

SAND2003-4509
Unlimited Release
Printed December 2003

ChemCell: A Particle-Based Model of Protein Chemistry and Diffusion in Microbial Cells

Steve Plimpton
Computational Biology & Evolutionary Computing Department
sjplimp@sandia.gov

Alex Slepoy
Computational Materials & Molecular Biology Department
aslepoy@sandia.gov

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0316

Abstract

Prokaryotic single-cell microbes are the simplest of all self-sufficient living organisms. Yet microbes create and use much of the molecular machinery present in more complex organisms, and the macromolecules in microbial cells interact in regulatory, metabolic, and signaling pathways that are prototypical of the reaction networks present in all cells. We have developed a simple simulation model of a prokaryotic cell that treats proteins, protein complexes, and other organic molecules as particles which diffuse via Brownian motion and react with nearby particles in accord with chemical rate equations. The code models protein motion and chemistry within an idealized cellular geometry. It has been used to simulate several simple reaction networks and compared to more idealized models which do not include spatial effects. In this report we describe an initial version of the simulation code that was developed with FY03 funding. We discuss the motivation for the model, highlight its underlying equations, and describe simulations of a 3-stage kinase cascade and a portion of the carbon fixation pathway in the *Synechococcus* microbe.

1 Introduction

One of the next grand-challenge goals in biology, building on the deluge of genomics and proteomics data, is to understand how individual cells function as a collection of interacting biochemical molecules and molecular machines. Prokaryotic single-cell microbes are the simplest of all self-sufficient living organisms; at a macroscopic level they are small bags of biochemical cytoplasm with little internal structure. Yet microbes create and use the fundamental molecular machinery common to all life: protein synthesis via DNA transcription and ribosomal mRNA translation, gene regulation, cell membrane control of small molecule transport, signal propagation and metabolic regulation via protein networks, etc. The DOE has also taken a strong interest in prokaryotic biology [7] due to the role microbes play in bio-remediation of waste and carbon sequestration (which affects CO_2 levels and thus global warming), and the national security implications of bio-terrorism.

From a modeling perspective, microbial organisms are an ideal starting point for building cellular models because their structure is simple and they contain a relatively small number of large bio-molecules. For example, consider *Escherichia Coli* (E Coli), the most studied and best understood of all microbes. E Coli is cylindrical in shape, a few cubic microns in size, with a dozen flagella. Its single strand of DNA has 3000 genes, most of whose function is known. An E Coli cell contains on the order of a few million protein molecules, tens of thousands of ribosomes, similar numbers of tRNA and mRNA molecules, and tens of millions of small organic molecules and ions, with the remaining 70% of the cell volume being water [5]. From a high-performance computing standpoint, where simulations of 100s of millions or even billions of particles or grid cells are becoming commonplace, a simulation of an E Coli cell could potentially model all the bio-molecules in the cell (excluding water).

It is worthwhile to take note of several cell modeling efforts at other labs and universities. At the continuum end of the spectrum, the Virtual Cell project [10] uses grids to model cell interiors and solves reaction-diffusion PDEs to track species concentrations spatially and in time. In a similar vein, Sandia researchers have used the finite-element MPSalsa code and its reaction-diffusion solvers in conjunction with complex meshes of convoluted internal cell structures, such as the endoplasmic reticulum (ER), to model calcium concentrations as a function of time inside the ER [2].

Many cellular events, particularly in small microbes, occur due to one or a few genes producing a few copies of a protein, which is a concentration level not trackable by continuum representations. Stochastic cellular models that handle low concentrations of reacting species have their origin in the work of Gillespie [4]. A vector of particle counts (one value per species) is stored, along with a list of chemical reactions and rates. Particles have no spatial location; the system is assumed to be “well-mixed” so that any particle can interact with any other. At each step in the simulation, each reaction is assigned a probability based on the reactant concentrations and the reaction rate. Random numbers are used to choose which reaction occurs and at what time interval. Time advances and the vector of particle counts is updated to account for reactants becoming products due to the selected reaction. Several strategies for making Gillespie-style stochastic simulations very efficient have been formulated [3]. Such models were used by Arkin and collaborators to study complex

gene regulatory networks, such as the phage λ lysis-lysogeny decision network in E Coli [1].

Larry Lok at The Molecular Sciences Institute (TMSI, a collaborator in our modeling effort, see Section 6) has implemented a Gillespie-style model called *Moleculizer* which does not require an input list of chemical reactions to run a simulation. Instead, rules are specified which describe binding sites on macro-molecules and what states the molecule may be in (e.g. phosphorylation or methylation state). A simulation is initiated with particle counts for a few simple species. Over time, new species are created on-the-fly as binding sites are occupied and large protein complexes form. When a new species comes into existence for the first time, all the association and dissociation reactions it can participate in are also enumerated as possible reaction choices and assigned estimated rates. The advantage of this approach is that for reaction networks involving protein complexes with several constituent proteins, each of which can be in several states, there can be a combinatorial explosion of intermediate species and reactions that must be tracked; *Moleculizer* does this without the need to pre-compute all the possibilities or the need to identify the interesting subset.

The *StochSim* model written by Bray’s group at Cambridge University [11] is also a stochastic simulator but models individual particles (proteins, protein complexes, etc). Each timestep, a pair of reactants (or a single reactant) is picked randomly and reacted at the appropriate rate. As before, the “well-mixed” assumption holds, so that Gillespie-style stochastic results are produced. *StochSim* has been used to model the bacterial chemotaxis effect in E Coli where the cell senses a chemical gradient in its extra-cellular environment and performs a series of protein-based chemical reactions to change its flagellar motor control and alter its motile behavior. In his thesis work, Shimizu enhanced *Stochsim* to include limited spatial molecular information [12], which is possible because *Stochsim* tracks individual particles. Specifically, membrane receptors in the chemotaxis pathway were allowed to cluster in different orientations and interact cooperatively.

Finally, the cell modeling effort most similar to our methodology is the *MCell* project of Stiles and Bartol [8]. They track individual particles (proteins, organic molecules, etc) moving via Brownian motion within a cellular volume that includes membrane surfaces. The surfaces are triangulated so they can be created with considerable geometric complexity to faithfully capture internal or external membrane topologies. Diffusing particles do not interact with each other in the *MCell* model, but each surface can be embedded with receptor patches that react with particles when they collide with the surface. Collisions are detected by ray-tracing the particle’s path thru nearby surface elements. *MCell* has been used with great success to model the bio- and electro-chemistry of neuromuscular junctions where the geometry of the junction surfaces plays a critical role [13].

The cellular model we describe in this report has similarities and differences with respect to the models just described. In our model each bio-molecule is represented as a single “particle” within a cell. As in all the stochastic models, a particle could be a single protein or protein complex (in a particular state) or a small organic molecule. The cell itself is represented as a simple geometric volume with semi-permeable internal and external boundaries that capture some of the biological characteristics of cell membranes. The model is

hierarchical in the sense that internal boundaries represent compartments (e.g. a nucleus or organelle) that can have their own external membranes and internal sub-compartments. Unlike MCell, the boundaries in our model are geometrically simple (e.g. a sphere), so they are more appropriate for prokaryotic cells than for eukaryotic cells with complex internal structure.

As time advances in the simulation, particles diffuse independently via Brownian motion, within an implicit background of water and small organic molecules. Compartment boundaries affect the motion as particles may be constrained to move within a membrane or compartment volume or be allowed to permeate from one compartment to the next. Each step, particles also interact with nearby particles (not with the membranes) in a probabilistic sense. As in several of the models described above, the rules for particle-particle interactions are encoded in chemical rate equations (e.g. $A + B \rightarrow C$) with specified rate constants.

