

# Dynamic cellular automata: an alternative approach to cellular simulation

David S. Wishart\*, Robert Yang, David Arndt, Peter Tang and Joseph Cruz

Depts. of Computing Science and Biological Sciences, University of Alberta,  
 Edmonton, AB, Canada T6G 2E8

\* Corresponding author  
 Email: [david.wishart@ualberta.ca](mailto:david.wishart@ualberta.ca)  
 Phone: +1-780-492 0383

Edited by E. Wingender; received October 04, 2004; revised and accepted November 18, 2004; published December 01, 2004

## Abstract

A wide variety of approaches, ranging from Petri nets to systems of partial differential equations, have been used to model very specific aspects of cellular or biochemical functions. Here we describe how an agent-based or dynamic cellular automata (DCA) approach can be used as a very simple, yet very general method to model many different kinds of cellular or biochemical processes. Specifically, using simple pairwise interaction rules coupled with random object moves to simulate Brownian motion, we show how the DCA approach can be used to easily and accurately model diffusion, viscous drag, enzyme rate processes, metabolism (the Krebs's cycle), and complex genetic circuits (the repressilator). We also demonstrate how DCA approaches are able to accurately capture the stochasticity of many biological processes. The success and simplicity of this technique suggests that many other physical properties and significantly more complicated aspects of cellular behavior could be modeled using DCA methods. An easy-to-use, graphically-based computer program, called SimCell, was developed to perform the DCA simulations described here. It is available at <http://wishart.biology.ualberta.ca/SimCell/>.

**Keywords:** cellular simulation, cellular automata, *E. coli*, genetic circuit

## Introduction

Generally speaking, cellular systems and subsystems can be simulated at three different scales: the nanoscale ( $10^{-10}$ ), the mesoscale ( $10^{-8}$ ) and the continuum or macroscale ( $10^{-3}$ ). At the level of atoms and molecules (nanoscale) we typically use molecular dynamics (MD) or Brownian dynamics [Qian, 2002] to model the behavior of 100's to 1000's of discrete atoms over relatively short periods of time ( $10^{-10}$  to  $10^{-9}$  s) and space ( $10^{-9}$  m). MD methods, which treat atoms and bonds in a semi-classical manner, are fully deterministic and remarkably accurate over the short temporal and spatial scales that are normally simulated [Warshel, 2002; Karplus and McCammon, 2002]. Because of their accuracy, MD techniques can be used to simulate state or conformational changes, predict binding affinities, investigate single molecule trajectories and model stochastic or diffusive interactions between small

### TABLE of CONTENTS:

Title

Abstract

Introduction

Methods

Results

Discussion

Conclusion

References

Figures

1 2 3 4 5 6

Tables

1 2

numbers (<5) of macromolecules [Karplus and McCammon, 2002]. However, even with today's computers it is still impossible to model large numbers of molecules ( $>10^6$ ) or macromolecules ( $>10^2$ ) over extended periods of times and space (milliseconds to millimetres). Under these extreme conditions one is compelled to model molecular events using continuum approaches where molecules essentially lose their discreteness and become infinitely small and infinitely numerous.

Therefore, to model at the macroscale or continuum level, we must turn to ordinary (ODE's) or partial differential equations (PDE's) to describe our systems of interest. For instance, time dependent ODE's are routinely used to describe individual enzyme reactions, diffusion events and other rate-dependent processes [Duggleby, 1995]. Alternately, systems of partial differential equations (SOE's) can be used to model much more spatially or temporally complex, interdependent systems such as metabolism or cell signaling [Endy and Brent, 2001]. Indeed, the new computer-driven SOE approaches lie at the heart of several notable cellular simulation efforts including VCell [Slepchenko *et al.*, 2003], E-cell [Ishii *et al.*, 2004], *Drosophila* embryogenesis [Meir *et al.*, 2002], models of T7 bacteriophage development [Endy *et al.*, 2000], yeast cell division [Sveiczzer *et al.*, 2004] and a wide variety of metabolic simulators [<http://sbml.org/index.psp>].

