation file, the simulation method and the number of independent trajectories (replicates) to run of that simulation. In this tutorial, we use Direct Gillespie Algorithm to sample stochastic bimolecular reaction system.

By running the simulation, you will see a long standard output showing the number of finished replicates. Generally, CME simulation will finish rather quickly.

`pySTDLM` is a library of standard functionality such as standard reaction systems, cell systems. In addition it contains a number of pre- and post-processing functionality. In the post processing part, we showed two build-in functions in `pySTDLM`: `PostProcessing.plotTrace` for a single replicate and `PostProcessing.plotAvgVarFromFile` for the whole ensemble to visualize the results.

In the jupyter notebook, we also introduce a more general way to do the analysis where we first serialize the output LM file into a 3D numpy array and plot using custom functions. You will use this method in Tutorial 2 and 3 when doing analysis.

Figure 2.1 shows the deterministic result from ODE and one single replicate's stochastic result

from CME.

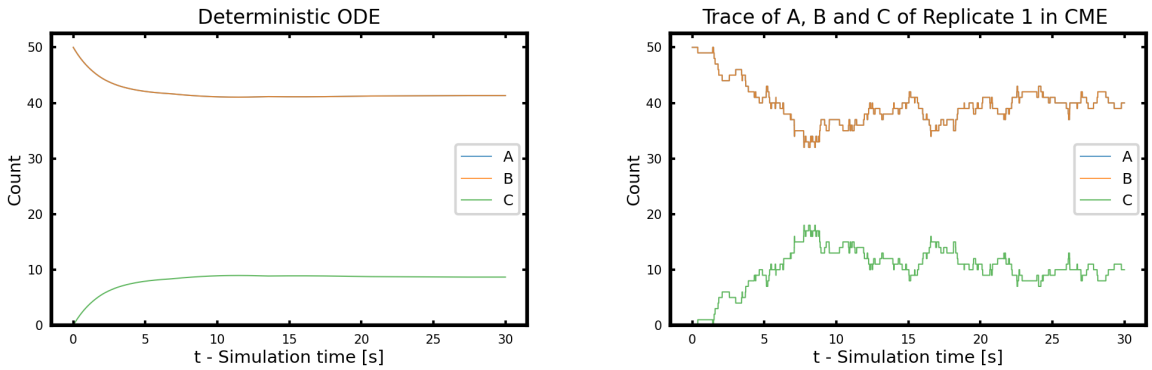


Figure 2.1: ODE and Single CME Trace of Bimolecular Reaction.

In ODE result, what you should note is that the count of each species varies smoothly across the time course. However, the count of each reactant must be in integer numbers due to the discreteness of molecules. Further, the reactions occur via the collision between two molecules, so the change of count also in integer units. Under such reasoning, the ODE result is not accurate anymore for microscopic reactions where the counts of reactants are low. Stochastic modeling was designed to address this point.

You might note that the behavior in stochastic result is qualitatively the same to the deterministic ODE result, however there appears to be considerable fluctuation, even after the system has come to equilibrium. This is due to the stochastic nature of the process, where the reaction can transiently fluctuate away from the equilibrium value. In addition, you may be able to tell that the changes in particle number from one time to another are in integer increments, though this will become considerably more obvious at lower number of particles.

The averaged, minimum and maximum of species' count across the whole replicates are shown in Figure 2.2.

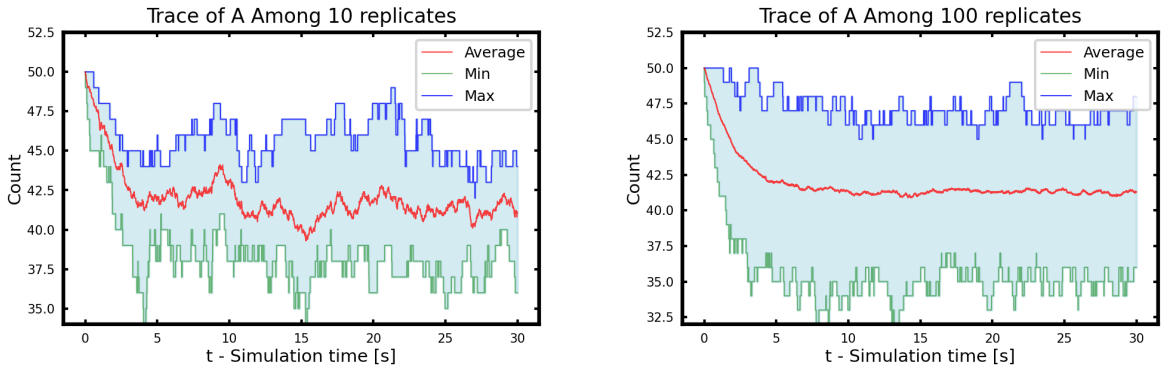


Figure 2.2: Ensemble Average of A in 10 (Left) and 100 (Right) replicates in Stochastic Bimolecular Reaction.

Question

1. Change the replicates number from 10 to 100 or even more in Tutorial 1.2. You need to change the variable *reps* and restart the Jupyter Notebook kernel to start a new CME simulation. See Figure 2.2 for the situation with 10 and 100 replicates. Does the higher replicates number lead to a smoother average and variance?
2. Multiple both the forward and backward rate constants by 10 or even 100 by changing variable *fold*, and restart and run ODE and CME Jupyter Notebooks again. In ODE, does the system converge to equilibrium faster? Does the equilibrium count change? In CME, does the fluctuation in single replicate become faster? How about the ensemble average? See Figure 2.3.

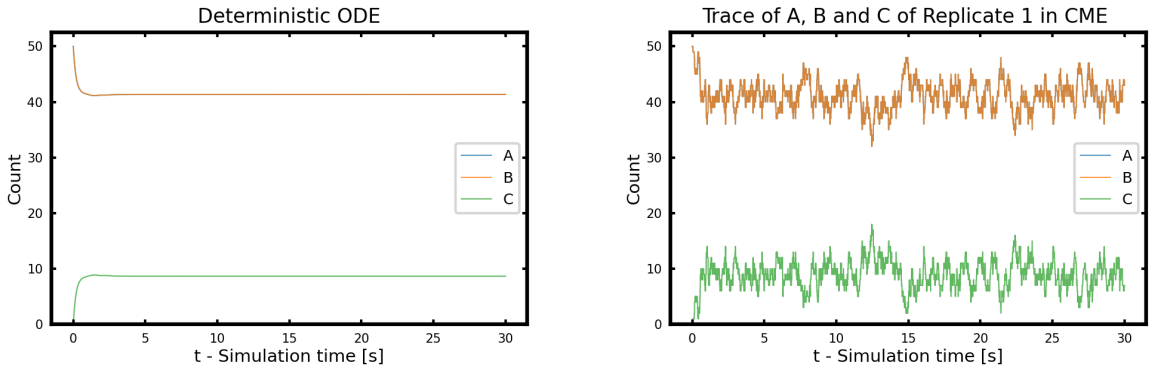
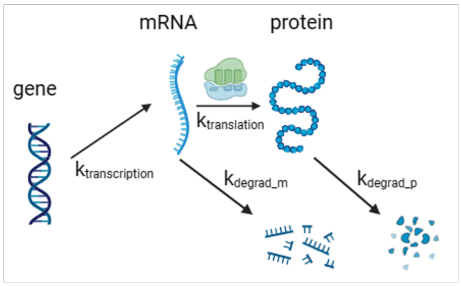


Figure 2.3: ODE and a Single CME Trace with 10 times faster Forward and Backward Rate Constants.

Question 3 and 4 are more challenging and not so relevant to the comparison between deterministic and stochastic kinetic. So please skip these two at the first time and we can have more Question after the whole tutorial.
3. $Challenge$ Can you derive an analytical solution for the system of equations? Based on the expression, try fitting the rate constants using the results of the stochastic simulations. You may find scipy.optimize.curve_fit() useful for this.
4. $Challenge$ Does the theoretical average number from CME always be consistent/same with the count from ODE? You may find Page 420 in McQuarrie's classic paper STOCHASTIC APPROACH TO CHEMICAL KINETICS helpful.

2.2 Tutorial 2: Genetic Information Process in CME

Open ChatGPT and ask this question: **Give an example on large differences in rate constants that lead to significant fluctuations**. You can ask ChatGPT to give you more examples. Please also check this answer: gene expression, which is our next tutorial: gene expression model.
With 3 species and 4 reactions, this classic and simplest genetic information process starts from the transcription of gene to mRNA. mRNA can be translated to protein or degraded to its monomers. Protein can also be degraded. The reactions and rate constant is shown in Figure 2.4 and Table 2.1. The rate constants are for DnaA Coding Gene (G_0001) of minimal cell. The first three rate constants are calculate based on the initial concentrations of nucleotides and amino acids charged tRNA in the Cell paper [4]. We also fix the gene copy number to 1 and assume initial count of mRNA to be 1. The initial count of protein is 0. The degradation rate of protein is estimated based half-life 25 hours [5].



$$\frac{\partial P_{m,n}}{\partial t} = k_{transcription}(P_{m-1,n} - P_{m,n}) + k_{translation}m(P_{m,n-1} - P_{m,n}) + k_{degra,mRNA}\left[(m+1)P_{m+1,n} - mP_{m,n}\right] + k_{degra,ptn}\left[(n+1)P_{m,n+1} - nP_{m,n}\right]$$

Figure 2.4: Genetic Information Processing Reaction Systems and its Chemical Master Equation

Table 2.1: Four reactions with their rate constants.

Names	**Reaction**	**Rate Constant (s^{-1})**	**Propensity (s^{-1})**
Transcription	Gene $\rightarrow$ mRNA	$k_{\text{transcription}} = 6.41E-4$	$k_{\text{transcription}}$
Degradation of mRNA	mRNA $\rightarrow \emptyset$	$k_{\text{deg,m}} = 2.59E-3$	$k_{\text{deg,m}} N_{\text{mRNA}}$
Translation	mRNA $\rightarrow$ mRNA + Protein	$k_{\text{translation}} = 7.20E-2$	$k_{\text{translation}} N_{\text{mRNA}}$
Degradation of Protein	Protein $\rightarrow \emptyset$	$k_{\text{deg,p}} = 7.70E-6$	$k_{\text{deg,p}} N_{\text{ptn}}$

Now go to Jupyter Notebook file **Tut2.1-GeneticInformationProcess.ipynb**. Here we will simulate this toy genetic information processing (GIP) model. In this given Jupyter Notebook, the default simulation length is **6300 seconds**, the entire cell cycle with write interval 1 second for **10 replicates**.