From a computational standpoint, this model is similar to other large-scale particle codes developed at Sandia and elsewhere - e.g. molecular dynamics, direct-simulation Monte Carlo (DSMC), and electromagnetic particle-in-cell (PIC) models. Computational issues that have been addressed in those models, such as Monte Carlo rules for probabilistic interactions, efficient neighbor finding, and parallel implementation, can be brought to bear on the cellular model as well. From a biology standpoint, the challenge is to represent cellular function with model inputs (diffusion constants, list of reactions, reaction rates) that enable both qualitative and quantitative comparison of simulation results with experiment, and which ultimately leads to biological insight. This is a difficult challenge, since many model inputs are often not known to high fidelity.

This report describes our initial implementation of this model. While still preliminary in some respects, it does capture several features thought to be important for modeling prokaryotic cell biochemistry. It allows for arbitrarily low concentrations of a particular particle species. Particles are tracked spatially which allows for concentration gradients and for diffusion-limited reaction effects to be captured. Particles can be localized within compartments, which cells use in conjunction with semi-permeable membranes to isolate and concentrate desired reactants. Similarly, membranes are modeled as two-dimensional diffusion/reaction environments with permeability options, which mimics their true biological behavior. Diffusing particles interact with other diffusing particles, as a model of solution-based chemistry.

The remainder of this report proceeds as follows. In the next section we provide details of our model. In section 3 we describe the simulation options and commands in a user's guide format. In section 4 we present the results of several simulations performed with the new code. Finally, in section 5 we highlight features we plan to add to the simulation model.

2 Model

In this section we describe a simple cellular model which can represent individual macro-molecules and their interactions with each other. The model has several components: particles, cellular geometry, particle motion, and particle interactions. We discuss each of these in turn, then describe how they are combined in the model to create a simulation.

As described in the previous section, the number of “interesting” macro-molecules in a microbial cell is often no more than a few million. By “interesting” we mean those that are participating in a reaction network one is attempting to model. Of course, for simplified models of small portions of a full network, the particle count may be much smaller. In our model, an individual particle represents a single macro-molecule. It could be an individual protein, a protein complex (e.g. a membrane-bound ABC transporter), a molecular machine (e.g. a ribosome), a portion of a DNA strand (e.g. a gene or operon), or a smaller organic molecule that will bind to a protein target. Particles in the model have an x,y,z position and a type (or species). A particular protein or complex that exists in several states (e.g. due to phosphorylation) can be represented by different types.

Cellular geometry is represented in a very simplified form in our model. This is due in part to our focus on prokaryotic cells which have limited internal structure, and in part to our focus on particle-particle interactions rather than particle-geometry interactions (e.g. with a convoluted membrane surface). Cells are represented as a hierarchy of compartments. An individual compartment is either a 3d sphere or a 2d membrane on the surface of a 3d sphere. (Additional compartment geometries are planned; see Section 5). In a topological sense, compartments can be placed inside other compartments which is what is meant by hierarchy. For example, Figure 1 is a 2d representation of a 3d model where a cell with an outer membrane contains a nucleus (with its own membrane) and another organelle. In this hierarchy, the nuclear volume is inside its membrane, which is inside the cell cytoplasm, which is inside the cellular membrane. The organelle is simply inside the cytoplasm. Relatively complex geometries can be built up using the rule that each compartment is inside either 0 or 1 other compartments.

Particles in our model move via Brownian motion and each species is assigned a diffusion coefficient. If the coefficient is infinite, the particle simply moves each timestep to a new random location anywhere within its compartment. The more interesting (and common) case is when the diffusion coefficient is finite. Since the cytoplasm is densely packed with macro-molecules, smaller organic molecules, and water, timesteps in the model are large compared to the time between molecular collisions. Particles undergo a random walk due to such collisions, which for large timesteps gives rise to diffusive behavior.

If a particle with diffusion coefficient D is at the origin at time 0.0 and diffuses for a time t , the probability it ends up at position (x,y,z) is given by

$$P(x, y, z, t) = \frac{1}{(4\pi Dt)^{3/2}} \exp\left(\frac{-r^2}{4Dt}\right)$$

where $r^2 = x^2 + y^2 + z^2$. The average distance \bar{r} a particle moves each timestep t is given by the integral of $r * P()$ over all volume which yields

$$\bar{r} = \frac{4}{\sqrt{\pi}} \sqrt{Dt}$$

$P(x, y, z, t)$ is the product of three 1d Gaussians, which means the movement of a diffusing particle can be computed with 3 random displacements in x, y, z , each sampled from a Gaussian distribution with

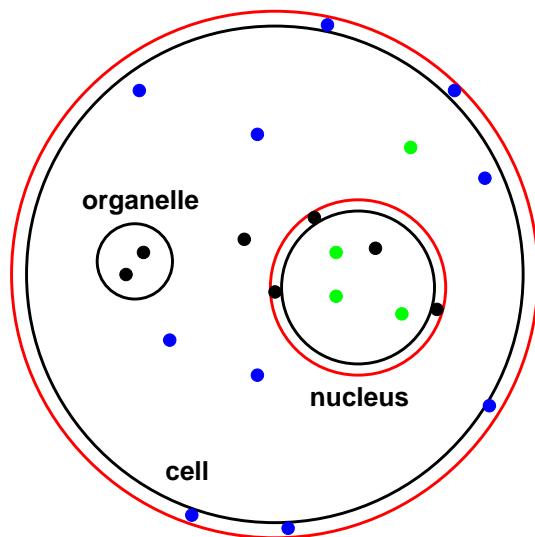


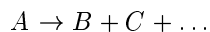
Figure 1: A 2d representation of a simple 3d cellular geometry with several compartments. The red circles are membranes around the cell and nucleus. An organelle without a membrane is also shown. Particles of various species (colors) diffuse within and between compartments.

the appropriate $\sigma = \sqrt{2Dt}$. This computation is performed each timestep for each particle. For particles in membranes, a 2d version of this computation is performed, and the particle diffuses within the zero-thickness membrane.

For 3d motion, the new position of the diffusing particle is tested to determine if it is outside its compartment or inside an interior compartment. If not, the move is finished. If so, movement from one compartment to another can be assigned a permeability factor from 0.0 to 1.0 (see Section 3). A random number is generated to determine whether the particle permeates from the original compartment to the new one. If it moves to the new compartment then it is placed on the boundary between the two compartments. If it stays in its original compartment then it is also placed on the boundary, but an additional diffusive move is done for a portion of the timestep to allow it to move away from the boundary.

Particle interactions in our model are derived from an input list of chemical reactions. A chemical reaction takes one or more reactants and converts them to one or more products at a specified rate. Without loss of generality our model treats reactions with one or two reactants, since reactions with 3 or more reactants can be written as a series of simpler reactions.

Consider the one-reactant reaction



with a rate constant k in units of 1/time. If there are N particles of species A in the simulation, then in

a timestep of size t , the number that should perform the reaction is Nkt . In our model, a decision for each A particle is made by performing the reaction if a random number is less than kt . If the reaction occurs, particle A is deleted, and new particles B, C, ... are created at A's location.

Now consider the two-reactant reaction



with a rate constant k in units of 1/molarity-time. For a well-mixed system of volume V with N_A particles of species A and N_B particles of species B, the number of reactions N that will occur in time t is given by

$$N = \frac{kN_A N_B t}{A_v V}$$

where A_v is Avogadro's number. We desire the same number of reactions to occur in our spatial system each timestep (on average), between pairs of nearby particles. One simple way to do this is to consider all A,B pairs closer than some cutoff distance R as potential reaction partners and react each of them with a probability F so as to produce N reactions. In a snapshot of a well-mixed dilute system, for small R , the number of A,B pairs closer than R apart is given by $N_P = N_A N_B V_R / V$ where $V_R = \frac{4}{3}\pi R^3$. If a fraction P of them react in a timestep, then $N = P N_P$. Setting the two expressions for N equal yields an equation that is independent of V , N_A , and N_B , namely

$$P = \frac{kt}{A_v V_R}$$

As expected, this means that for a given k and t there is an inverse relationship between the choice of R and P . The larger R is chosen, the smaller P must be set to induce the correct number of reactions. Conversely, the smaller R is chosen, the larger P must be. Since P cannot exceed one, this sets a lower limit on R . Note that the choice of P and R can be different for each reaction. This leaves considerable freedom for setting P and R values in our model. Most generally, the reactivity cutoff distance R should be a function of the diffusion rates for A and B. This is an option we are exploring; see Section 5. In the current version of the code, the user sets a maximum P_m for any reaction (e.g. 0.5) and the simulation sets a universal R value for all reactions that gives the desired P_m for the fastest reaction. The P values for all the other reactions are set accordingly to values less than P_m . Note that these same reaction rules are used (and assumed valid) for non-homogeneous systems where the well-mixed assumption no longer holds.

