

# Cell++ Tutorial

## **Introduction**

This guide provides a brief tutorial on how to create your own simulations. The system as it is provided is configured to simulate metabolic pathways, however the program is capable of simulating signaling pathways as well as other small molecule systems such as calcium signaling.

In this tutorial, directions are provided to introduce you to making changes to Cell++. First, a small change to the metabolic pathway simulation will be described. After that, that are instructions on converting the program to simulate an intracellular signaling pathway.

## **Metabolic Pathways**

In a metabolic pathway, many different enzymes catalyze all the necessary reactions to take substrate molecules through chemical intermediates to their final products. In glycolysis, for example, at least a dozen enzymes work to convert glucose into a variety of useful metabolites: glycogen, glycerol, pyruvate, ATP and NADH, among others. Researchers have used metabolic pathway simulations to model disease states caused by enzyme mutations, the presence of unusual concentrations of metabolites and other perturbations. Generally, results from a metabolic pathway simulation take the form of a chart following each metabolite's concentration through the duration of the simulation.

Defining such a simulation requires these specifications:

- 1) The enzymes
  1. What the different species are and how they behave in terms of the reactions they catalyze and their mobility inside the cell.
  2. How many of each species is present at the start of the simulation, and where they are.
- 2) The metabolites
  1. What the different species are and how free are they to diffuse in each part of the cell.
  2. What the initial concentrations are for each species at each location within the cell.
- 3) The exact configuration of the space within the cell, including how large it is and at what resolution it will be computed.
- 4) How much time will pass during the simulation, and at what temporal resolution

it will be computed.

### **Current System Definitions**

Cell++ is configured for a simple glycolysis pathway simulation by default. It has these four enzymes, listed in biochemical pathway order:

phosphoglycerate kinase  
phosphoglycerate mutase  
enolase  
pyruvate kinase

Each of them is assigned appropriate kinetic parameters. There are also five metabolites defined:

2,3-bisphosphoglycerate  
3-phosphoglycerate  
2-phosphoglycerate  
phosphoenolpyruvate  
pyruvate

The cell dimensions are set to a cube 500 nm on each side, modeled with a 10x10x10 lattice, each element of which have sides 50 nm wide. This results in a total simulation volume of  $1.25 \times 10^{-16}$  L. The simulations are set to run for 50ms, with a temporal resolution of 5 $\mu$ s. Both the time and space resolutions are set in Main.cpp as TIMESCALE and SPACESCALE respectively.