Listing 2.2: `Tut.2.1-GeneticInformationProcess` — Code to set up and execute the gene expression model with CME.

```
# Import Standard Python Libraries
import os
import numpy as np
import matplotlib.pyplot as plt

```

```
# Import pyLM Libraries
from pyLM import *
from pyLM.units import *
from pySTDLM import *
from pySTDLM.PostProcessing import *

# Constants
k_transcription  = 6.41e-4       # Transcription, s^-1
k_degra_mRNA = 2.59e-3     # degradation of mRNA, s^-1
k_translation = 7.2e-2        # translation, s^-1
k_degra_ptn = 7.70e-6      # degradation of protein, s^-1

### Define CME simulation

# Create our CME simulation object
sim = CME.CMESimulation(name='Gene Expression')

# Define our chemical species
species = ['gene', 'mRNA', 'ptn']
sim.defineSpecies(species)

# Add reactions to the simulation
sim.addReaction(reactant='gene', product=('gene','mRNA'), rate=k_transcription
    )
sim.addReaction(reactant='mRNA', product='', rate=k_degra_mRNA)
sim.addReaction(reactant='mRNA', product=('mRNA','ptn'), rate=k_translation)
sim.addReaction(reactant='ptn', product='', rate=k_degra_ptn)

sim.addParticles(species='gene', count=1)
sim.addParticles(species='mRNA', count=1)
sim.addParticles(species='ptn', count=0)

# Simulation time is 6300 (entire cell life cycle of Minimal Cell).
writeInterval = 1
simtime = 6300

sim.setWriteInterval(writeInterval)
sim.setSimulationTime(simtime)

filename = "./T2.1-GeneticInformationProcess.lm"
os.system("rm -rf %s"%(filename)) # Remove previous LM file
sim.save(filename)

# Run multiple replicates using the Gillespie solver
reps = 100
sim.run(filename=filename, method="lm::cme::GillespieDSolver", replicates=reps
    )

### Post-Processing
plotfolder = './plots_GeneticInformationProcess/'

if not os.path.exists(plotfolder):
    os.mkdir(plotfolder)

```

```
# Plot the trace of mRNA or protein of a single replicate
rep = 3
picturepath = plotfolder + 'mRNA_replicate{0}.png'.format(rep)
PostProcessing.plotTraceFromFile(filename, species=['mRNA'], replicate=rep,
    outfile=picturepath )
picturepath = plotfolder + 'ptn_replicate{0}.png'.format(rep)
PostProcessing.plotTraceFromFile(filename, species=['ptn'], replicate=rep,
    outfile=picturepath )
```

The methods that we use to construct this gene expression simulation is very similar to Tutorial 1. First, we can plot the traces of mRNA and protein in different single replicates to see the stair-stepping trace of mRNA and the burst of protein in Figure 2.5. You will an increase/burst in protein count when there are mRNAs and the protein count holds or gradually decrease when no mRNA. You are encouraged to compare the pattern shown in the PDF to your plots.

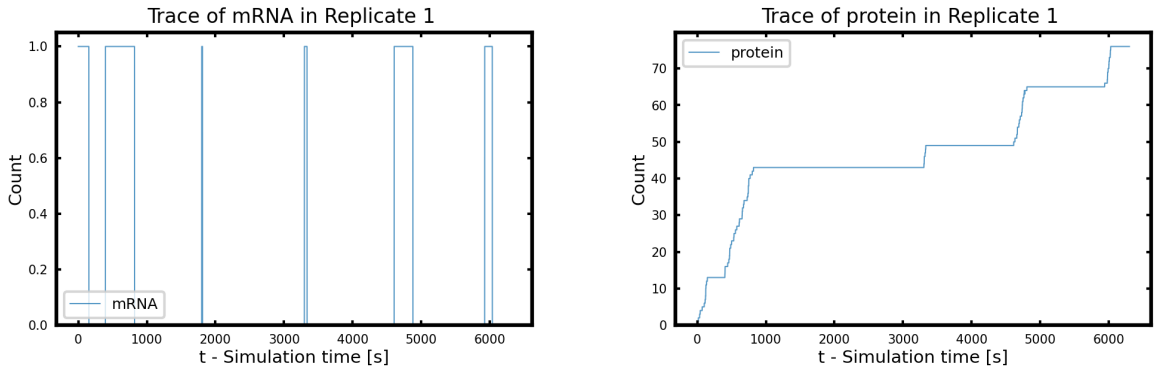


Figure 2.5: Single Cell Trace of mRNA (Left) and protein (Right). Left: Stair-stepping trace of mRNA. Right: Burst of protein due to the stochastic mRNA's translation.

We can also see the average, minimum and maximum of mRNA and protein counts over the whole simulation time span.

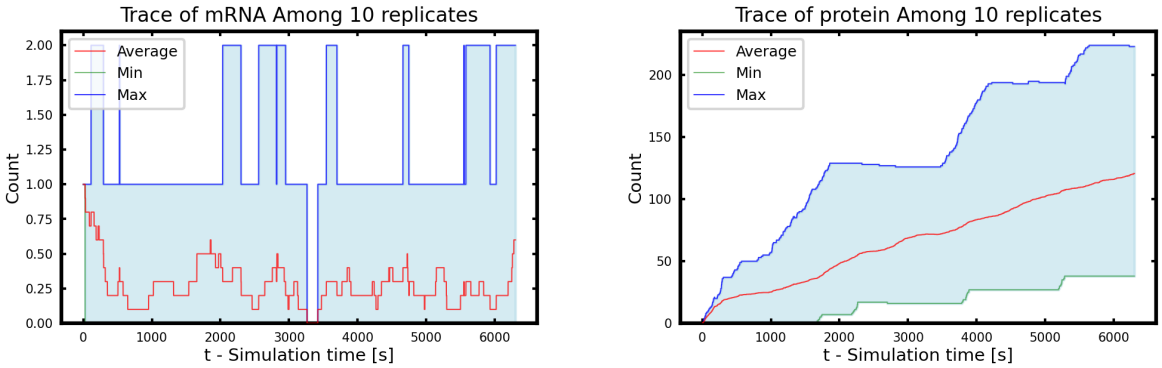


Figure 2.6: Average, Minimum and Maximum of mRNA (Left) and protein (Right) among 10 replicates.

Question

1. Try to increase the replicates numbers *reps* from 10 to 100. Do mRNA and protein reach steady-state during the 6300 seconds' simulation? How can you tell this from the plots? If not reach steady-state, what could be the reason?
2. The initial count of protein P_0001/DnaA from experimental proteomics data is 148 (Check Figure 2.7). Compare the mean count of generated protein at the end of the cell cycle to this experimental count. Does the simulation roughly generate 148 proteins during the entire cell cycle? And why this is important? Please consider cell division.

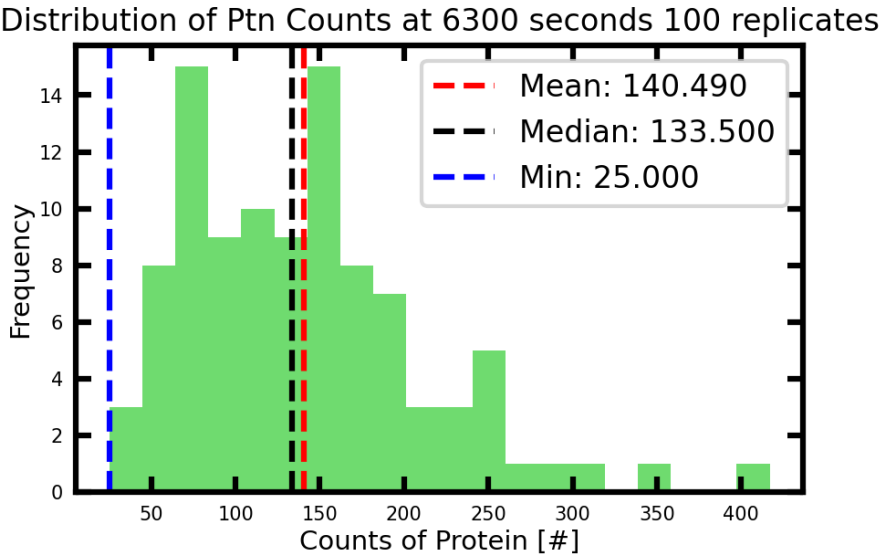


Figure 2.7: Distributions of protein counts. Protein counts Distribution of 100 replicates at the end of the cell cycle.

2.3 Tutorial 3: CME/ODE Whole Cell Model of Minimal Cell, JCVI-syn3A

In this section, we will show you how to use Lattice Microbe to simulate the Minimal Cell with hybrid CME/ODE algorithm.
Since the simulation of the entire cell cycle takes around 6 hours, we will let you only run 2 minutes' cell life. Please go to **commands.md** and launch bash file **mpirun.sh** to simulate 4 cells for 2 minutes in parallel. In the analysis part, you will use prepared 10 healthy trajectories of the entire cell cycle.

2.3.1 Coding Implementation

Launching Simulation in Parallel

To increase the simulation speed, running independent replicates in parallel is inevitable. In our current implementation, we use *mpirun* module to launch simulation python scripts in parallel (Figure 2.8).