However, not all sets of differential equations are solvable nor are all systems amenable to descriptions by differential equations. For instance, state changes, discontinuities, irregular geometries or discreteness (low copy numbers) are not easily described by differential equations. Furthermore, due to their continuum nature, the solutions to differential equations always generate smooth curves or surfaces that fail to capture the true granularity or stochasticity of living systems. An additional challenge to using differential equations is that they require a strong foundation in mathematics, a detailed knowledge of many (sometimes unknowable) rate constants or initial conditions, and a powerful "solver" to generate solutions.

So how do we connect the atomic realism of nanoscale simulations with the predictive power of continuum modeling? The answer, we suggest, lies in mesoscale modeling. At the mesoscale ( $10^{-8}$  –  $10^{-7}$  m) macromolecules can still be treated as discrete objects occupying a defined space or volume. In fact, this "atomism" actually allows mesoscale models to display the stochasticity or granularity found in real biological systems. Additionally, because Brownian motion dominates over all other forces at this scale, it turns out that significant dynamic simplifications are possible with mesoscale modeling. These simplifications potentially allow very long time scales and large numbers of entities or reactions to be modeled (*vide infra*).

Here we wish to show how mesoscale simulation can be performed using a concept we are calling dynamic cellular automata (DCA) or agent-based modeling [Reeves, 1983; Reynolds, 1987] (<http://www.red3d.com/cwr/boids/>). Specifically using simple pairwise interaction rules and simulated Brownian motion on a 2D grid we show how DCA can be used to easily and accurately model a variety of spatial and temporal phenomena including: macromolecular diffusion, viscous drag, enzyme rate processes, the Krebs's cycle, and complex genetic circuits. We also compare these models to experimentally measured data and demonstrate a remarkably good agreement between experiments and the DCA predictions. A key advantage to the DCA modeling processes is that it is very easy to implement and requires remarkably little skill in mathematics or any knowledge of how to formulate or solve time-dependent ODE's, PDE's or SOE's. Furthermore, the DCA approach appears to be much more scalable, visually more realistic and applicable to a much wider range of cellular phenomena than many other simulation methods. A more complete description of the method, along with a discussion of its strengths and limitations follows.

---

## Methods

## Background and motivation

Cellular automata (CA) are multi-object computer simulation tools that consist of large numbers of simple identical components with local interactions layered over a lattice or grid. The states or values of the components evolve synchronously in discrete time steps according to identical rules. The value of a particular site is determined by the previous values or states of a neighbourhood of sites around it. Originally conceived by von Neumann in the late 1940's (<http://www.brunel.ac.uk/depts/AI/alife/al-ca.htm>), cellular automata have been used to model a wide range of physical processes including heat flow, spin networks and reaction-diffusion processes [Wolfram, 2002]. Cellular automata also have a long history in biological modeling. Indeed, one of the first computer applications in biology was a CA simulation called Conway's Game of Life [Gardner, 1970]. This simple model simulated the birth, death and interaction between cells randomly scattered over a square lattice or grid. The fate of every cell was determined according to three pair-wise interaction rules. These interaction rules were typically "if-then-else" statements describing what a cell could do depending on the number of adjacent neighbors. From these simple rules some remarkably interesting behaviours or patterns would emerge. The success of the Game of Life in generating interesting patterns from disordered arrays led to the development of other CA models to describe far more complex and biologically realistic pattern formation processes [Ermentrout and Edelstein-Keshet, 1993; Wolfram, 2002]. However, over the past 10 years biological modeling using cellular automata has largely faded from use. Nowadays, perhaps the most widespread and familiar use of CA can be found in popular computer games such as SimCity (<http://simcity.ea.com/>), SimEarth and The Sims (<http://thesims.ea.com/us/index.html>). These games allow users to visualize, simulate and interact with synthetic planets, cities and households. Their spectacular color graphics and highly interactive features basically conceal a CA simulation engine that is not much different than the one first used in Conway's Game of Life.