### **Compiling and Running Cell++**

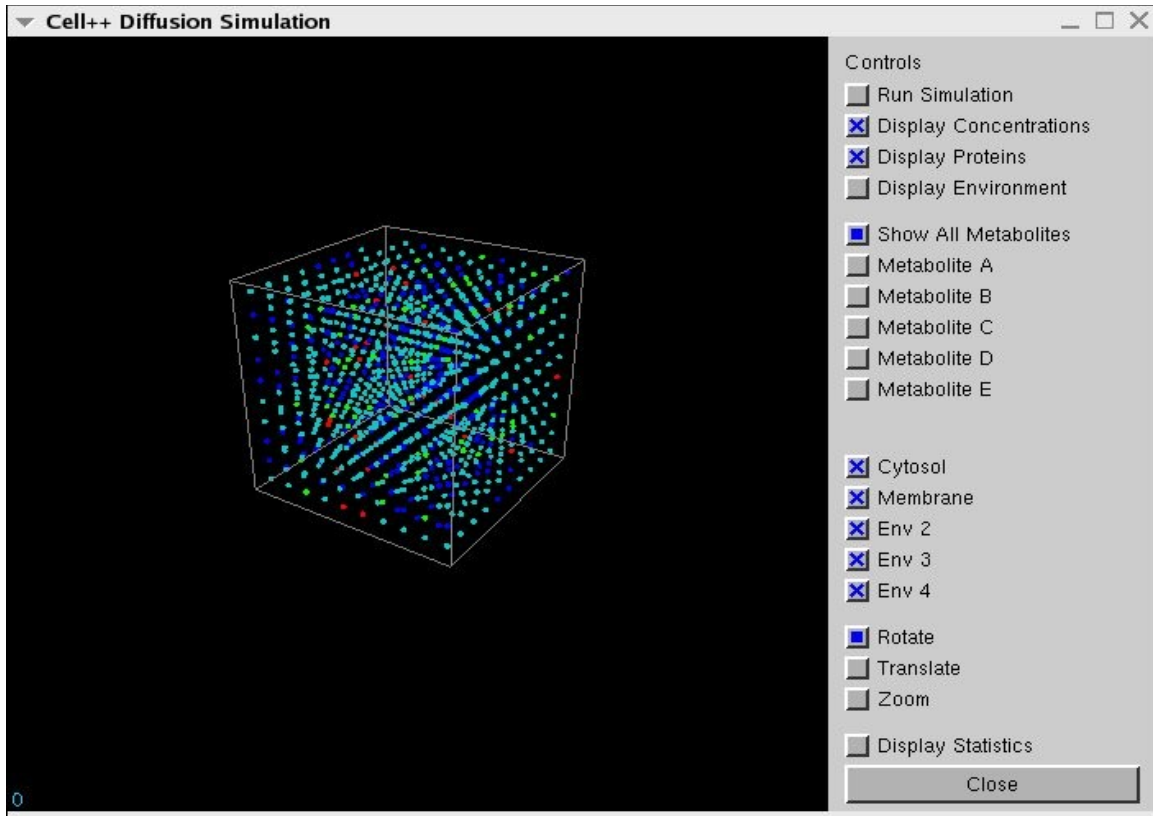
In the `met` directory, run

```
make
```

to compile Cell++. If all goes well, the program should be created, and entering

```
./simulation -g
```

at the command line should launch the simulator in graphical mode. Expect to see a screen like this:



This shows a view of the entire cell simulation, with the borders of the cell drawn in white wireframe, and the locations of the enzymes indicated with coloured spheres. The number in the bottom-left corner of the display shows the current timestep. The panel on the controls the display settings.

Press the button labeled “Run Simulation” to start processing. You will see colour gradients representing metabolite concentrations spread throughout the cell.

Once the simulation has completed, the output will be printed to standard output as a tab-delimited table, suitable for importing into spreadsheet programs. There are five columns, one for each metabolite, and 1000 rows with data sampled periodically throughout the simulation.

### **Adding a Fifth Enzyme**

Next we will extend the pathway being simulated by adding another metabolite and enzyme from glycolysis. Before any of the steps in the current simulation can occur in a real cell, glyceraldehyde 3-phosphate is converted into 2,3-bisphosphoglycerate through the action of glyceraldehyde 3-phosphate dehydrogenase.

The first step is to add the new metabolite into Cell++. This is done by modifying the file *definition.h* and adding META\_F to the list of metabolites at line 49. The enumeration should look like this after you change it:

```
// Metabolites
enum {
    META_A = 0,
    META_B,
    META_C,
    META_D,
    META_E,
    META_F,
    NUM_META
};
```

A fifth enzyme, ENZYME\_E, is already defined, but we need to modify it to perform the right chemical reaction. Open *Enzyme.cpp* and change the appropriate reactant at line 132 to META\_F (glyceraldehyde 3-phosphate) and the product at line 133 to META\_A (2,3-bisphosphoglycerate). The enzyme parameters should now look like this:

```
case MOL_ENZYME_E:
    mobility = 1.0;
    reactant = META_F;
    product = META_A;
    reactionRate = 0.2;
```

Next, we need to specify the enzyme kinetics. Add these lines into *input/params.in*:

```
FORWARD_REACTION_RATE_E=0210
REVERSE_REACTION_RATE_E=0010
EQUILIBRIUM_CONSTANT_E=0006
MAXIMUM_RATE_E=16700
```

In the same file, create some molecules of that enzyme by changing this line:

```
NUM_MOL_ENZYME_E=1000
```

Since the simulation volume is  $1.25 \times 10^{-16}$  L (a 500x500x500nm cube), populating it with 1000 molecules of this enzyme will create a concentration of 13  $\mu$ M.