A simulation using the model just described has two phases: setup and timestepping. During the setup phase, particle species and reactions are defined, as is the cellular compartment geometry and topology. In the setup phase particles with desired counts and spatial distributions are also created within specified compartments.

The timestepping portion of the simulation proceeds in the following way. Each timestep consists of 3 stages: particle motion, neighbor finding, and reactions. At the end of each timestep, output of various statistics or particle snapshots can (optionally) be performed.

The particle motion stage moves each particle to a new position within its compartment (or to the boundary of a neighboring compartment) using the Brownian motion rules described above. A key point is that each particle moves independently of all others. The next stage is to find pairs of nearby particles. This is done by binning the particles into 3d bins whose size is equal to the reaction cutoff R , described above. The binning is an $O(N)$ operation in the number of particles N . The key point is that two particles can only interact in this timestep if they are in the same or adjacent bins.

The final stage is to perform reactions by looping over the particles and checking surrounding bins for potential reaction partners. If two reactants are within a distance R of each other, a random number is generated and the reaction occurs if it is less than the corresponding P -value. If it occurs, the reactant particles are deleted, and the product particles are created. They are placed at the spatial midpoint between the 2 reactants if both reactants have non-zero diffusion coefficients. If one of the reactants has a zero diffusivity, the products are created at its location. The products are flagged so that they cannot perform a second reaction in the same timestep. After all neighbors are checked, if the particle participates in one (or more) one-reactant reactions, random numbers are used to check for that possibility as well. The assumption is that in any timestep a single particle can participate in at most a small number of reactions (few neighbors), each of which has a small likelihood of occurring. Thus treating the reactions as independent possibilities is both accurate and efficient.

3 Code Commands

The basic model described in the previous section has been implemented in a C++ code called ChemCell, that is designed to be both modular and easy to extend with new features. For example, the current version has only spherical volumes and spherical membranes as geometry options. However, the C++ classes that implement the spherical equations could be cloned to create sibling classes that implement alternate geometries (ellipsoids, capped cylinders, etc). The current version of the code is serial (runs on a single processor), but the data structures and algorithms of the code were designed with parallelism in mind. See section 5 for more discussion of these issues.

ChemCell takes an input script as its argument (“chemcell in.kinase”), which is a text file containing a list of commands. The code reads the file, one line at a time, and acts on the command. Typically the command sets some particle or geometry option within the simulation. When a “run” command is read, the simulation runs for a number of timesteps. Subsequent commands can change desired options and invoke additional “run” commands. The code exits after the last line of the input script is read.

The input script can contain blank lines. The first word of a non-blank line is treated as a command. Subsequent words (separated by white space) are arguments for the command. Any text following a '#' character is treated as a comment and ignored. Commands and their arguments must be in lower case, with the exception that user-specified IDs and filenames can include upper-case characters. Generally speaking, commands can appear in any order in the input script; exceptions are noted below.

ChemCell commands can be grouped in 4 categories as follows:

Global settings:

timestep	set the timestep size
seed	set the random number seed
dump	dump output to a file
screen	write output to the screen
max_prob	set the maximum probability that a reaction will occur
move_style	set the style of Brownian motion

Geometry commands:

compartment	define a compartment within a cell
inside	define relationship between 2 cell compartments

Particle commands:

species	name a particle species
distribution	define spatial distribution of particles in a compartment
create	create a group of particles
diffusion	set a diffusion coefficient
reaction	define a chemical reaction
permeable	set permeability for a species into or out of a compartment

Miscellaneous:

include	include another input file
run	run a simulation

We now list each command in alphabetic order, giving arguments and options for each command. Note that some commands have default values, which means they do not have to be specified in an input script unless the default needs to be changed.

COMPARTMENT COMMAND

Syntax:

compartment ID style args

ID = any string

style = "sphere" or "membrane"

args:

for style = sphere, args = x y z r
x y z = center of sphere
r = radius of sphere
for style = membrane, args = x y z r
x y z = center of membrane
r = radius of membrane

Examples:

```
compartment cytoplasm sphere 0 0 0 10
compartment outer-membrane membrane 0 0 0 10
compartment nucleus sphere 2 2 3 3.5
```

Discussion:

Create a cellular compartment to hold particles. A sphere is a 3d sphere. A membrane is the surface of a 3d sphere. All numeric values are specified in units of microns.

Related commands:

inside, permeable

CREATE COMMAND

Syntax:

```
create N species-ID comp-ID dist-ID
N = number of particles to create
species-ID = what species they are
comp-ID = what compartment to create them in
dist-ID = what spatial distribution to create them with
```

Examples:

```
create 1000 CO2 cytoplasm dist1
create 50 receptor3 outer-membrane uniform
```

Discussion:

Create N particles of a species in a compartment using the specified distribution. The species, compartment, and distribution IDs must exist before this command is used.

Related commands:

species, compartment, distribution

DIFFUSION COMMAND

Syntax:

```
diffusion species-ID value
```

species-ID = ID of a particle species
value = diffusion coefficient in cm^2/sec

Examples:

```
diffusion CO2 1.0E-5
diffusion RuBisCO 0.0
diffusion ribulose INF
```

Discussion:

Set the diffusion coefficient for particles of a given species. A value of 0.0 means the particle never moves. A value of INF means it moves to a random new location within its compartment in a timestep. The default value is 0.0 for all species. The species ID must exist before this command is used.

Related commands:

species

DISTRIBUTION COMMAND

Syntax:

```
distribution ID style args
```

ID = any string

style = "uniform" or "exp" or "gaussian"

args:

for style = uniform, args = r1 r2

r1, r2 = starting, ending coords (from 0.0 to 1.0)

for style = exp, args = r1 r2 Δ

r1, r2 = starting, ending coords (from 0.0 to 1.0)

Δ = exponent of the distribution (sign is ignored)

for style = gaussian, args = sigma r0

sigma = width of Gaussian

r0 = mean of Gaussian (from 0.0 to 1.0)

Examples:

```
distribution dist1 uniform 0.0 1.0
```

Discussion:

Create a distribution function to use in particle creation. This enables particles to be created at varying densities in different regions of a compartment. Distributions are unitless; they only get mapped to a real compartment when used with the create command.

A style of "uniform" means particles will be created with a uniform volumetric density within some sub-region of the compartment. If r1 = 0.0 and r2 = 1.0 then the entire compartment is filled. If r1 = 0.0

and $r2 = 0.5$, then no particles are created at a radius greater than half the compartment size; only the innermost half of the sphere would be filled.

A style of “exp” biases the distribution radially using the weighting factor $\exp(-\Delta r)$ where Δ is the 3rd argument listed above. The $r1$ and $r2$ arguments are the same as in the “uniform” case. Thus if $r1 = 0.5$ and $r2 = 1.0$ and $\Delta = 1.0$, then only the outer half of the spherical compartment will be filled with particles, and they will be more dense at the inner radius of the sphere ($r1$) than they are at the outer ($r2$). The larger Δ is, the larger the weighting factor that is applied.

A style of “gaussian” creates a Gaussian spread of particles centered at a specified radius ($r0$) within the spherical compartment. The width of the spread is given by the sigma parameter. Thus if $r0 = 0.5$ and $\sigma = 0.1$, then a relatively narrow radial shell of particles is created at half the radius of the spherical compartment.