Given that CA methods can be used to model cells, humans, households, cities, and planets, could they also be used to model cellular components? In many respects the behavior of proteins and other macromolecules inside a cell is not all that different from the interacting blocks in Conway's Game of Life. In particular, if we look at macromolecules at the mesoscale, they appear as discrete, but largely unstructured tubes, cubes or spheres. The objects (proteins, DNA) can still interact with each other, but their interactions are no longer constrained or described by shape complementarity or lock-and-key fitting as they would be if we were modeling at the nanoscale. Instead the interactions can be described by simple, logical operations or statements such as:

1. If A is adjacent to B, then A binds B.
2. While C is bound to D, produce F.
3. If A is adjacent to M, then M is converted to P

Therefore, at the mesoscale level, we would argue that molecular interactions need not be governed by electrostatic and Van der Waals forces, but rather they can be treated using phenomenological or logical descriptors of previously observed or known events.

The simplification attainable concerning object interactions at the mesoscale is also extensible to object motions. In particular, macromolecular motion at the mesoscale level is not dictated by Newton's equations, but rather by Brownian motion [McCammon, 1987; Ayton *et al.*, 2004]. The dominance of randomized, Brownian motion at this level means that object movements can be best described by randomized, discrete-state jumps. The fact that macromolecules can be simplified in shape (spheres or squares), their interactions simplified in terms of rapidly calculable logic statements and their motions described by adjacent grid jumps means that many mesoscale events can be modeled using a technique we are calling dynamic cellular automata (DCA). DCA differs from conventional CA in that the DCA model attempts to simulate real motions via Brownian dynamics. In other words, the motions

of particles are intended to mimic the motions observed in real macromolecules. Therefore, random objects cannot be taken up and randomly scattered over the grid with each iteration as they are in most CA models. Rather, DCA requires that regular time steps be taken (using a master clock), that objects can only move to adjacent squares with each time step, that object collision detection be included and that the grid size and time-steps be sufficiently small to be consistent with physical laws or experimentally measured parameters (i. e. diffusion rates).

### Program description

In building our DCA cell simulator, called SimCell, we tried to create a very general structure that would allow the accurate simulation of most cellular phenomena (transport, enzyme kinetics, metabolism, reaction-diffusion, signaling, genetic circuits and transcription/translation). This required selecting a set of molecular components and choosing a collection of interaction rules that would be sufficiently diverse to describe the events that typically go on both inside and outside a cell. Based on these considerations we identified 4 kinds of component molecules: 1) small molecules (metabolites, ligands), 2) membrane proteins (which can only exist in membranes), 3) soluble protein/RNA molecules and, 4) DNA molecules (which are non-mobile). Because cells are composed of subcellular organelles and because some simulations require compartmentalization of reactants, products or metabolites, we also introduced a fifth type of molecule or super-molecule – the membrane. In our model, membranes are non-mobile entities that describe boxes or borders that may be permeable or impermeable to certain molecules. These membrane super-molecules may be used to define cytoplasmic compartments, periplasmic compartments, nuclear compartments or other subcellular organelles.

Molecular interactions within cells can also be described in a relatively simple, straightforward manner. Specifically, there are 3 major types of pairwise interactions. Any two molecules or molecular entities may bind and stick (B&S), touch and go (T&G), or be directionally transported (TR). Molecules that bind and stick have association and dissociation constants or on/off probabilities and rates. Many protein receptors, DNA binding proteins, catabolic repressors and gene activators will bind and stick to a target molecule. Under the SimCell rules, molecules that bind and stick with each other must necessarily transform themselves into a new entity (a third molecule) or, in the case of DNA binding, lead to changes in the creation or transcription rate of a third molecule. In contrast to bind and stick interactions, non-interacting molecules will typically touch and go (bump), always leaving each other in tact. However, some touch and go operations do lead to a molecular transformation, particularly when we consider enzymes such as metabolic enzymes, nucleases or proteases. While enzymatic catalysis technically involves some binding, given the time step used in our mesoscale simulations (1 ms), catalysis is, for all intents and purposes, instantaneous. In these enzymatic touch and go operations, two objects will meet leaving one object in tact and one transformed. Finally, in transport operations an object (usually a small molecule) is taken and almost instantaneously (< 1 microsecond) moved across a barrier (usually a membrane). This is a specialized case of touch and go, but with the requirement that the movement of the substrate is directional – not random.