Finally, we need to tell the simulation to begin with an initial concentration of

glyceraldehyde 3-phosphate, rather than 2,3-bisphosphoglycerate. Modify LatticeEnvironment.cpp and change META\_A to META\_F at line 1617:

```
void LatticeEnvironment :: pumpInMetabolite(float
amount_to_pump_in){
    int id = arr[0][0][0]->getID();
    Tuple * row = stateMatrix->getRow(id);
    row->tuple[META_F] += amount_to_pump_in;
}
```

Now that the changes have been made, run

```
make
```

followed by

```
./simulation
```

and you will be rewarded with a six-column table of the metabolite concentrations.

### **Signaling Pathways**

A signaling pathway defined here consists of a set of discrete interacting components which are linked in such a way that a signal is transmitted through the network of interactions. In a simple example (e.g. a kinase cascade), you may have two distinct components - molecule A and molecule B. If molecule A is 'activated' (e.g. phosphorylated) it is in a state which allows it to interact with and 'activate' molecule B, thus transducing a signal between the two molecules. Cell++ is able to simulate such phenomena within its three dimensional simulation environment. When comparing between different simulations (i.e. alternative architecture of pathways; influence of compartments) it is important to note that many simulations must be undertaken for the same set of conditions to obtain statistics for the simulations. Probably the most important output variable is the *time of signal transduction* i.e. how long for a signal to reach from the outside of the cell to the inside. In addition, it is possible to obtain statistics along the lines of how many molecules of each type were activated at certain times in the simulation.

In order to undertake such simulations - several critical parameters must first be defined :

A) *The simulation environment*

The 3D simulation environment consists of a lattice of cubes representing different environments. The collection of cubes may be in the shape of a large cube, or sphere or whatever shape you want. The size (number of cubes) can also vary - typically we vary our simulations from a 10x10x10 lattice to as many as 100x100x100, beyond this simulations can start to get very slow. Each cube is then defined as a particular environment (e.g. outer membrane, inner membrane, cytosol, canal (see below), nucleus within which different components may take on different behaviors e.g. outer membrane receptors are constrained to move only within the outer membrane environment. Obviously using a larger lattice allows for a more detailed geographical description of the simulation environment.

### *B) The molecules*

Each molecule to be simulated must be predefined, along with a set of molecule specific parameters - these include what type of molecule they are (membrane versus cytosolic); their relative numbers; which compartments they can move in; how they can move (for example membrane components move in discrete units - this is a 'feature' to ensure that they can travel around the corners and edges of the membrane); their speed of diffusion; colour in the visualization and so forth. At the beginning of the simulation, the molecules are placed at random in their defined environment.

### *C) Molecule interactions*

The architecture of the signaling pathway must also be defined, this is simply a list of which molecule 'activates' which other molecule. The probability of activating a molecule is currently determined by their relative distance using the following formula :

$$p(\text{Molecule A activates Molecule B}) = (K^2 - x^2) / K^2$$

where x is the relative distance (in lattice units) between Molecule A and Molecule B components, K is a constant (defined in these simulations as 1.5 for all molecule interactions). Again this formula is tunable.

### **Converting to Signaling**

Cell++ is by default configured to perform a metabolic simulation, but it is not difficult to change it to a signaling pathway mode. All that needs to be done is to disable some of the metabolism-specific code, and enable the signaling-specific routines. Small molecule diffusion, for example, is not required in our signaling pathway experiment, so we remove some lines of code in Simulation.cpp that use it. Comment out lines 501 through

506, which add in the initial metabolite and invoke the small molecule diffusion routines:

```
// if (numCycles == 1)
// {
//     lEnv->pumpInMetabolite(1.0);
// }

// lEnv->diffuse();
```

And line 54 of LatticeEnvironment.cpp:

```
//     buildDiffusionPatterns();
```

Also comment out lines 541 and 542, which output the final table of metabolite concentrations:

```
//     Matrix * stats = lEnv->getStatisticsMatrix();
//     stats->display();
```