Related commands:

create, compartment

DUMP COMMAND

Syntax:

dump N filename

N = dump particle snapshots every this many timesteps

filename = file to dump to

Examples:

dump 100 cell.out

dump 0

Discussion:

Dump a snapshot of particle coordinates and species types to the specified file every N steps. If $N = 0$, no dump is performed and the file need not be specified. The default for N is 0.

Related commands:

screen

INCLUDE COMMAND

Syntax:

include filename

filename = new file to take input commands from

Examples:

include reaction.list2

include all_species

Discussion:

Begin reading input commands from the specified file. When that file ends, return to the current file and continue reading. This allows long lists of reactions or species to be stored in separate files. Include files can be nested.

INSIDE COMMAND

Syntax:

```
inside comp-ID n1 n2 n3 ...
```

comp-ID = ID of parent compartment

n1, etc = IDs of one or more children compartments that are inside the parent

Examples:

```
inside membrane cytoplasm
```

```
inside cytoplasm carbox1 carbox2
```

Discussion:

Define what compartments are inside another compartment. Each compartment can have 0 or more compartments inside it. Each compartment can have 0 or 1 compartment outside it. The various compartment IDs must exist before this command is used.

Related commands:

compartment, permeable

MAX_PROB COMMAND

Syntax:

```
max_prob value
```

value = number between 0.0 and 1.0

Examples:

```
max_prob 0.2
```

Discussion:

Set the maximum probability for any reaction to occur in a given timestep. This value is used in computing the actual probability that a specific reaction will occur. See Section 2 for a discussion of how this parameter is used. The default value for max_prob is 0.5.

Related commands:

reaction

MOVE_STYLE COMMAND

Syntax:

```

move_style geom_style sample_style
    geom_style = "cube" or "sphere"
    sample_style = "uniform" or "brownian"

```

Examples:

```

move_style sphere brownian
move_style cube uniform

```

Discussion:

Set the style for how particles move as they diffuse in 3d space. The “cube” setting means they move within a small cube surrounding their current location. The “sphere” setting means they move in a radially symmetric fashion within a small sphere. The size of the cube or sphere is determined by the diffusion coefficient for the particle.

The “uniform” setting means particles move with equal probability to any location in the cube or sphere. The “brownian” setting means the new location is sampled from a Gaussian that is truncated to fit within the cube or sphere. See the discussion in Section 2 for more details.

The default `move_style` setting is `cube brownian`.

Related commands:

```
diffusion
```

PERMEABLE COMMAND

Syntax:

```

permeable species-ID comp1-ID comp2-ID value new-species-ID
    species-ID = species ID of particle in its original compartment
    comp1-ID = ID of compartment particle is moving from
    comp2-ID = ID of compartment particle is moving to
    value = fractional permeability (0.0 to 1.0)
    new-species-ID = species ID the particle becomes if it moves to comp2-ID

```

Examples:

```

permeable CO2 cytoplasm carboxysome 1.0 CO2
permeable CO2 carboxysome cytoplasm 0.5 CO2
permeable Signal interior membrane 0.1 Signal-bound

```

Discussion:

Set the permeability for a particle as it attempts to move from one compartment to another. The two compartments must be connected via an “inside” command. The permeability is directional; one value can be specified for movement from compartment A to B, and another value for B to A. The particle can become another species if it successfully moves to the new compartment, or `new-species-ID` can be the same as

species-ID. The default permeability is 0.0 for all species and compartment connections. The species and compartment IDs must exist before this command is used.

Related commands:

compartment, inside

REACTION COMMAND

Syntax:

```
reaction reac1 reac2 value prod1 prod2 prod3 ...
```

```
reaction reac1 value prod1 prod2 ...
```

reac1, reac2 = species IDs of 1 or 2 reactants

value = reaction rate

prod1, etc = species IDs of 0 or more products

Examples:

```
reaction A B 1.0e-7 C
```

```
reaction C 1.0e10 A B
```

```
reaction Receptor Signal 1.0e-8 ActiveReceptor
```

Discussion:

Define a reaction that converts reactants to products at the specified rate. All reactions must have one or two reactants. Any number of products is allowed. The value is the rate at which the reaction occurs in a well-mixed system. The rate is in units of 1/molarity-time if there are 2 reactants; it is in units of 1/time if there is only 1 reactant. All species IDs must exist before this command is used.

Related commands:

max_prob

RUN COMMAND

Syntax:

```
run N
```

N = number of timesteps

Examples:

```
run 1000
```

Discussion:

Run a simulation for N timesteps.

Related commands:

timestep

SCREEN COMMAND

Syntax:

screen N

N = print particle counts to screen every this many timesteps

Examples:

screen 100

Discussion:

Print particle counts to the screen every N steps. If N = 0, no information is written to the screen. The default for N is 10.

Related commands:

dump

SEED COMMAND

Syntax:

seed value

value = 1 to 8-digit integer random number seed

Examples:

seed 9585938

Discussion:

Set a unique random number seed that will be used to initialize the random number generator and thus determine the sequence of random numbers used for Monte Carlo decisions within the simulation. The default seed is 123456.

SPECIES COMMAND

Syntax:

species ID alias1 alias2 alias3 ...

ID = any string

alias1, etc = additional strings that refer to the same species

Examples:

species CO2

species Rap3 0x0001847 signal-protein

Discussion:

Define a species of particle that will be used in the simulation. Each species has a base ID and 0 or more aliases. In any command that uses a species ID, the base ID or any of the aliases may be used. Aliases

are useful when lists of species names are generated by other programs, sometimes in a computer-generated format. See the discussion of Molecuizer in Section 5.

TIMESTEP COMMAND

Syntax:

```
timestep value
value = size of timestep (in seconds)
```

Examples:

```
timestep 0.01
```

Discussion:

Set the size of a timestep in the simulation. The default value is 0.01 sec.

4 Results

In this section we discuss results from 3 models we have simulated with ChemCell. The first is for a very simple system, involving only the bi-directional $A + B \leftrightarrow C$ reaction. The second is for a 3-stage kinase cascade (9 species, 7 reactions) that is a simplified model for how a signaling event occurs within a cell. In both cases we compare the spatial ChemCell output with that of a stochastic Gillespie model which includes no explicit spatial representation of its particles. The third model is for a portion of the carbon fixation process in a microbial cell, involving 12 chemical species and 11 reactions.

Consider three species A, B, C undergoing two competing reactions: $A + B \rightarrow C$ and $C \rightarrow A + B$, with respective rate constants 10^{10} per molarity-sec and 1.0 per sec. The simulation is run for 1000 timesteps in a 3d spherical volume of 10^{-11} liters with initial counts of 3000 A particles and 1000 B particles. Figure 2 shows the concentration of the three species over time when each particle has an infinite diffusion coefficient (moves to a random new position within the volume each timestep). Within a second, the two reactions reach equilibrium and the species concentrations are roughly constant. The black curves in the figure are for a stochastic Gillespie simulation (zero-dimensional) of the same reactions and particle counts. The Gillespie model treats diffusion only in the limit that all particles are assumed to be well-mixed at each timestep. The two models give statistically identical results.

In Figure 3, particle count traces are shown for runs of the same model with varying particle diffusivities. The black curve is data from Figure 2 with infinite diffusivity. For $D = 1.0E-6 \text{ cm}^2/\text{sec}$ (red), the results are essentially identical. For a 10x and 100x smaller diffusivity, the curves begin to diverge, though they are asymptoting to the same equilibrium value. This is because the system has become diffusion-limited; for small diffusivities the particles do not move fast enough to find new reaction partners and so the reactions proceed more slowly.