These three interaction scenarios and their corresponding interaction consequences can be summarized as follows. Bind and stick (B&S) interactions lead to either the generation of a new molecule type, with new properties (0), the enhanced synthesis of one or more protein/RNA molecules (1), or the repressed synthesis of one or more protein/RNA molecules (2). Touch and go (T&G) interactions lead to either no change (0), elimination of a target molecule (1) or transformation of a target molecule to one having new properties (2). Transport (TR) leads to unidirectional movement across a barrier (0), bidirectional movement across a barrier (1) or unidirectional movement across a barrier with a concomitant transformation of the target (2). The allowed interactions and state-change consequences for each of the 5 different molecular types are catalogued in the interaction matrix shown below ([Table 1](#)). This is typical of the kind of interaction matrix that might be generated for a

given model prior to running a DCA simulation.

**Table 1:** Interaction matrix for the 5 molecular components (Small Molecule, Membrane Proteins, Soluble Protein, DNA and Membrane) in SimCell.

	Small Molecule	Membrane Protein	Soluble Protein	DNA Molecule	Membrane
Small Molecule	T&G  0 – –	T&G  0 1 2  B&S  0 – –  TRA  0 1 2	T&G  0 1 2  B&S  0 – –	T&G  0 – –	T&G  0 1 –  TRA  0 1 2
Membrane Protein	T&G  0 1 2  B&S  0 – –  TRA  0 1 2	T&G  0 – 2  B&S  0 – –	T&G  0 1 2  B&S  0 – –  TRA  0 1 2	N/A	N/A
Soluble Protein	T&G  0 – 2  B&S  0 – –	T&G  0 1 2  B&S  0 – –  TRA  0 1 2	T&G  0 1 2  B&S  0 – –	T&G  0 – –  B&S  0 1 2	T&G  0 1 2  B&S  0 – –  TRA  0 1 2
DNA Molecule	T&G  0 – –	N/A	T&G  0 – –  B&S  0 1 2	N/A	N/A
Membrane	T&G  0 1 –  TRA  0 1 2	N/A	T&G  0 1 2  B&S  0 – –  TRA  0 1 2	N/A	N/A

T&G means touch and go, B&S means bind and stick, TRA means transport and N/A means not applicable.

Allowed states or interaction operators (0, 1 or 2) are designated beside each interaction..

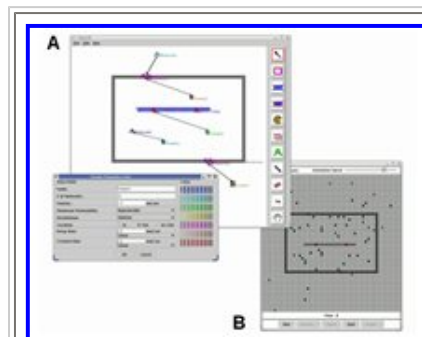
In SimCell the motion for all objects (except membranes and DNA molecules) is performed in a step-wise fashion over the space defined by a regular 2-dimensional grid. Within the grid each object or molecule occupies a square, typically measuring 3 nm on a side. The choice of 3 nm as the standard grid increment is based on a number of criteria. For instance, the diameter for an average protein is approximately 3 nm, the average width of a cellular membrane is approximately 3 nm, the average width of a supercoiled DNA molecule is about 3 nm and the average velocity of a macromolecule in a cell is approximately 3 nm per millisecond [Sundararaj *et al.*, 2003] ([http://redpoll.pharmacy.ualberta.ca/CCDB/cgi-bin/STAT\\_NEW.cgi](http://redpoll.pharmacy.ualberta.ca/CCDB/cgi-bin/STAT_NEW.cgi)). This macromolecular diffusion rate also helps to define a time step that is appropriate for these mesoscale simulations – namely a millisecond. Macromolecules such as proteins or RNA molecules are allowed to occupy only a single square at a time, whereas up to 100 small molecules (which typically measure 0.3 nm across) are permitted to occupy a single square. This multiple occupancy rule for small molecules avoids the need to reduce the grid size and time steps by a factor of 10. This simplification also makes the simulations run much faster. Small molecules are known to diffuse at approximately 50 nm/ms [Jacobson and Wojcieszyn, 1984], or approximately 10 times faster than macromolecules. To accommodate their faster diffusion rate, SimCell performs 10 small molecule movements/interaction checks for each macromolecular movement step. To provide additional flexibility, SimCell users are free to define the grid size and geometry, depending on the type, number and complexity of the reactions, models or components. The default geometry and grid size is a square composed of 100 x 100 elements.