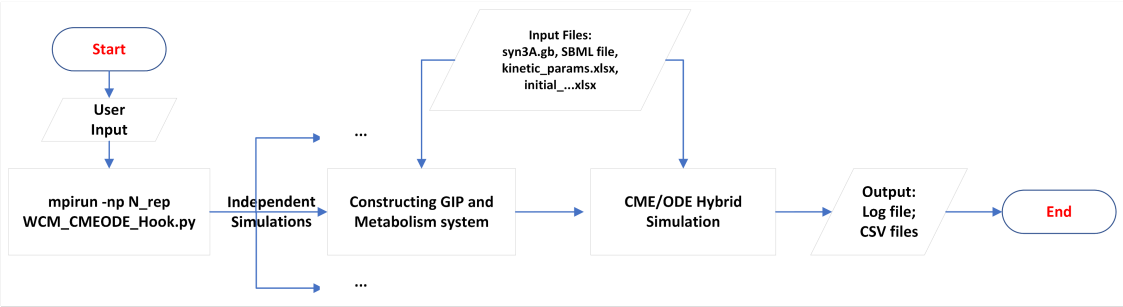


Figure 2.8: Launching Independent Whole Cell Simulations in Parallel using *mpirun* module. The user input will be passed to claim the replicates number, time length, and hook interval. Each simulation is independent with each other. In current simulation we have four main input files and the trajectories will be stored in CSV files with an additional log file.

We launch independent replicates to obtain the statistically significant cellular dynamics. The main script is **WCM_CMEODE_Hook.py** calling multiple other python scripts to construct and simulate the genetic information processes, metabolism and their interactions over the whole cell cycle.
Essentially, we use pyLM and build-in Gillepsie algorithm to construct and simulate the genetic information processes. The ODEs are written/constructed by odecell, a package developed by former group member and simulated by well-known LSODA algorithm in Scipy Package.
The CME/ODE simulation is solely performed on CPUs, however NVIDIA GPUs and CUDA are needed to run Lattice Microbe. Using Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz, each simulation will finish between 5 to 12 hours with 2 GB RAM usage. The variable time cost mainly comes from the adaptive time step LSODA ODE integrator.
The output files of current model are one log file and three CSV files respectively, recording counts of species in genetic information processes and metabolism, fluxes of metabolic reactions, and cell surface area and volume. General size for counts CSV file are 100 MB with approximate 5000 species and 6301 time points.

Input Files

Four main input files are used in whole cell simulation, two Excel files, one Genbank file and one SBML file. The Genbank file contains the sequences and functions of genes, RNAs, and proteins and the SBML file contains the metabolic reactions. The *intitial_concentration.xlsx* file contains the initial count/concentrations of proteins and metabolite while *kinetic_params.xlsx* contains the kinetic parameters of the metabolic reactions.

Table 2.2: Input Files and Information Read In

File	File Format	Information Read IN
syn3A.gb	Genbank	Sequences and functions of genes, RNA and protein
Syn3A_updated.xml	SBML	Stoichiometric Coefficients of Metabolic Reactions
initial_concentrations.xlsx	Excel	Initial counts/concs of proteins, medium and metabolites
kinetic_params.xlsx	Excel	Metabolic Reactions and kinetic parameters
SSU_assembly.json	JSON	Assembly pathways of Ribosome Small Subunit
Largesubunit.xlsx	Excel	Assembly pathways of Ribosome Large Subunit

Now, we will focus on gene JCVISYN3A_0011 to show how the different input files function in our simulation.

The genome information of JCVI-syn3A is stored as standard Genbank file on NCBI with ACCESSION Number CP016816.2. The Genbank file holds much more information than the pure nucleotide sequence.

Go to Genbank file under input_data folder and open with text editor. Search one example JVCISYN3A_0011 and you will see in Figure 2.9. JCVISYN3A_0011 is the LocusTag of this certain gene in organism JCVI-syn3A, serving as a unique identifier. 0011 is the locusNum that will be heavily used in the modeling to distinguish genes, RNAs, and proteins from each other. The start and end index of JCVISYN3A_0011 is 15153 and 16799, respectively and the nucleotide sequence is at the end of the Genbank file. JCVISYN3A_0011 is protein-coding gene and the protein is "Nucleoside Transporter ABC substrate-binding protein" with amino acid sequence shown. This protein is one subunit of ribonucleoside ATP-binding cassette import system that is assumed to import all nucleosides; the system consists of total 4 subunits, protein 0008, 0009, 0010, and 0011.

```
complement(15153..16799)
/gene="rnsB; uptB"
/locus_tag="JCVISYN3A_0011"
complement(15153..16799)
/gene="rnsB; uptB"
/locus_tag="JCVISYN3A_0011"
/inference="EXISTENCE: similar to AA
sequence:RefSeq:WP_020862425.1"
/codon_start=1
/transl_table=4
/product="Nucleoside ABC transporter substrate-binding
protein"
/protein_id="AVX54579.1"
/translation="MKKLLTVLTTFIGISGSVSMVISCKVPTFAEGILGQRVLVVTDG
GNIRDKTFNESSWEGVIKYGSQIHSNFDIKNELEARQFNYKSSIGGHTKWDETKHMFI
NEDYEYAKQNSNNYVETPDHTIDAFRTSYNTAIYKKADALLLAGFGHLGAVDYAADRM
KKAGNKTVVFLDAQYKKDNVISVIFNSELAGFNAGWDAIMWANLPKMTSLNSGEFSKE
AIDASNSKTDMVLQGSATGNKYISIGMFGGITNKNAVDNYMWGLLAAMHVYNNKFAGK
EIELTDNKENKVKYKLQPVYYANLGSKAEVEKLNDVNESSWFSKGFDVGGAKKSGIVD
SLIKNQADIIFPVAGPQINDVLESTGHKPYVIGVDTDQVTSVGSSKKGNETRFITSAK
KNIVSASVYALNRARSLQKAIIKDKTYTSKHEKEIKDGTTLVGEQADWSISSSRKADT
KWSVEKVNGSLTNAANLSVESIDYSKDKAKEIEKNLKETLLKSGETFKQYLTKTSLDK
ALESISKSIKDEEWEKLTLKDNGIAGIKNYWEMLIQSTKK"
```

Figure 2.9: Entry JCVISYN3A_0011 in Syn3A's Genbank file.

SBML (System Biology Markup Language) format is widely used in storing computational models of biological processes. The SBML file here contains the whole metabolic system of Syn3A, including compartments (cellular and extracellular), species (197 cellular and extracellular metabolites), reactions , gene product (enzymatic or transporter proteins associated with reactions), and objective function (for Flux Balance Analysis). It's worth noting that the kinetic constants of 175 reactions are in *kinetic_params.xlsx*.

Ribonucleoside ATP-binding cassette importer import all nucleoside from extracellular media into the cell; one reaction is DSGNabc, irreversible transport of deoxyguanosine in nucleotide metabolism. Figure 2.10 shows reaction DSGNabc in SBML file. We define *reversible* as *false* since DSGNabc is a irreversible transport reaction. There are three reactants and four products. The geneProductAssociation rule is *and*, meaning the four subunit proteins need to function cooperatively.


```
</reaction>
<reaction id="R_DGSNabc" name="DGSNabc" reversible="false" fast="false" fbc:lowerFluxBound="cobra_0_bound" fbc:upperFluxBound="cobra_default_ub">
  <listOfReactants>
    <speciesReference species="M_atp_c" stoichiometry="1" constant="true"/>
    <speciesReference species="M_dgsn_e" stoichiometry="1" constant="true"/>
    <speciesReference species="M_h2o_c" stoichiometry="1" constant="true"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="M_adp_c" stoichiometry="1" constant="true"/>
    <speciesReference species="M_dgsn_c" stoichiometry="1" constant="true"/>
    <speciesReference species="M_h_c" stoichiometry="1" constant="true"/>
    <speciesReference species="M_pi_c" stoichiometry="1" constant="true"/>
  </listOfProducts>
  <fbc:geneProductAssociation>
    <fbc:and>
      <fbc:geneProductRef fbc:geneProduct="G_MMSYN1_0008"/>
      <fbc:geneProductRef fbc:geneProduct="G_MMSYN1_0009"/>
      <fbc:geneProductRef fbc:geneProduct="G_MMSYN1_0010"/>
      <fbc:geneProductRef fbc:geneProduct="G_MMSYN1_0011"/>
    </fbc:and>
  </fbc:geneProductAssociation>
</reaction>
```

Figure 2.10: Entry Reaction DSGNabc in SBML file.

The kinetic constants for this DSGNabc reaction in *kinetic_params.xlsx* is shown as Figure 2.11.

Reaction Name	Subsystem	Parameter Type	Related Species	Value	Units
DGSNabc	Transport	Eff Enzyme Count		P_0009-P_0010	#
DGSNabc	Transport	GPR rule		and	
DGSNabc	Transport	Substrate Catalytic Rate Constant	k^+	1	1/s
DGSNabc	Transport	Product Catalytic Rate Constant	k^-	0	1/s
DGSNabc	Transport	Michaelis Menten Constant	**M_dgsn_e**	0.0019	mM
DGSNabc	Transport	Michaelis Menten Constant	**M_atp_c**	0.023	mM
DGSNabc	Transport	Michaelis Menten Constant	*M_dgsn_c*	0.02	mM
DGSNabc	Transport	Michaelis Menten Constant	*M_pi_c*	10	mM
DGSNabc	Transport	Michaelis Menten Constant	*M_adp_c*	2.8	mM

Figure 2.11: Kinetic Constants for Reaction DSGNabc. Since the counts of protein 0008 and 0011 are poorly covered, only protein 0009 and 0010 are included in DGSNabc reaction.

2.3.2 Minimal Genome and Genetic Information Processes

Minimal Cell and Minimal Genome

Minimal cell, JCVI-syn3A is a synthetic bacterium based on mother organism Mycoplasma mycoides capri published by J. Craig Venter Institute in 2016 [6]. JCVI-syn3A has a doubling time of about 2 hours and consistently forms spherical cells of approximately 200 nm in radius.
JCVI-syn3A's genome is minimal in all living organism, 543 kbp long and containing 452 genes code for proteins and 38 genes for RNAs. The ratio of unclear protein-coding genes is 90 out of 452, which is also smallest compared to other well-studied organism, including yeast and E. coli [7].