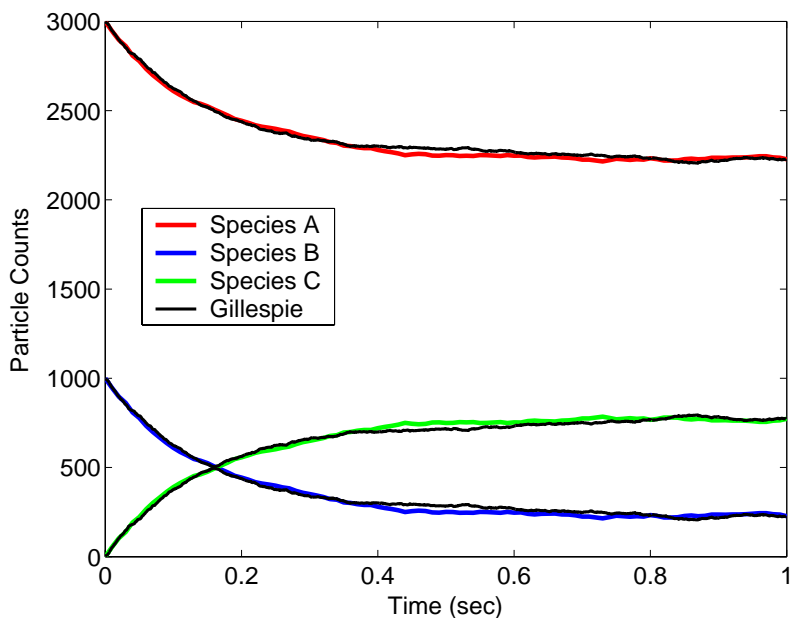
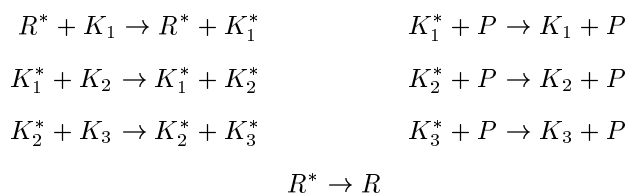


Figure 2: Particle counts versus time for A (red), B (blue), and C (green) species in a ChemCell simulation. The diffusion rate for all 3 species is infinite. Black curves are the corresponding particle counts for the same system run with a stochastic Gillespie model.

Kinase proteins often act within signaling networks within cells. Several kinase species can work in series to trigger the desired response. A 3-stage kinase cascade can be modeled as a set of 7 coupled reactions as listed below:



An activated receptor R^* will react with the first kinase species K_1 to produce an activated K_1^* . The activated K_1^* will in turn react with inactive K_2 to produce activated K_2^* . Likewise, activated K_2^* will produce activated K_3^* . Each of the activated kinase species are de-activated by a reaction with a phosphatase P molecule. Similarly, the last reaction in the list indicates that the activated receptor R^* slowly becomes inactive over time. The choice of rate constants for the various reactions determines how quickly and how strongly each of the activated kinase species will be triggered by its antecedent.

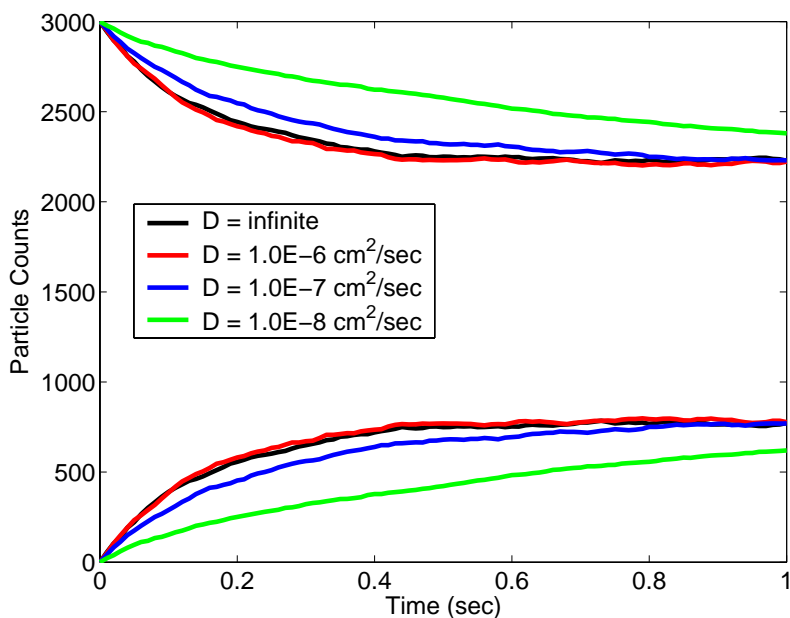


Figure 3: Particle counts versus time for the A and C species of Figure 2 simulated at different diffusivities. At $D = 1.0E-6 \text{ cm}^2/\text{sec}$, the counts track the infinite diffusivity (or Gillespie) model; at smaller diffusivities, the reaction system becomes diffusion-limited.

A plot of the particle counts for the 3 activated kinases and the originating signal receptor is shown in Figure 4 for a ChemCell simulation of 5000 timesteps in a volume of 10^{-12} liters with prototypical reaction rates assigned to each of the 7 reactions. Initially there are 1000 activated receptors. The concentration of each of the 3 kinase species spikes upward, one after the other, and their concentrations decay as the signal receptors go inactive. The colored curves are for a simulation with infinite diffusion coefficients; the black curves are the counts for the corresponding stochastic Gillespie simulation. As before, the 3d particle simulation tracks the zero-dimensional Gillespie model, with only a slight statistical variation. Both the peak heights and their positions (in time) are well-matched.

In Figure 5 the particle counts for the activated Kinase 1 species are shown for ChemCell simulations with different diffusivities. As before, with a relatively large diffusion coefficient of $1.0E-6 \text{ cm}^2/\text{sec}$, the system tracks the infinite diffusion (or Gillespie) model. For smaller diffusivities, the kinase responds more slowly to the activated receptor and thus the peak height is smaller, due to the decay in receptor counts. The other two kinase peaks exhibit similar behavior. As before, this is a signature of diffusion-limited reactions, where the kinase particles do not find reaction partners quickly enough for the appropriate reactions to occur. These results for both the simple (ABC) and kinase model indicate the importance of accounting for spatial locality (ignored in the zero-dimensional Gillespie model) in cellular systems where diffusion effects

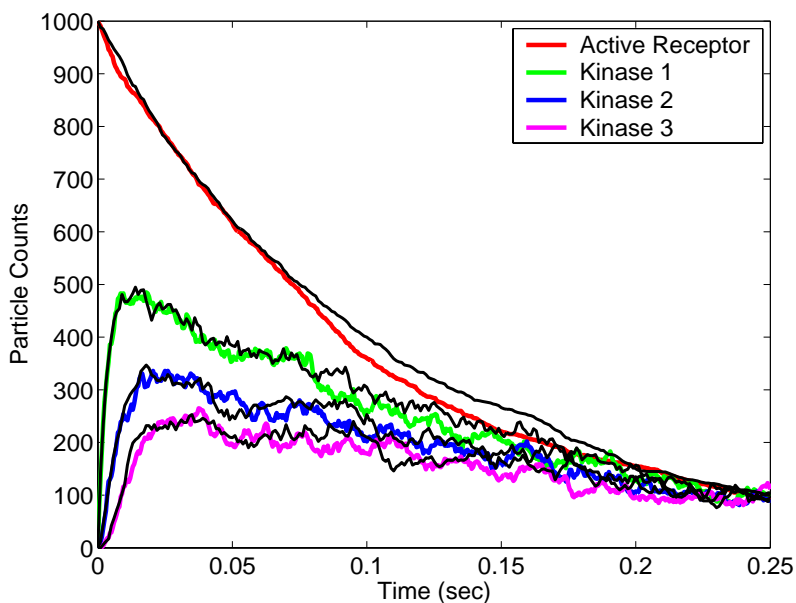


Figure 4: Particle counts versus time for the signal receptor (red) and 3 activated kinase species (green, blue, magenta) in a ChemCell simulation. The diffusion rate for all species is infinite. Black curves are the corresponding particle counts for the same reactions run with a stochastic Gillespie model.

are important.