For mobile molecules such as protein/RNA molecules, small molecules and membrane proteins, the initial spatial location (uniformly distributed, North, East, West, South bias, outside, inside cell or

nucleus), number of molecules, molecule colour (for viewing), molecule properties (permeability with respect to the membrane, velocity, complexation state, decay rate, creation rate, etc.) and molecule label are all defined by the user during the initial set-up phase. These properties, which are entered into pop-up property cards, may be changed between simulation runs. It is important to note that mobile molecules also have an optional "creation" rate (linear or exponential) that can be entered into their property card (the default value is always 0). The creation option may be used to describe a general or external source for nutrients, metabolites or macromolecules – without the need to fully map out superfluous metabolic or transcriptional pathways. The molecular decay rate (linear or exponential) may be used in a similar manner to describe external or internal sinks needed to eliminate individual or monomeric molecules. However, the decay rate can also be used to model the disassociation of complexes composed of two or more molecules.

Static or non-mobile entities in SimCell are treated slightly differently than mobile molecules. For genes, operons or gene/promoter elements residing on the cell's DNA, users must designate the location of these components (via dragging and dropping a coloured gene icon onto the DNA strand) once the DNA molecule is drawn. Properties for the specific genes may also be entered, including the transcription rate, the number of genes, the gene order and the type of protein/RNA molecules that may be synthesized from the selected gene or operon. Protein synthesis, in the current version of SimCell, skips the steps of RNA production, processing and ribosomal binding. This abstraction is done to both simplify the model building process and accelerate the simulation run.

To facilitate interactive model design SimCell uses its own graphical modelling tool, written in Java. This model rendering tool allows a schematic description (using single representative molecules or entities) of the model components to be drawn on a simple canvas using a set of drag-and-drop palette tools (see [Figure 1a](#)). As each molecular object is moved onto the canvas, the user is prompted with a pop-up window or dialog box to fill in specific information about that molecule or entity. The molecular objects and their property cards can be edited or deleted from the canvas at any time during the model construction process. Additionally, the SimCell GUI permits a variety of interaction arrows or connectors to be drawn between molecular entities. These interaction arrows are used to describe the allowed pairwise interactions and their interaction consequences. Through SimCell's model rendering tool, new models may be generated or old models may be uploaded, edited and saved. The program also allows general notes or comments to be added about each saved model. Prior to generating a model through SimCell's model rendering tool, it is suggested that users compile a list of all enzymes, proteins, ligands, reactants and products needed to describe the model. A common error among first-time SimCell users is the tendency to neglect to include products or resultant complexes in their schematic diagrams. Once the model's schematic wiring diagram is completed and the necessary fields filled in, the simulation is ready to be run.



**Figure 1:** A) Screen capture of the SimCell model rendering graphical user interface (GUI) illustrating a cell containing a chromosome with 2 genes, 3 types of soluble proteins, 2 types of membrane proteins and 2 types of small molecules. Interaction arrows describe which molecules interact with each other. A pop-up property window is also shown. B) Screen shot of the simulation window for SimCell showing various moving molecular objects placed on the grid. This window is launched when users select "Run" from the rendering window, shown in A).

To initiate the run phase, SimCell maps the schematic pathway or wiring diagram onto its simulation grid. This involves generating the 2D grid, defining the compartments, placing the requisite number of molecules over the grid and assigning initial velocities/directions to each molecule. Error or consistency checks are performed during this mapping phase to ensure that all objects have a place to go, that key information is not missing and that the interactions and the interaction grid are logically consistent. Once the model mapping is complete, a simulation viewer (Figure 1b) is launched in a separate window.