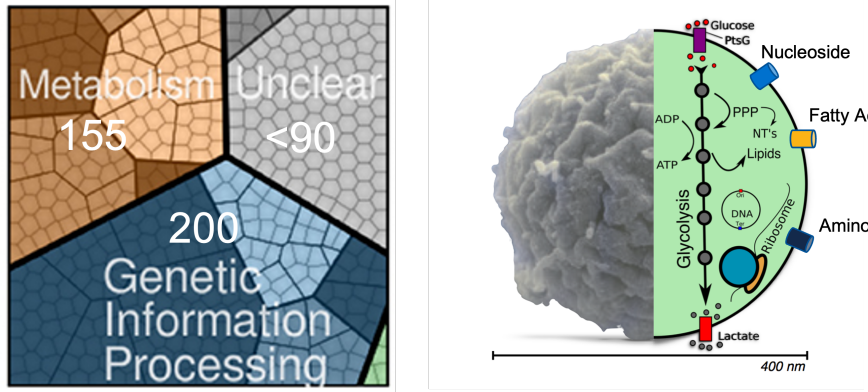
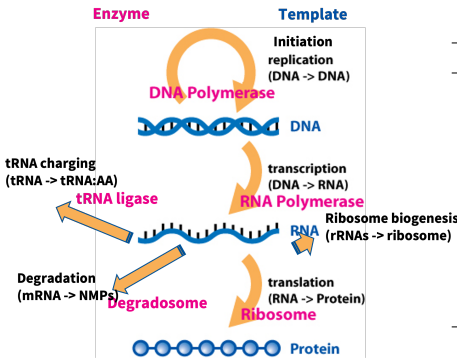


Figure 2.12: Left: Protein Coding Genes of JCVI-syn3A. Only less than 90 genes are unclear. Right: JCVI-syn3A with genetic information processes and metabolism shown

Genetic Information Processes

Genetic information processes connect the blueprint genes to functional proteins. In our current whole cell model of minimal cell, we mainly consider seven types of processes listed in table. Replication copies the genetic information with the regulation of replication initiation. Transcription copies sequential information from DNA to RNA. There are three types of RNA, including mRNA, rRNA and tRNA that are tightly connected by translation. Translation takes place on ribosomes, where mRNA is read and an amino acid chain is generated according to the sequence of mRNA. rRNA, together with other ribosome proteins make up ribosomes in ribosomal biogenesis [8]. tRNA is the carriers and identifier of amino acid to the ribosome.



Name	Biological Function
Replication	Copy genetic Information
Replication Initiation	Regulate replication
Transcription	Copy DNA into RNA
Translation	Convert mRNA into functional proteins
Degradation	Regulate translation and recycle NMPs
Ribosome Biogenesis	Assembly ribosomes
tRNA Charging	Carry AAs for translation

Figure 2.13: Left: Genetic Information Flow. Right: Functions of Seven Genetic Information Processes

The translation and degradation reactions in current whole cell model are more precise compared to the simplest case in Tutorial 2 as for each reaction, mRNA first needs to bind with a complex machinery, degradosome or ribosome. As the busiest species in genetic information processes, mRNA can also be degraded by binding with degradosome. This competition of mRNA to bind with ribosome or degradosome is a important pathway to regulate genetic information processes that will shown in the following result.

Replication initiation is modeled by the binding of multi-domain protein DnaA (encoded by JCVISYN3A_0001) and replisome with certain region called oriC on the chromosome [5].

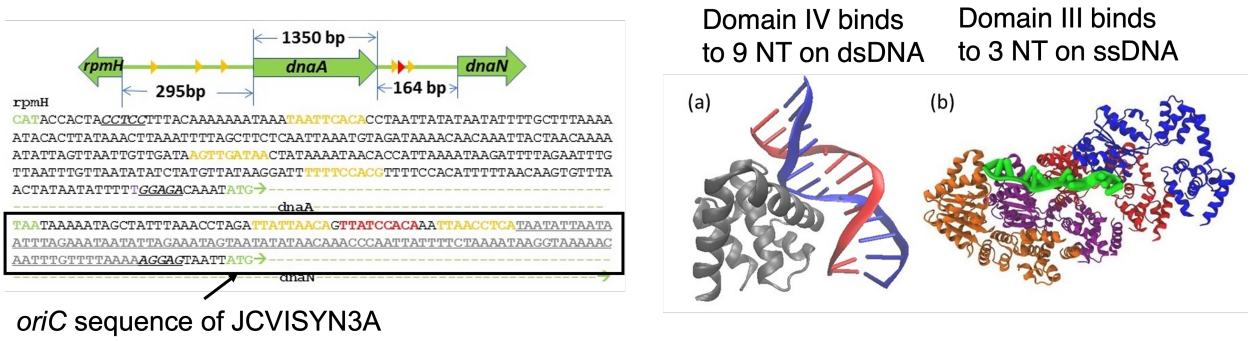


Figure 2.14: Left: oriC region. 9 nucleotide signature binding with DnaA domain IV shown in yellow and red, 3 nuclotide AT rich region binding with DnaA domain III shown in grey Right: PDB structure of DnaA domain IV and domain III binding with chromosome a): [1] b): [2]

First stage of initiation is the binding of DnaA's Domain IV with nine-nucleotide signatures on double-strand DNA(three of them, shown in red and yellow in Figure 2.14). This binding will open a pocket for DnaA's Domain III to bind with AT rich region on single-strand DNA following the nine-nucleotide signatures to build a filament of DnaA on single-strand DNA. The final step is the binding of replisome with the chromosome after the filament grows above 15 DnaA with 30 DnaA maximum due to the length of AT rich region is 90. In current model, the assembly of replisome is not explicitly included.

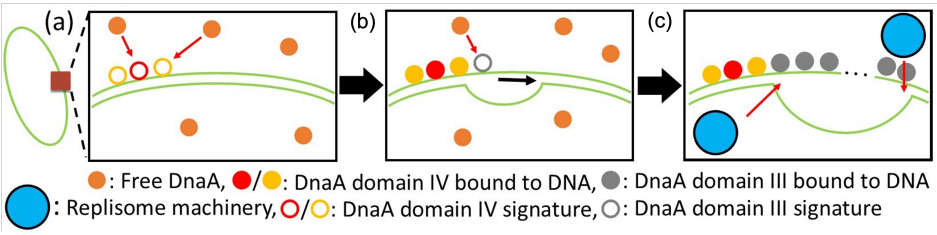


Figure 2.15: Three stage DNA replication initiation

Replication occurs gene by gene when replisomes move along the circular chromosome from oriC to ter in two directions. Since the locations of genes are fixed, the replication happens in order. So our replication model is ordered and bidirectional.

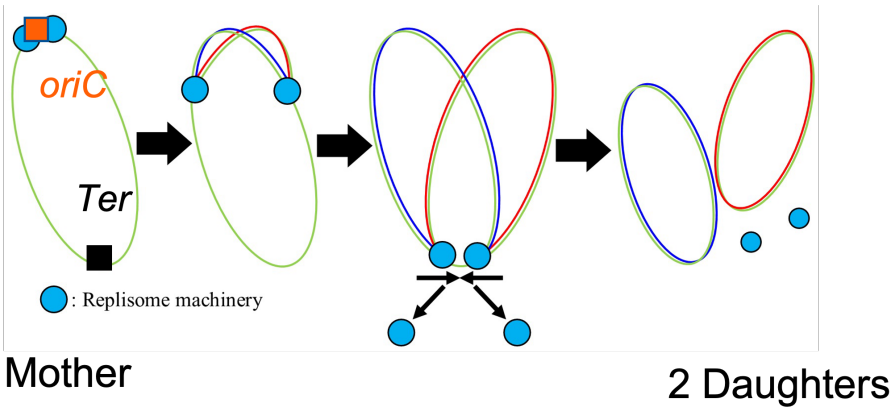


Figure 2.16: Ordered and Bidirectional Replication

The equivalent reaction to duplicate certain gene is in Table 2.3 The rate to replicate one gene in Table 2.4 is determined by a polymerization rate form - Hofmeyr rate law [9].

Table 2.3: Reactions and Rates for Replication, Transcription, Translation and Degradation

Processes	**Reactions**	**Kinetic Constant**
Replication	$G_{\text{locusNum}} \rightarrow 2G_{\text{elongation}}$	$k^{locusNum}_{\text{replication}}$
Transcription	$RNAP + G_{\text{locusNum}} \rightarrow RNAP : G_{\text{locusNum}}$	$k^{\text{binding}}_{\text{trsc}}$
	$RNAP : G_{\text{locusNum}} \rightarrow R_{\text{locusNum}} + RNAP + G_{\text{locusNum}}$	$s^{locusNum}_{\text{trsc}} k^{elongation}_{\text{trsc}}$
Translation	$Ribosome + R_{\text{locusNum}} \rightarrow Ribosome : R_{\text{locusNum}}$	$k^{\text{binding}}_{\text{trans}}$
	$Ribosome : R_{\text{locusNum}} \rightarrow P_{\text{locusNum}} + Ribosome + R_{\text{locusNum}}$	$k^{elongation}_{\text{trans}}$
Degradation	$Degradosome + R_{\text{locusNum}} \rightarrow Degradosome : R_{\text{locusNum}}$	$k^{\text{binding}}_{\text{degra}}$
	$Degradosome : R_{\text{locusNum}} \rightarrow NMPs + Degradosome$	$k^{depoly}_{\text{degra}}$

Transcription, translation and degradation are all depicted as a two-step binding and (de)polymerizatoin reactions where the polymerization in transcription and translation shares the same rate form as that in replication. The rate for degradation from a mRNA to its monomers is calculate by divide the monomer depletion rate over the length of mRNA.