Finally, we describe a ChemCell model that is a simplified encoding of a portion of the carbon fixation process in a microbial cell. Autotrophic organisms use photosynthesis to convert inorganic carbon into amino acids and starches for use in their metabolism. This carbon fixation process is a fundamental step at the base of the food chain and is also a factor in global warming scenarios since large amounts of CO_2 are sequestered by organisms that fix carbon.

Carbon fixation is facilitated by the RuBisCO enzyme, which has a dual reaction profile wherein it can act as a carboxylase (reacting with CO_2) or an oxygenase (reacting with O_2). This dual propensity reduces the efficiency of carbon fixation, and a variety of methods have been evolved by organisms to enhance the process. Photosynthetic marine bacteria, like *Synechococcus*, use carbon concentration mechanisms (CCMs) to create a high inorganic carbon concentration in the proximity of its RuBisCO proteins. This occurs in two stages, as indicated in the schematic of Figure 6. First, HCO_3^- (bicarbonate ion) is concentrated in the cytoplasm via transport through the cell membrane from the surrounding sea water. Second, the organisms contain small organelles called carboxysomes which contain carbonic anhydrase, an enzyme which converts HCO_3^- to CO_2 . Since the cell's RuBisCO is also inside the carboxysomes, the enzyme is thus exposed to a high concentration of CO_2 .

In our model we use two nested compartments to represent the basic geometry of the CCM, one for

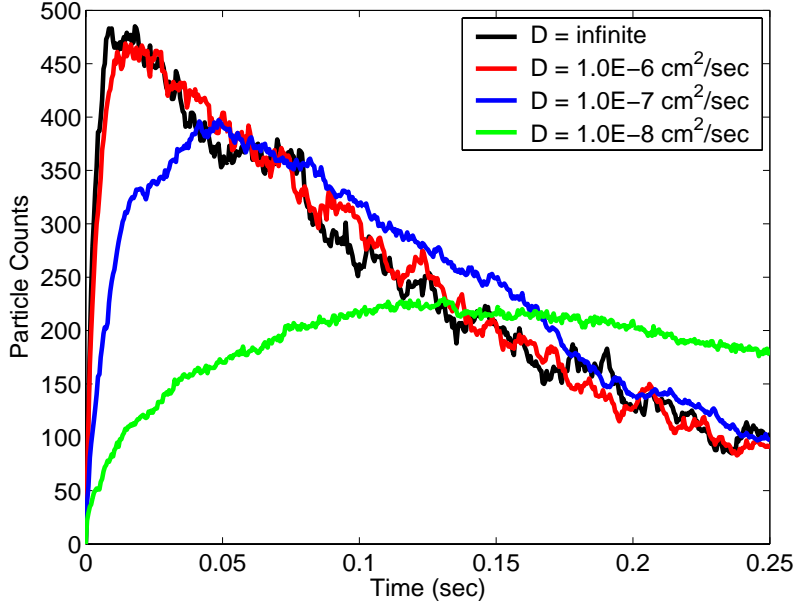
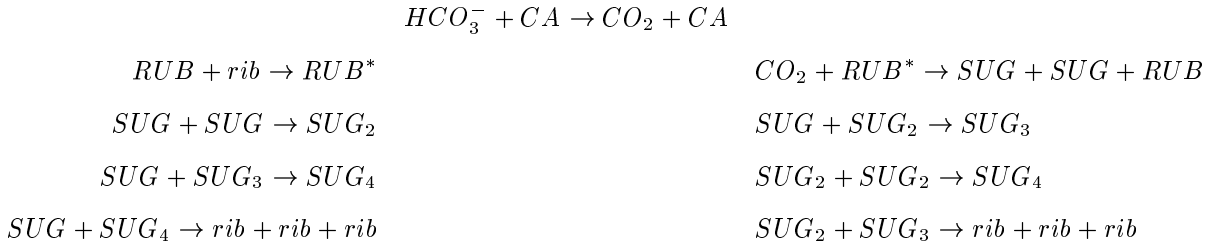


Figure 5: Particle counts versus time for the Kinase 1 species of Figure 4 simulated at different diffusivities. As before, at high diffusivity, the count tracks the infinite diffusivity (or Gillespie) model; at smaller diffusivities, the peak height is shifted in time and height due to diffusion-limited reactions.

the Synechococcus cell, and one for an internal carboxysome. The two compartments are shown outlined with extra particles in Figure 7 which is a 2d slice through the 3d simulation model. Non-diffusing carbonic anhydrase (CA) and RuBisCO (RUB) particles are placed inside the carboxysome, shown in pink and red in the figure. Initially, HCO_3^- particles (black) are placed outside the cell. The permeability of the outer cell membrane for HCO_3^- is set 10x larger for diffusion into the cell than out of the cell, mimicking the operation of the active transporters for HCO_3^- in the membrane. The HCO_3^- is also free to diffuse into the carboxysome. There it is converted by CA to CO_2 . The carboxysome membrane has a low permeability for CO_2 back into the cytosol, which is a known restrictive function of the carboxysome's proteinaceous membrane. This enable RuBisCO to enzymatically react with the CO_2 to produce organic sugars. Specifically, the 11 reactions modeled in our simplified version of the process are as follows:



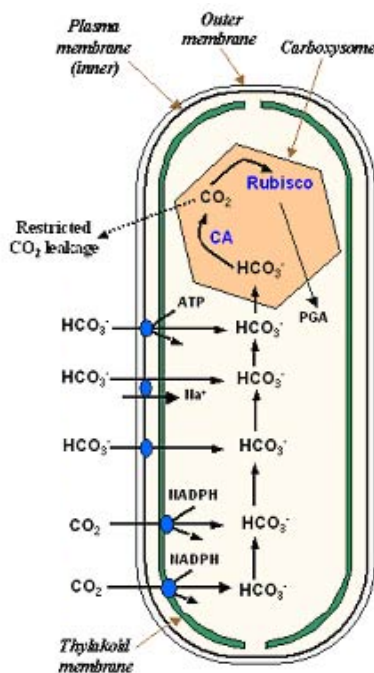
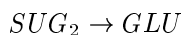


Figure 6: A schematic of the carbon concentration mechanism in cyanobacteria such as *Synechococcus*. Taken from the Price and Badger research group's WWW page at <http://www.rsbs.anu.edu.au/profiles/price.asp>.



RuBisCO (RUB) must first bind a small organic substrate, a ribulose molecule (rib) (small blue in figure), in order to become enzymatically active (RUB^*). After this occurs it can perform the carboxylase reaction to create 3-carbon sugars (SUG). The sugars react in various combinations to form two products: more ribulose substrate molecules, and 6-carbon glucose (GLU). These reactions are written above in a simplified form that ignores many intermediate steps and enzymatic partners, but does conserve the number of carbon atoms in the model.

An initial and final snapshot from the ChemCell simulation are shown in Figures 7 and 8. 200 initial HCO_3^- ions were used and the simulation ran for 50,000 timesteps. All the intermediate species are not visualized, but production of the final product (glucose) is evident in the final snapshot.

A plot of the particle counts (concentration) of the HCO_3^- , CO_2 , and glucose versus time is shown in Figure 9. HCO_3^- is depleted in the environment external to the cell and builds up in the cytosol (and carboxysome). Similarly, CO_2 concentration grows within the carboxysome, with the result that glucose is steadily produced.