SimCell's simulation viewer is essentially a computer movie generator. It simply displays the 2D mesoscale dynamics of the model as calculated using SimCell's DCA algorithm. For the most part a DCA simulation looks like a series of multicolored objects moving randomly about, with some disappearing and others appearing. The simulation viewer can also display multi-color line graphs of molecular species as their quantities change over time. Information on quantities, spatial positions, state and time can also be saved as a flat file for more detailed analysis at a later time. The simulation viewer, also allows a simulation to be stopped, edited or altered and rerun at any given time. For very large and complex simulations, the graphical rendering option may be turned off to accelerate the overall computational speed. Because each model generated by SimCell's model rendering tool has a unique random number associated with it, every time the model is run the same "movie" will be regenerated. In other words, all simulations can be replayed at any time, on any computer without any loss of information. The use of a random number seed avoids the need to store great quantities of positional and temporal data for every simulation.

The dynamics seen in SimCell's simulation viewer are generated through global movement cycles calculated by a DCA algorithm. Each movement cycle involves up to 8 steps or checks that are performed on every molecule in the system. These include: a creation/decay state check, the assignment of random direction vector, a mobility check, a boundary check, a collision check, a movement step, a neighborhood check, and finally a neighbor interaction step. During the creation/decay check, each molecule or class of molecules is queried to see if it has been assigned a non-zero creation or decay rate. Those molecules having non-zero rates are then either added, disassociated or removed in a stochastic manner (+/- 10% error) relative to their assigned rates. After the creation/decay calculations have been performed, individual molecular motion vectors are then assigned. Specifically, each mobile molecule (protein/RNA, membrane protein, small molecule) in the lattice is given a random number between 1 and 9. Numbers 1-8 designate movement to adjacent squares (N, S, E, W, NE, SE, NW and SW), while 9 means that the molecule remains in that square. After the direction vectors have been assigned, a mobility check is run to deal with any membrane proteins in the system. The mobility checker determines whether the assigned movement vector will still keep the protein inside the membrane (note that membrane proteins move at 0.03 nm/ms, not 3 nm/ms as with soluble proteins). If not, the motion is disallowed and the protein stays put. After the mobility check is run, a boundary check is performed. In the boundary check, molecules are tested to see if they lie at the outer boundaries of the simulation grid. SimCell uses periodic boundary conditions so that those molecules designated to move beyond a grid boundary will always appear back within the simulation frame, but on the opposite side of the grid. Any molecules designated to move beyond a grid boundary either stay in their current position or choose another direction vector, according to the user's specified setup parameters.

After the boundary check is completed, a collision detection step is performed on all mobile molecules. The collision detector checks to see if the given molecule's randomly chosen direction vector will send it to a square currently occupied by another molecule or another non-mobile entity such as a membrane or a DNA molecule or a square which no more small molecules can occupy. If the

properties assigned to that molecule allows it to co-occupy the chosen space the movement is allowed. If the properties assigned to that molecule do not allow it to co-occupy that space, then the move is disallowed and the molecule remains stationary. After collision detection has been done on all relevant molecules, the movement step is finally performed. In the movement step, all molecules that can be moved are moved according to their assigned velocities, movement rules and directions.

Once all the molecules on the grid have been moved, a neighborhood checking algorithm is run. During the neighborhood checking step each molecule is queried to see if it is in contact with another molecule or entity with which it may interact. Using the model's interaction matrix and the assigned interaction rules, interaction operations are performed on that central molecule. If interactions have defined on/off rates or probabilities, random numbers are generated to determine if the interaction (binding, catalysis) is allowed to take place. If no rates or probabilities are provided, the program assumes these interactions occur with 100% probability. It is important to note that many interaction steps require relatively different molecular movements or grid manipulations. For instance, if two molecules are allowed to bind to each other to create a third molecule, the exterior (small) molecule is merged into the central (protein) molecule and the new central molecule is assigned the properties of the third molecule. If the two binding molecules are proteins, the same merging process is performed. In the case of degradation or proteolysis, the surrounding molecule or molecules are eliminated, while the central molecule remains unchanged. For enzymatic reactions, the surrounding molecules are transformed from substrate to products or *vice versa* – depending on the local substrate/product abundance and pre-assigned conversion probabilities or rate constants. For transport interactions, the adjacent molecule or molecules are moved "instantaneously" across the membrane to the nearest free square on the opposite side of the membrane. To model DNA binding or transcriptional activation/repression, the binding molecule remains stationary and attached to the DNA binding site for as long as the on/off rates require. When the bound protein is released from the DNA, it is moved to the nearest free, non-adjacent square.