Table 2.4: Rate Form for Replication, Transcription, Translation and Degradation

Processes	**(de)polymerization Constants**	**Formulation**
Replication	$k^{\text{replication}}_{\text{elongation}}$	$\frac{k^{\text{cat}}_{\text{replication}}}{\frac{K_{D1}K_{D2}}{[dNTP_1][dNTP_2]} + \sum_i \frac{K_{Di}}{[dNTP_i]} + L_{DNA} - 1}$
Transcription	$k^{\text{trsc}}_{\text{elongation}}$	$\frac{k^{\text{cat}}_{\text{transcription}}}{\frac{K_{D1}K_{D2}}{[NTP_1][NTP_2]} + \sum_i \frac{K_{Di}}{[NTP_i]} + L_{RNA} - 1}$
Translation	$k^{\text{trans}}_{\text{elongation}}$	$\frac{k^{\text{cat}}_{\text{translation}}}{\frac{K_{D1}K_{D2}}{[tRNA{:}aa_1][tRNA{:}aa_2]} + \sum_i \frac{K_{Di}}{[tRNA{:}aa_i]} + L_{protein} - 1}$
Degradation	$k^{\text{degra}}_{\text{depoly}}$	$\frac{k^{\text{degra}}_{\text{cat}}}{L_{mRNA}}$

2.3.3 Metabolism and Rate Law

As mentioned, replication, transcription, and translation all require monomers (deoxyribonucleotide (dNTPs), ribonucleotide (NTPs), and amino acids (AAs)) for the polymerization reactions.
All the supply of monomers comes from metabolism of JCVI-syn3A. The (d)NTPs are generated in nucleotide metabolism that transport extracellular nucleoside and convert nucleoside to (d)NTPs used in the DNA and RNA synthesis. The other building blocks, such as phosphoribosyl pyrophosphate (prpp), phosphoenolpyruvate (pep), and 1,3-Bisphosphoglyceric acid (1,3dpg) along these pathways are the products in glycolysis in central metabolism 2.17.

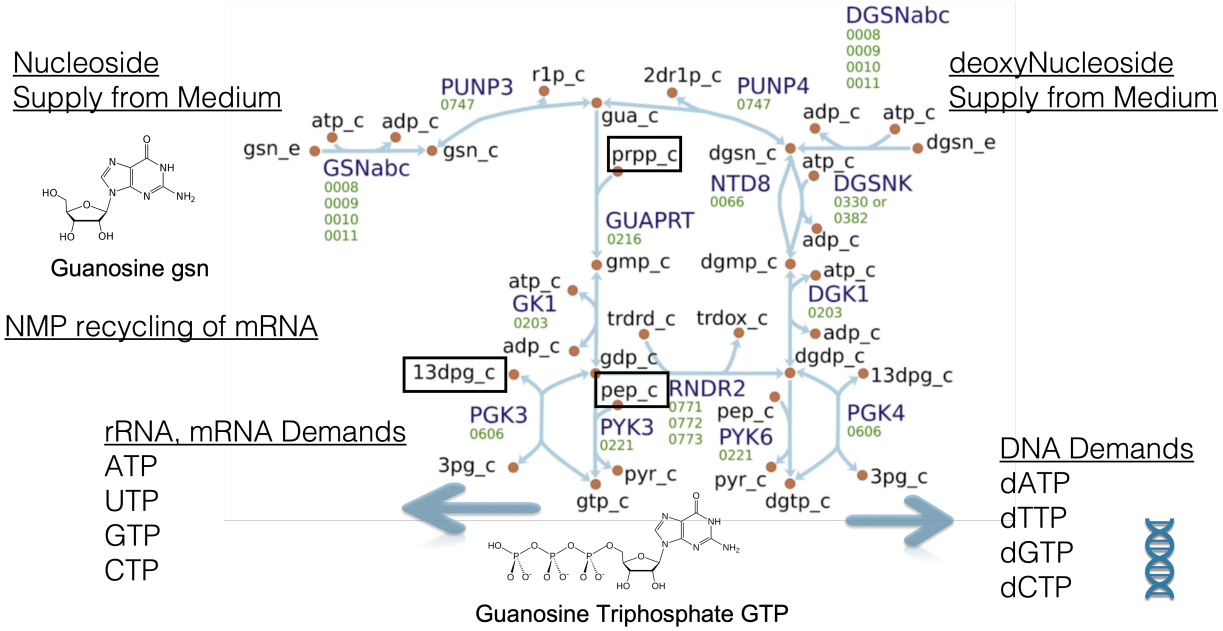


Figure 2.17: Supply of GTP and dGTP for DNA and RNA synthesis in nucleotide metabolism. The metabolites in the bracket come from central metabolism

The whole metabolism network can be separated to five connected sub-networks, including central, nucleotide, lipid, cofactor and amino acid. Glucose 6-phosphate (g6p) the immediate downstream of glucose in glycolysis connects central and lipid metabolism, while Adenosine triphosphate (ATP) that generated in glycolysis provides energy for reactions in all five sub-networks.

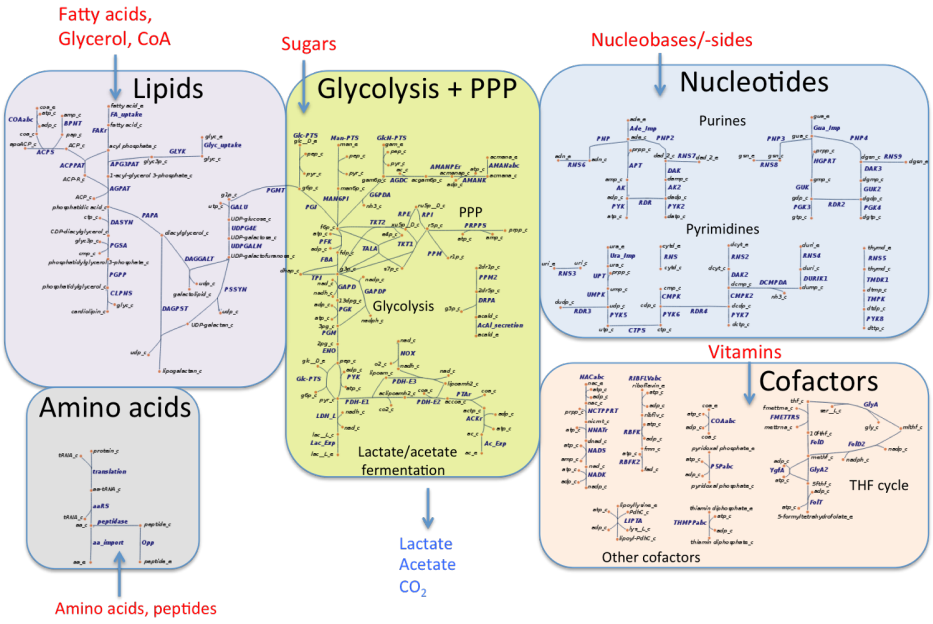


Figure 2.18: Whole Metabolism of JCVI-syn3A with five subsystems: central, nucleotide, lipid, co-factor and amino acid

In our current whole cell modeling, 197 metabolites are included, 148 cytoplasmic and 49 extracellular. 175 reactions connect the metabolites. Shown in the network of central metabolism, the orange nodes are the metabolite IDs. We use suffix '_c' to denote cytoplasmic and '_e' for extracellular. The blue arrows are reactions connecting different metabolites, with names in dark blue and related proteins' locusNums under the names.

ATP as the major energetic molecules energize cellular processes, without which the cell will die. In JCVI-syn3A, glycolysis is the only way to generate ATP. Phosphoenolpyruvate (pep) is an intermediate in both the upstream and downstream in glycolysis.

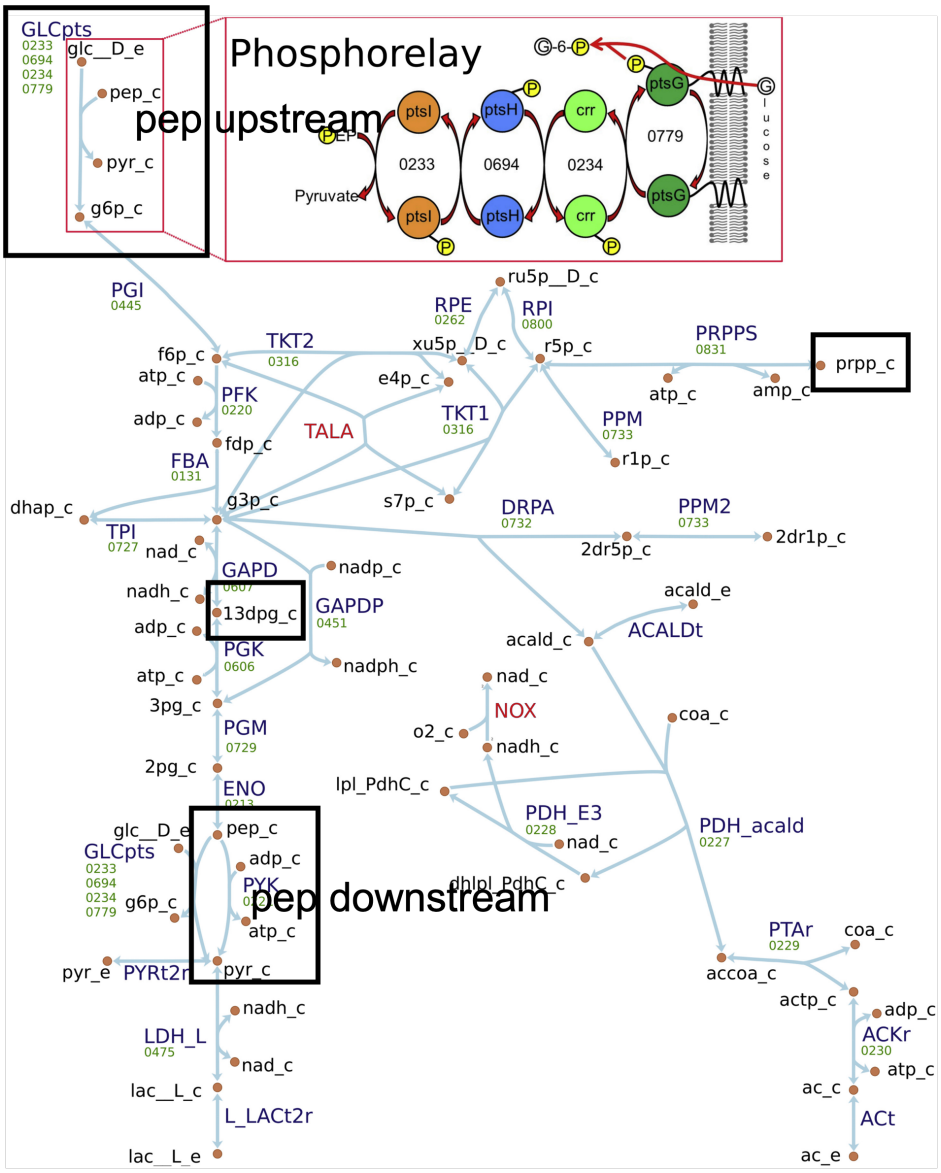


Figure 2.19: Central Metabolism in JCVI-syn3A. Phosphoenolpyruvate (pep) is both the upstream and downstream of glycolysis. 1,3-Bisphosphoglyceric acid (1,3dpg), Phosphoribosyl pyrophosphate (prpp) and pep are in nucleotide metabolism.