This model is an over-simplification of the true carbon fixation process. Our initial implementation also

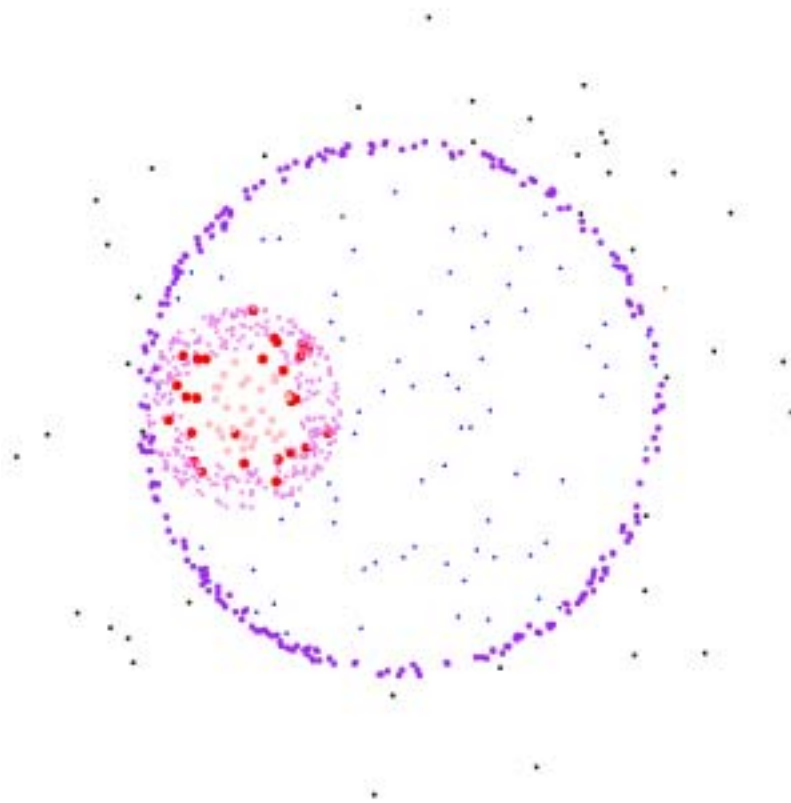


Figure 7: *Initial snapshot of ChemCell simulation of microbial cell with an inner carboxysome organelle. A 2d slice of the 3d simulation geometry is shown. HCO_3^- (black) begins outside the cell; ribulose molecules (blue) are in the cytosol; carbonic anhydrase (pink) and RuBisCO (red) enzymes are inside the carboxysome.*

involves educated guesses for diffusion rates, reaction rates, and background concentrations. However, the results are indicative of the kinds of issues and questions that can be addressed with a model that includes spatial locality at accurate length scales and involves realistic numbers of reacting particles. The simulation is able to model the influence of various carbon concentration mechanisms and track glucose production on time scales that are experimentally relevant (a few molecules per second). We are currently enhancing the simulation model to examine questions such as (1) which reaction rates and permeability factors is the glucose production sensitive to, (2) will inclusion of more compartments (for other stages of the carbon fixation process) alter the simulation dynamics, and (3) how does placement of RuBisCO versus CA in the carboxysome affect glucose production rates.

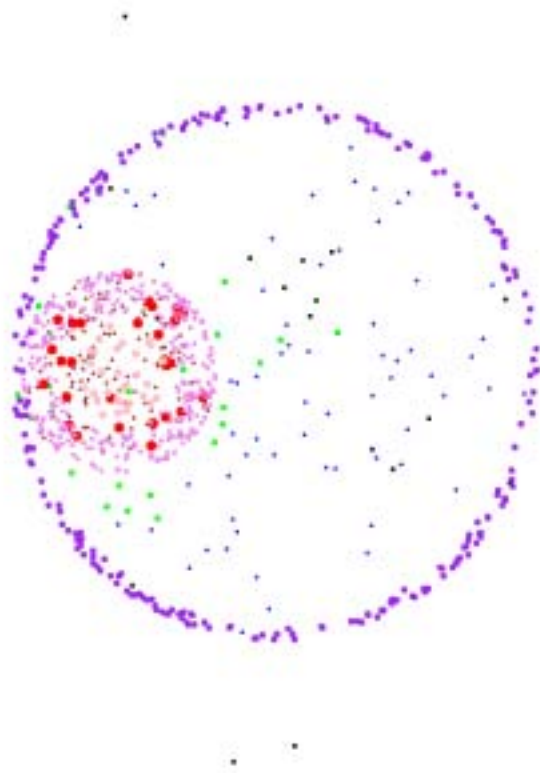


Figure 8: *Final snapshot of ChemCell simulation. HCO_3^- has diffused into the carboxysome where it has been converted into CO_2 (brown). A series of reactions mediated by the RuBisCO enzyme has converted the inorganic CO_2 to organic glucose (green).*

5 Future Plans

The work discussed in this report indicates it is possible to model the biochemistry of prokaryotic cells with a particle-based model that tracks the diffusive motion of individual molecular reactants and products within a simplified compartment model of the cell geometry. However, the initial version of ChemCell discussed here is still rudimentary in several respects. We view our work to date as a proof-of-concept implementation that illustrates the promise such a simulation holds for modeling cellular processes.

These are some of the new features we are currently adding to ChemCell to make it a more realistic cellular model:

(1) Geometry enhancements: The current version only recognizes spheres and spherical membranes. Ellipsoidal and capped cylinder compartments are being added. Sub-structures (patches) within membranes are also being considered to enable modeling of more realistic membrane properties.

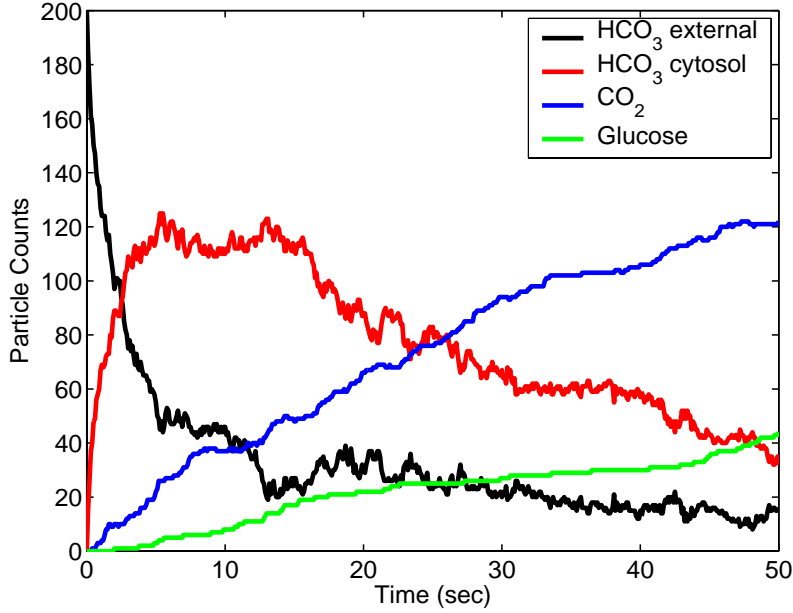


Figure 9: *Counts of bicarbonate ions, carbon dioxide, and glucose molecules during the ChemCell simulation.*

(2) Reactions: The reaction mechanism described in Section 2 is a simple formulation that reproduces well-mixed reaction statistics. A more realistic formulation is to couple the reaction probability to the probability of “overlap” between two diffusing particles. Consider one particle at the origin and a second a distance R away. If both particles move via Brownian motion with diffusion coefficients D_1 and D_2 for a time T , then the fraction of time F they spend overlapping each other (within a distance L_0) is given by the integral

$$F(T, R, D_1, D_2, L_0) = \frac{1}{4\pi} \int_0^T \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{(D_1 t)^{3/2}} \frac{1}{(D_2 t)^{3/2}} \exp\left(\frac{-r_1^2}{4D_1 t}\right) \exp\left(\frac{-(r_2 - R)^2}{4D_2 t}\right) \exp\left(\frac{-(r_2 - r_1)^2}{L_0^2}\right) dr_1 dr_2 dt$$

L_0 represents the concept of reaction volume; i.e. if the 2 particles are within a distance L_0 of each other, then they are close enough to react. Physically, it might be related to the size of a globular protein or protein complex. Our plan is to pre-tabulate solutions to this integral for all pairs of reacting species as a function of R since D_1 , D_2 , and L_0 are constants and T is the simulation timestep. For a pair of nearby particles, a reaction probability can then be computed using the pre-tabulated values and the Monte Carlo strategies described in Section 2.