It is only after all allowed interactions on all grid molecules have been performed that the next movement cycle in the simulation may be started. This iterative movement and state-updating process is repeated until the simulation is terminated – either interactively by the user or by specific limits placed on the program. When SimCell is run on a moderately fast desk-top computer (1 GHz processor) with 256 MB of memory, the typical update or movement cycle for an averaged sized model (300 molecular entities, 10,000 squares) takes approximately 0.1 CPU seconds. Since each cycle typically corresponds to 1 millisecond of "real" time inside the cell, this means that the DCA method allows simulations to be run with a fractional slow-down of just 100. Additional temporal or spatial abstractions can sometimes allow DCA simulations to be run in real time (no slow-down). Compared to standard molecular or Brownian dynamics calculations where fractional slow-downs of  $10^{13}$  are typical [Karplus and McCammon, 2002; Shen and Freed, 2002], the DCA approach clearly provides a very significant speed-up.

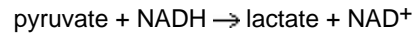
### Test simulations and validation

Obviously, the DCA approach is relatively new to cellular modeling. To validate its utility and performance we attempted to model 4 fundamentally different but important cellular phenomena for which experimental measurements were available or for which results are well characterized. These included: 1) the modeling of simple enzyme kinetics; 2) the modeling of macromolecular diffusion in different environments; 3) the modeling of metabolism and metabolic mutants in the Krebs's or TCA cycle; 4) the modeling of the repressilator oscillatory gene circuit [Elowitz and Leibler, 2000]. These models span the range of most kinds of modern cellular simulations, from simple chemical kinetics to more complex metabolism and genetic circuits. For a number of these models we also investigated the utility of DCA in monitoring or simulating the effects of finite numbers and stochasticity on certain

processes.

### Enzyme kinetics

To model enzyme kinetics we constructed a generic enzyme, product, substrate system modeled after lactate dehydrogenase. Lactate dehydrogenase (LDH) is a liver enzyme that catalyzes the following reaction:



To simplify the model, we ignored the NAD cofactor and assumed the reaction was essentially unimolecular, first order. Recently measured kinetic parameters for a reptilian LDH isozyme indicate a forward (pyruvate to lactate) reaction rate with a  $V_{max}$  of 0.56  $\mu\text{mol}/\text{min}/\text{ml}$  and a reverse  $V_{max}$  of 0.27  $\mu\text{mol}/\text{min}/\text{ml}$  at pH 7.0 [Javed *et al.*, 1997]. This means the difference between the forward and reverse reaction rates is approximately 2 fold. To model in this reaction in SimCell we constructed a 2D grid containing 10,000 squares (100 x 100) measuring 3 nm on a side and populated the grid with 1000 molecules of pyruvate and 1 molecule of lactate dehydrogenase. In this simulation, each grid square was assumed to contain 100 molecules of water (concentration = 55 M), meaning that the initial concentrations of pyruvate and lactate dehydrogenase were 55 mM and 55  $\mu\text{M}$  respectively. To investigate the effects of low substrate concentration, we also determined the enzyme kinetics with an initial concentration of pyruvate of 2.7 mM. To study the influence of reversibility, a second set of simulations was run in which the forward reaction rate was 100x greater than the reverse rate. All simulations were run over a period between 15,000 and 30,000 cycles. To match the  $V_{max}$  values experimentally measured for this enzyme it was necessary to assume that only 1/60 of the enzyme-substrate interactions led to pyruvate conversions. This is equivalent to setting the simulation time step to 1 second. The resulting kinetic or "enzyme progress" curves were also fit to the following idealized rate equation:

$$[\text{Lactate}] = [L]_0 (1 - e^{-kt})$$

where [Lactate] is the concentration of lactate in mM,  $[L]_0$  is the steady state lactate concentration in mM,  $t$  is time in seconds and  $k$  is the turnover rate.