Convenience Rate Law

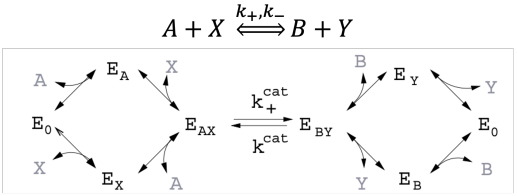
Most reactions in the metabolism requires certain proteins produced in genetic information processes as enzymes or transporters. Convenience rate law and random binding model are used to depict the kinetics of these reactions [10].

The random binding model assumes the substrates bind to the enzyme/transporter in arbitrary order and are converted into the products, which then dissociate from the enzyme in arbitrary order. Convenience rate law further assumes the conversion step is the rate-determining step and

the binding reactions are in quasi-equilibrium. For reaction

$$\mathrm{A} + \mathrm{X} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \mathrm{B} + \mathrm{Y}$$

, the rate law is



$$flux = \frac{k_+[E]\frac{[A]}{K_{M,A}}\frac{[X]}{K_{M,X}} - k_-[E]\frac{[B]}{K_{M,B}}\frac{[Y]}{K_{M,Y}}}{\left(1+\frac{[A]}{K_{M,A}}\right)\left(1+\frac{[X]}{K_{M,X}}\right)+\left(1+\frac{[B]}{K_{M,B}}\right)\left(1+\frac{[Y]}{K_{M,Y}}\right)-1}$$

K_A dissociation constant, k_+ and $k_$forward and backward rate

Figure 2.20: Mechanism of reaction A+X to B+Y in random binding model. [E] is the concentration of enzyme, [A] concentration of molecule A

Similar to genome information, the metabolic network is also stored in standard Systems Biology Markup Language (SBML) file. In the given SBML file of JCVI-syn3A, there are two compartments, cellular and extracellular with metabolites and reactions. However, SBML file itself does not store kinetic parameters and we choose to store them in a Excel file with multiple sheets for the simulation.
We can look at one reaction entry in the SBML file for more insights. Go to the SBML file and search DGSNabc. DGSNabc, the irreversible transport of deoxyguanosine (one of RNS ABC import system) in nuclotide metabolism for example, has three reactants and four products with four proteins involved. The geneProductAssociation is AND, meaning four proteins are all needed to perform this reactions. Biologically, four proteins are four subunits of nucleoside ABC transporter.

2.3.4 Hybrid CME/ODE Algorithm

As mentioned in the Chemical Master Equation, the discreteness and stochasticity of chemical kinetics play a role when numbers of reactants are now significantly high. For the species in genetic information processes, gene's number approximately is 1 or 2 and for most proteins, their counts are less than 1000. Based on the initial volume of JCVI-syn3A (200 nm radius, 0.035 fL, 1 particle count equals 50 nM), the concentrations of genes are 50 or 100 nM, of proteins less than 50 μM. All of these are dramatically lower than the concentrations of metabolites.
Therefore, it is necessary to simulate the kinetics in genetic information processes with stochastic chemical master equation (CME) and metabolism with deterministic ordinary differential equation (ODE).
To simulate the co-evolution of genetic information processes and metabolism, the communication needs to be performed to describe the interactions between these two subsystems.
Our current implementation of communication was first carried out in Yeast's galactose switch modeling published by our group in 2018 [11]. We first discretize the entire simulation length into piecewise communication time steps (hook intervals, t_H). During each communication time steps, t_H length CME simulation is performed to describe the kinetics in genetic information processes,

Table 2.5: Concentrations of Species in Genetic Information Processes and Metabolism

	Numbers (Discrete)	**Concentrations (Continuous)**
G_0001	1 or 2	50 or 100 nM
R_0001	< 10	< 500 nM
P_0001	$100 \sim 10^3$	5 μM $\sim$ 50 μM
Ribosome, degradosome	$100 \sim 10^3$	5 μM $\sim$ 50 μM
ATP	$\sim 10^6$	$\sim$ 5 mM

followed by the communication from CME to ODE by passing the counts of proteins, consumption of monomers (dNTPs, NTPs, and AAs) and recycling of nucleoside monophosphates (NMPs) in mRNA degradation. Then t_H length ODE simulation is performed and the updated cell volume and concentrations of monomers passed to the CME.

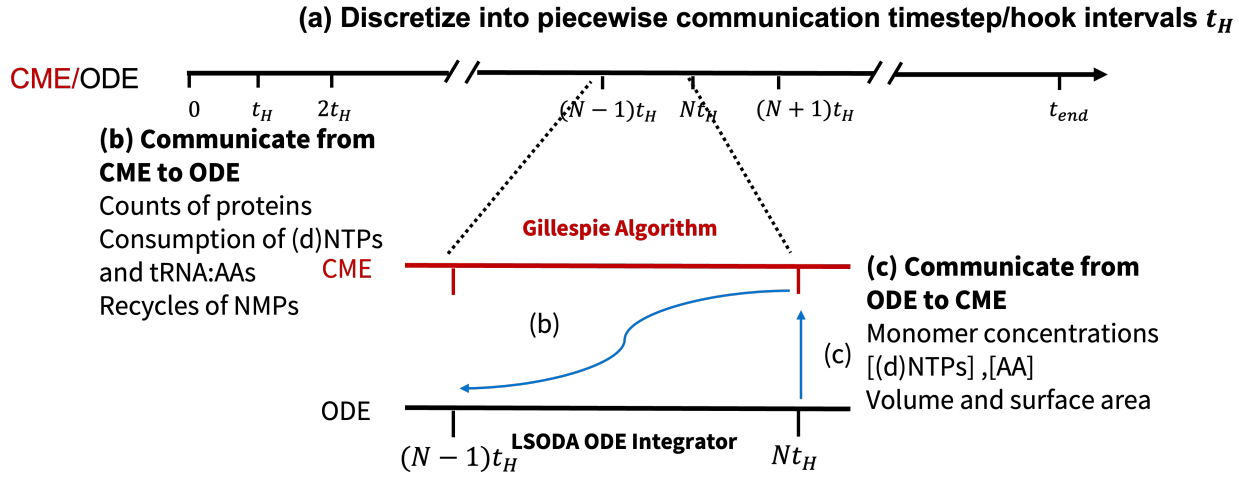


Figure 2.21: Communication Between CME and ODE to simulate the co-evolution of Genetic Information Processes and Metabolism.

In our current model, 6300 seconds' entire cell cycle is discretized with communication step 1 second.

Flowchart of CME/ODE WCM

Here is a detailed flowchart on the communication between CME and ODE. One extra thing to notice is that the CME rates are also updated per second after the metabolism simulation in ODE to account for the possible shortage of monomers that will decrease the rate in genetic information process.

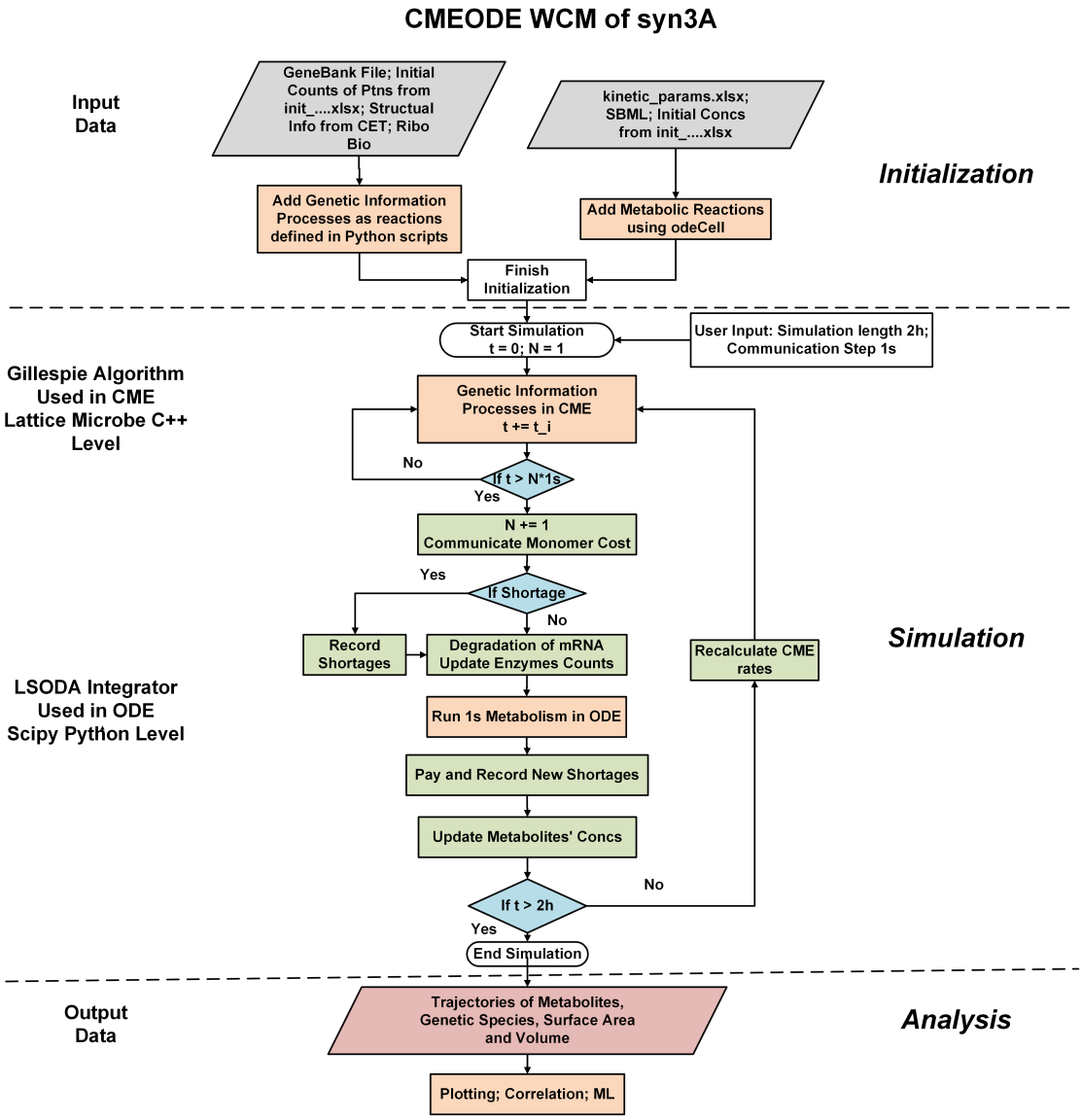


Figure 2.22: Flowchart of one CMEODE WCM simulation.