(3) Parallel implementation: The current version of ChemCell is serial, but parallel speed-ups will be needed as models are run with larger particle counts and more realistic reaction networks. The diffusion

and reaction stages of each timestep are naturally parallel since particles move independently and reactant pairs can also be treated independently. Since diffusion and reactions occur with spatial locality, the natural partitioning of the model for parallel is a spatial decomposition. This is complicated by the fact that cell geometries and particle densities can be somewhat non-uniform. We plan to use a dynamic load-balancer that partitions based on geometry metrics to enable effective spatial partitioning. The Zoltan library [6] has several tools for both partitioning and particle exchange between partitions that will be useful in this effort.

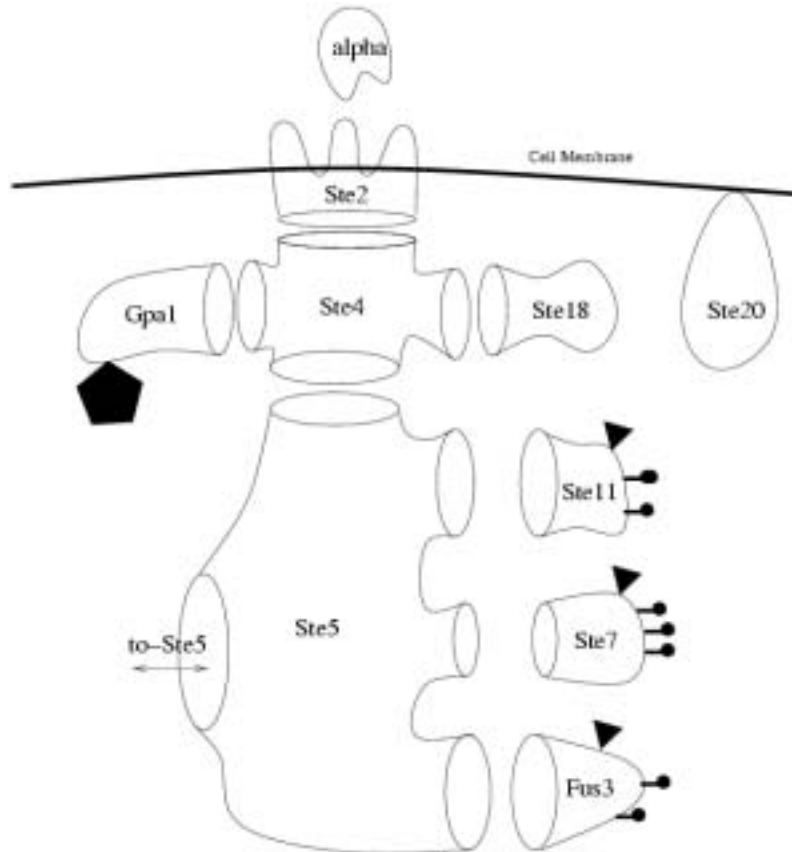


Figure 10: *Schematic of a portion of the reproductive alpha pathway in yeast. A signal is triggered by the extra-cellular alpha factor binding to a Ste2 membrane protein. Ste2 in turn can bind to Ste4 which binds to a scaffolding protein Ste5. Various other proteins can bind to the complex, some of which can be in one of several phosphorylation states indicated by the small dark circles. GDP and GTP also bind to the Gpa1 protein as indicated by the dark pentagon. A complex series of events results in the Fus3 protein being released to transmit the signal into the cell interior. This figure is due to Larry Lok at TMSI.*

(4) Species and reaction network generation: Complex reaction networks such as that of Figure 10 can involve dozens of interacting proteins. Each partially formed complex (which proteins are bound, what

states they are in) is a unique species in the ChemCell nomenclature. For the portion of the alpha pathway illustrated there could clearly be hundreds or thousands of intermediate species and related reactions (the combinatorial explosion described in Section 1). In the general case, the full set of reactions and reaction products for all intermediate species is not known *a priori*. It would thus be difficult in ChemCell to specify all species and reactions as inputs. We are working on this problem with collaborators at The Molecular Sciences Institute (TMSI) and their Molecuizer code. As described in Section 1, Molecuizer has the ability to generate new species and new reactions on-the-fly as a stochastic simulation progresses, based on user identification of molecular binding sites and rules for how large protein complexes are built from simpler components. We are working to automate the use of Molecuizer output to be used as ChemCell input; a longer-term goal is to merge the functionality of the two codes.

(5) Output: We are developing a richer set of output options within ChemCell for simulations with large numbers of particles, species, and reactions. A related issue is the need for better post-processing 3d visualization of cell geometries and particle motion. We are investigating several possible approaches to this problem, including the DReAMM visualization package distributed by the MCell group [9].

6 Acknowledgements

We gratefully acknowledge discussions with Larry Lok and Roger Brent at The Molecular Sciences Institute (TMSI) that have helped us formulate key features of this cellular model. Larry’s stochastic Molecuizer code has also been useful to us in generating reaction networks for input to ChemCell and well-mixed outputs to compare against. We thank Louis Romero at Sandia for assistance in developing the integral equation for reactions and its analytic solutions as discussed in Section 5.

FY03 funding for this project came from two sources: the LDRD program at the DOE’s Sandia National Laboratories, and the DOE’s Genomes-to-Life program (www.doegenomestolife.org) under the project “Carbon Sequestration in *Synechococcus* Sp.: From Molecular Machines to Hierarchical Modeling,” whose URL is www.genomes2life.org.

References

- [1] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.
- [2] L. J. D. Frink, S. L. Rempe, S. A. Means, M. J. Stevens, P. S. Crozier, M. G. Martin, M. P. Sears, and H. P. Hjalmarson. Predicting function of biological macromolecules. Technical Report SAND2002-3743, Sandia National Laboratories, Albuquerque, NM, 2002.
- [3] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem.*, 104:1876–1889, 2000.
- [4] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22:403–434, 1976.

- [5] D. S. Goodsell. *The Machinery of Life*. Springer-Verlag, New York, 1993.
- [6] <http://www.cs.sandia.gov/kddevin/Zoltan.html/Zoltan.html>.
- [7] <http://www.doegenomestolife.org>.
- [8] <http://www.mcell.cnl.salk.edu>.
- [9] <http://www.mcell.psc.edu/DReAMM>.
- [10] <http://www.nrcam.uchc.edu>.
- [11] C. J. Morton-Firth and D. Bray. Predicting temporal fluctuations in an intracellular signalling pathway. *J. Theor. Biol.*, 192:117–128, 1998. See also <http://info.anat.cam.ac.uk/groups/comp-cell/StochSim.html>.
- [12] T. S. Shimizu, N. Le Novere, M. D. Levin, A. J. Beavil, B. J. Sutton, and D. Bray. Molecular model of a lattice of signalling proteins involved in bacterial chemotaxis. *Nat. Cell. Biol.*, 2:792–796, 2000.
- [13] J. R. Stiles and T. M. Bartol. Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. In E. De Schutter, editor, *Computational Neuroscience: Realistic modeling for experimentalists*, volume 35, pages 87–127. CRC Press, 2001.

Distribution

5	MS 0310	Steve Plimpton, 9212
5	0310	Alex Slepoy, 9235
1	0316	John Aidun, 9235
1	0310	Alex Backer, 9212
1	0885	Grant Heffelfinger, 1802
1	0316	Elebeoba May, 9212
1	0310	Shawn Means, 9212
1	0310	Danny Rintoul, 9212
1	1110	Louis Romero, 9214
1	0316	Richard Schiek, 9233
1	1111	John Shadid, 9233
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616

Damian Gessler
National Center for Genomics Research
2935 Rodeo Park Dr E
Santa Fe, NM 87085

Larry Lok
The Molecular Sciences Institute
2168 Shattuck Ave
Berkeley, CA 94704

Roger Brent
The Molecular Sciences Institute
2168 Shattuck Ave
Berkeley, CA 94704