Then uncomment line 499, which will activate the code that processes activation events by signaling molecules:

```
    checkMoleculeInteractions();
```

Next we need to create an environment that is more appropriate for the signaling experiment. Increase the cell dimensions to 40x40x40 lattice units by changing input/params.in:

```
X_DIM=40
Y_DIM=40
Z_DIM=40
```

Add a cell membrane by adding this line near the top of LatticeEnvironment.cpp:

```
#define PLASMA_MEMBRANE
```

For our experiment, we will want to localize ENZYME\_A to the plasma membrane and ENZYME\_E to the nucleus. The other three large molecule species must be free to wander throughout the cell. This is done in the input/compatible\_env.in file. Each line of this file consists of two numbers: the first number specifies the molecular species, and the

second number defines which environment type it is allowed to enter. We need to make the file look like this:

```
1 0
2 0
3 0
```

```
0 1
1 1
2 1
3 1
```

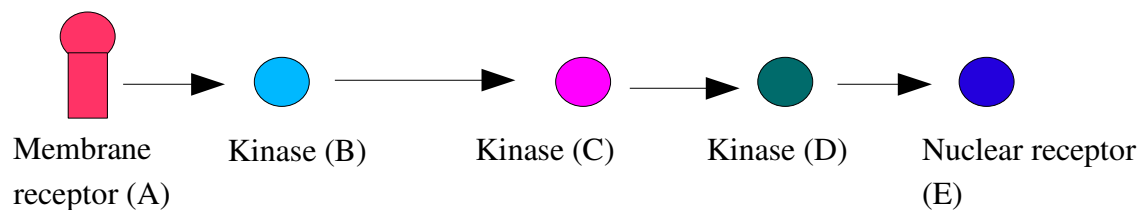
```
1 2
2 2
3 2
4 2
```

Which says that molecules numbered 1-3 are allowed in the cytosol (environment 0), 0-3 are allowed in the plasma membrane (environment 1) and 1-4 are allowed in the nucleus (environment 2.)

The metabolism simulation used thousands of molecules, but here we only want a few. In input/params.in, reduce the number of each signaling molecule to 20:

```
NUM_MOL_ENZYME_A=20
NUM_MOL_ENZYME_B=20
NUM_MOL_ENZYME_C=20
NUM_MOL_ENZYME_D=20
NUM_MOL_ENZYME_E=20
```

The architecture of the signaling pathway now in Cell+ is drawn below :



Initially only the membrane receptors are to be initialized as 'active'. Other components

get activated by coming into contact with other components in the pathway. For example, Kinase (C) can be activated by contacting either 'active' form of Kinase (B). Have Cell++ activate the membrane receptors by adding this code at line 139 of Simulation.cpp:

```
if (type == MOL_ENZYME_A)
{
    m->activate();
}
```

Finally, increase the timescale at line 53 of Main.cpp:

```
double TIMESCALE = 5.0e-4;
```

and the number of timesteps at line 524 of Simulation.cpp so that interesting behaviour will be observed during the simulation:

```
const static int MAX_CYCLES = 100000;
```

We can now compile and run the simulation:

```
make
./simulation -g
```

You'll see some output in the console window as the simulation progresses :

```
>      73285   Enzyme C      (5.90932, 37.6841, 35.0744)   -->
Enzyme D      (5.81385, 37.4892, 34.8525)
>      73902   Enzyme A      (8.3428, 4.21219, -0.419346)   -->
Enzyme B      (8.10534, 4.43373, -0.497688)
```

These two lines show that a molecule of type C has activated a molecule of type D after 73285 iterations of the simulation. After 73902 iterations, a molecule of type A has activated a molecule of type B. The 3D co-ordinates of each are given (within the defined 40x40x40 lattice cell).

The simulation can be run in non-graphical mode by omitting the '-g' option e.g.:

```
simulation
```

This greatly increases the speed of the simulation - important for collating statistics over several hundred simulations.