2.3.5 Results Analysis

Now run **pkl.ipynb** and then **plots.ipynb** on the Jupyter Notebook webpage to conduct the analysis. You should be able to see the identical plots in the folders.

Cell Growth

In our current simulation, we use 105 minutes as the cell cycle. Lipid molecules in metabolism and membrane proteins contributes to the membrane surface area. For the volume increase, we assume the cell remains spherical until the volume doubles. Based on the surface area, the radius then the volume is calculated. Most of the cells double the volume around 1 hour, between 50 to 70 minutes.

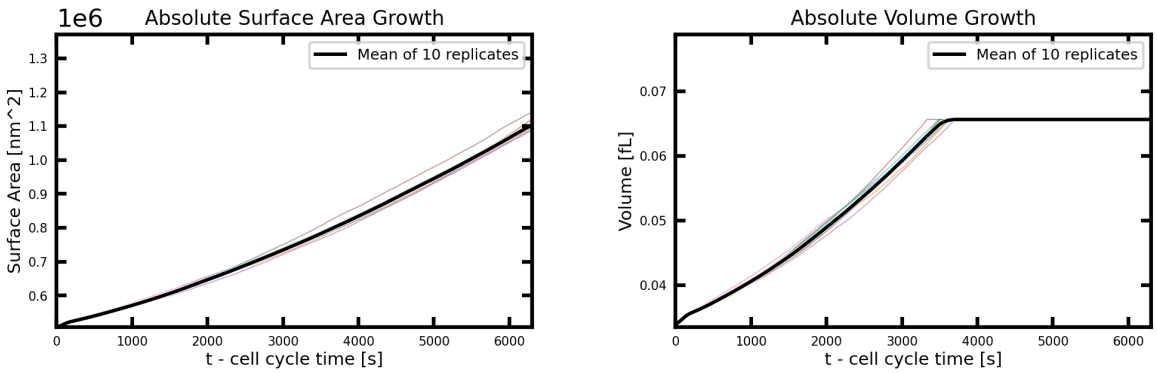


Figure 2.23: Cell surface area and volume increase.

Replication

For the replication, initiation can happen maximum 7 times during the whole cell cycle and the mean is 4.4. In this current model, we only allow replication initiation occurs after the replication finishes.

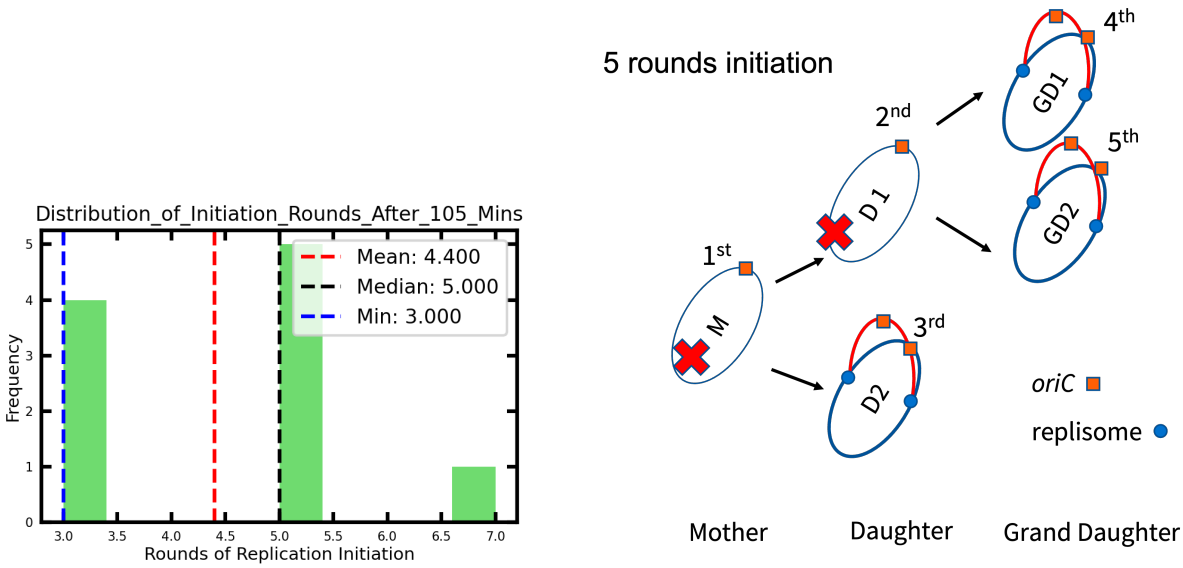


Figure 2.24: Distributions of rounds of initiation and the one possible theta structure of circular chromosomes of 5 rounds of replication initiation.

For 5 rounds of replication initiation, the data structure of chromosomes are shown in the right side of Figure 2.24. When the mother chromosome replication initiated and replicated, two daughter chromosomes was generated with no mother chromosome exits. Both daughter chromosomes initiates replication and one of two finishes the replication to generate two grand daughter chromosomes. And both two grand daughters initiates replications. So in this case, we will have three initialized but not finished replication in the cell, one daughter and two grand daughters chromosome.

The first round initiation finishes with 16 minutes with mean 5.9 minutes. And the filament length of DnaA on ssDNA can grow above 20 in Figure 2.25.

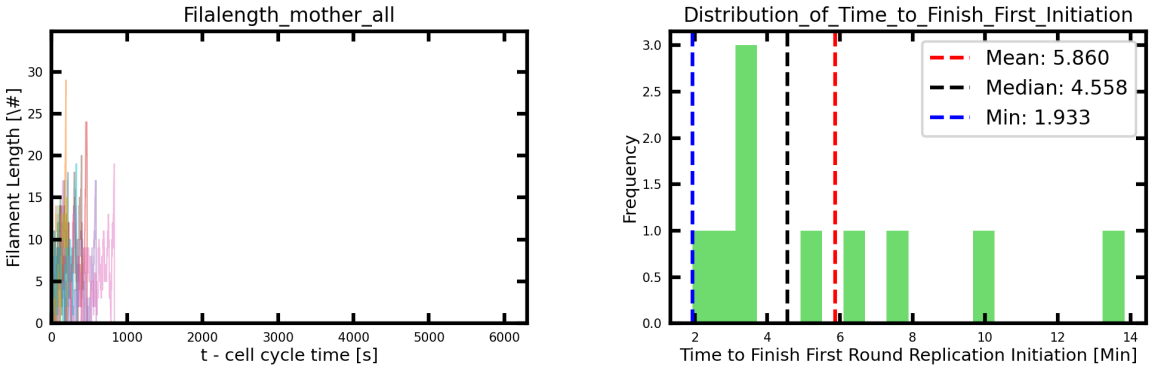


Figure 2.25: Distributions of protein counts. Left:Protein counts of 100 replicate at the end of the cell cycle. Right:Analytical Solution at steady state.

After the whole cell cycle, average gene copy number is 3.9 with minimal 2.8, meaning all genes are duplicated. It takes 40 to 60 minutes for the whole genome to be replicated.
As mentioned, the supply of nucleotide from metabolism is vital for the function of genetic information processes. In the DNA replication, the deoxythymidine triphosphate (dTTP) may encounter shortage, causing the pause of replication (See right upper picture in Figure 2.29).

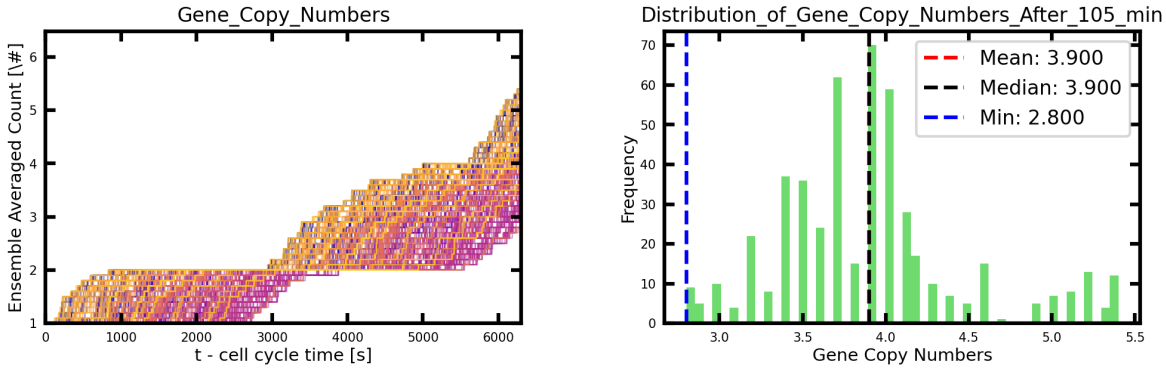


Figure 2.26: Gene Copy Numbers and the Distribution of Copy number at the end of the cell cycle.

Transcriptomics and Proteomics

In the transcriptomincs, we will see that due to the binding of degradosome and ribosomes, mRNAs seldom exist as free mRNA but under translation or degradation. Along with the gene replication, the count of all mRNAs (free, bound to degradosome, and to ribosome) also increase and doesn't reach steady state.

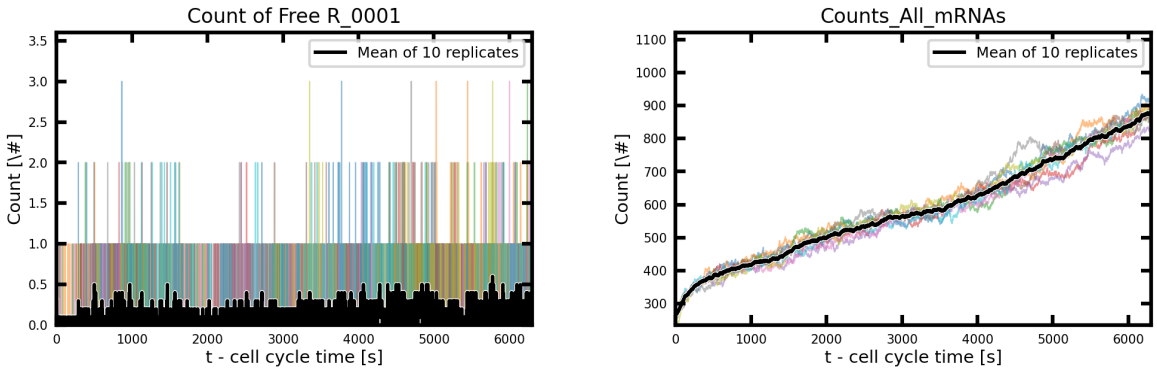


Figure 2.27: Traces of free mRNA, R_0001 of 10 replicates and the number of all mRNAs over the cell cycle.

Due to the fact that no degradation of protein considered in our current model, the protein count accumulates over time. Most proteins roughly double after one cell cycle.

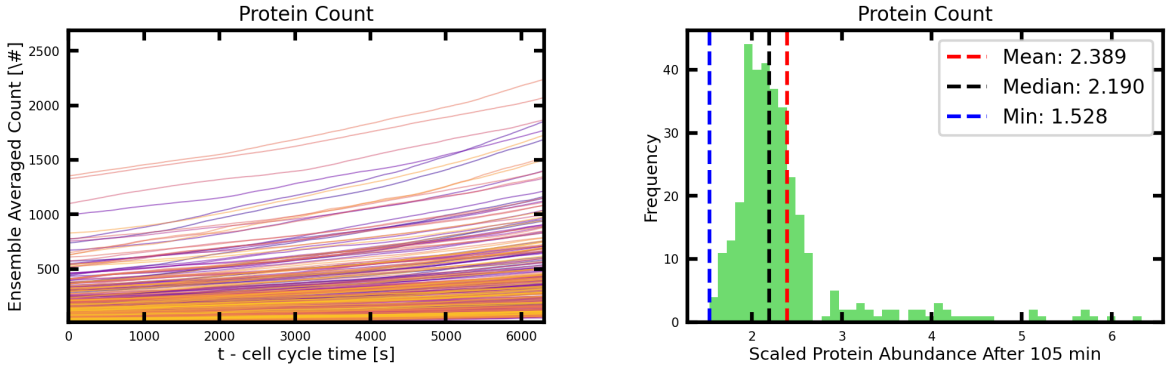


Figure 2.28: Ensemble averaged count of different proteins over the cell cycle and the distribution of scaled protein abundances.

Traces of ALL Metabolites

From our current simulation, we can also easily plot all traces of metabolites over the whole cell cycle.

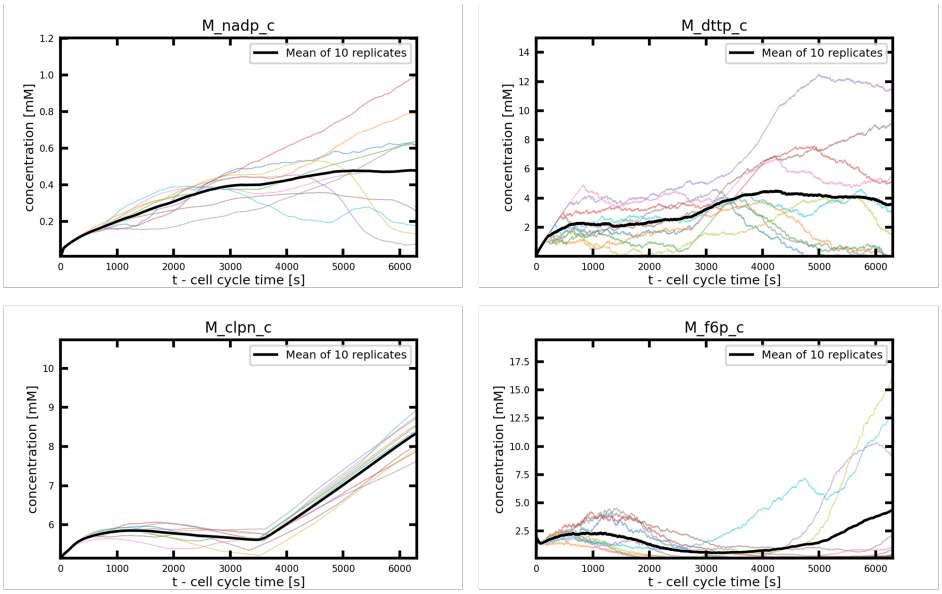


Figure 2.29: Traces of Four Metabolites Over the Entire Cell Cycle

Question

1. In this simulation, the minimal cell is assumed to maintain a spherical shape until its volume doubles, at which point cell division begins. The cell completes division once its surface area has doubled (see Figure 3E in Cell 2022 [4]). Given an initial radius of 200 nm, give the range of times at which the volume and surface area double among 10 replicates, and describe the cell cycle. The function to extract the surface area and volume traces is in `plotting.ipynb`'s block `Cell Growth`.
2. Identify the outliers in the scaled protein abundance that exceed a value of 3 at 105 minutes. Give the list of gene locusNums and their initial counts. Find the initial counts in **Comparative Proteomics** sheet in `WholeCellModel/input_data/initial_concentrations.xlsx` Excel file. The function to calculate scaled protein abundance is in Jupyter Notebook `plotting.ipynb`'s block `Proteomics`.

Chapter 3

License and Copyright

Bibliography

[1] Fujikawa, N, Kurumizaka, H, Nureki, O, Terada, T, Shirouzu, M, Katayama, T, Yokoyama, S (2003) Structural basis of replication origin recognition by the DnaA protein. *Nucleic Acids Research* 31:2077–2086.

[2] Duderstadt, KE, Chuang, K, Berger, JM (2011) DNA stretching by bacterial initiators promotes replication origin opening. *Nature* 478:209–213 Publisher: Nature Publishing Group.

[3] Peterson, J, Hallock, M, Cole, J, Luthey-Schulten, Z (2013) A Problem Solving Environment for Stochastic Biological Simulations. *Proceedings High Performance Computing Networking, Storage and Analysis Companion (SCC)*.

[4] Thornburg, ZR, Bianchi, DM, Brier, TA, Gilbert, BR, Earnest, TM, Melo, MC, Safronova, N, Sáenz, JP, Cook, AT, Wise, KS, Hutchison, CA, Smith, HO, Glass, JI, Luthey-Schulten, Z (year?) Fundamental behaviors emerge from simulations of a living minimal cell. 185:345–360.e28.

[5] Thornburg, ZR, Melo, MCR, Bianchi, D, Brier, TA, Crotty, C, Breuer, M, Smith, HO, Hutchison, CA, Glass, JI, Luthey-Schulten, Z (2019) Kinetic Modeling of the Genetic Information Processes in a Minimal Cell. *Frontiers in Molecular Biosciences* 6.

[6] Hutchison, CA, Chuang, RY, Noskov, VN, Assad-Garcia, N, Deerinck, TJ, Ellisman, MH, Gill, J, Kannan, K, Karas, BJ, Ma, L, Pelletier, JF, Qi, ZQ, Richter, RA, Strychalski, EA, Sun, L, Suzuki, Y, Tsvetanova, B, Wise, KS, Smith, HO, Glass, JI, Merryman, C, Gibson, DG, Venter, JC (2016) Design and synthesis of a minimal bacterial genome. *Science* 351:aad6253 Publisher: American Association for the Advancement of Science.

[7] Breuer, M, Earnest, TM, Merryman, C, Wise, KS, Sun, L, Lynott, MR, Hutchison, CA, Smith, HO, Lapek, JD, Gonzalez, DJ, de Crécy-Lagard, V, Haas, D, Hanson, AD, Labhsetwar, P, Glass, JI, Luthey-Schulten, Z (2019) Essential metabolism for a minimal cell. *eLife* 8:e36842 Publisher: eLife Sciences Publications, Ltd.

[8] Earnest, TM, Cole, JA, Peterson, JR, Hallock, MJ, Kuhlman, TE, Luthey-Schulten, Z (2016) Ribosome biogenesis in replicating cells: Integration of experiment and theory. *Biopolymers* 105:735–751 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bip.22892.

[9] Hofmeyr, JHS, Gqwaka, OP, Rohwer, JM (2013) A generic rate equation for catalysed, template-directed polymerisation. *FEBS Letters* 587:2868–2875 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1016/j.febslet.2013.07.011.

[10] Liebermeister, W, Klipp, E (2006) Bringing metabolic networks to life: convenience rate law and thermodynamic constraints. *Theoretical Biology and Medical Modelling* 3:41.

[11] Bianchi, DM, Peterson, JR, Earnest, TM, Hallock, MJ, Luthey-Schulten, Z (year?) Hybrid CME–ODE method for efficient simulation of the galactose switch in yeast. 12:170–176.