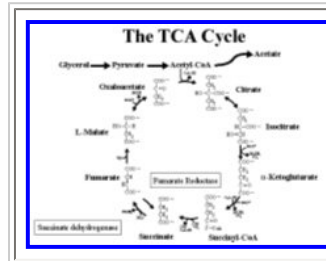
### Macromolecular diffusion

The second test or assessment of the DCA method involved modeling macromolecular diffusion inside a cell. In this test we wanted to investigate whether simple interaction, collision and movement rules could be developed to more accurately model the effects of heterogeneous binding and high macromolecular density within the cytoplasm of a cell. Previous experimental studies [Kao *et al.*, 1993; Jacobson and Wojcieszyn, 1984; Dayel *et al.*, 1999] have indicated that macromolecules move much slower (3-5 times) in the cytoplasm than would be predicted from pure diffusional considerations. The slowed diffusion of cytoplasmic molecules is thought to arise from three factors: collisions with obstacles, increased intrinsic viscosity of the medium (microviscosity due to small molecule solutes), and binding interactions or "sticking" to other macromolecules. To more thoroughly investigate these effects with SimCell, diffusion conditions were studied by tracking a single macromolecule moving in three different environments: 1) an aqueous, non-viscous solution (no other molecules present in the grid); 2) a solution where 25% of the volume was occupied by macromolecules (similar to the situation in a cell's cytoplasm) and 3) a solution where 50% of the volume was occupied by macromolecules (a density similar to the endoplasmic reticulum). We also explored 6 interaction or movement rules which were designed to progressively decrease the speed or rate of diffusion of the tracking molecule. These

included: a) limiting the tracking molecule to 8 movement directions instead of 9 (i. e. forcing it to always move from its last square) and allowing the tracking molecule to ignore collisions, viscosity and sticking; b) allowing 9 movement options but still allowing the tracking molecule to ignore collisions, viscosity and sticking; c) allowing 9 movement options, including collision detection but ignoring viscosity and sticking; d) allowing 9 movement options, including viscosity but ignoring collisions and sticking; e) allowing 9 movement options, including collision detection and sticking, but ignoring viscosity; and finally, f) allowing 9 movement options, including collision detection, sticking and viscosity. In developing a macromolecular "sticking" rule we assumed that macromolecules that were in contact with the tracking molecule would reduce its motion by causing transient binding or disruption of the water shell surrounding the tracking molecule via surface tension effects [Kao *et al.*, 1993]. To implement this macromolecular sticking rule, we counted the number of macromolecules adjacent to the tracking molecule at each movement cycle and reduced the probability of movement for the tracking molecule by 50% for each neighbor. To model the effects of small molecule viscosity, we simply reduced the probability of movement by a set factor (30%) consistent with microviscosity measurements previously reported [Dayel *et al.*, 1999; Kao *et al.*, 1993]. Collision detection was handled such that if the tracking molecule was selected to go to a square occupied by another macromolecule, it could not move until another movement cycle had passed that assigned the tracking molecule to a square to which it could move. All simulations were run on a simple 2 dimensional grid containing 100 x 100 squares and populated with above-mentioned concentrations of macromolecules molecules. Diffusion rates or molecular velocities were measured by counting the number of non-redundant squares that the tracking molecule sampled over 100 iterations or movement cycles (i. e. squares per cycle). The rate or velocity determined under scenario 1b (macromolecule in aqueous conditions, 9-move set) was calibrated to the known rate of 27 nm/ms [Jacobson and Wojcieszyn, 1984] ([http://redpoll.pharmacy.ualberta.ca/CCDB/cgi-bin/STAT\\_NEW.cgi](http://redpoll.pharmacy.ualberta.ca/CCDB/cgi-bin/STAT_NEW.cgi)) and all tracking molecule velocities measured for the other 17 scenarios were calibrated to this value.

### Metabolic simulation

To investigate the utility of the DCA method for undertaking more complex and potentially more interesting cellular simulation problems we turned to metabolic simulation – specifically the well-known and well-studied tricarboxylic acid (TCA) or Krebs cycle. In modeling the TCA cycle via SimCell we used the pathway described in Figure 2. Here glycerol was used as the primary carbon source, with acetate as the final waste product. All 11 substrates/products and 11 enzyme species were included in this pathway. In total, more than 20 interactions were simulated in the DCA model. To simplify the model it was assumed that the reaction rates for all enzymes were identical (with the exception of the reverse rate for fumarate reductase) and that all enzymes were water soluble (which is not the case). Likewise, the production of other cofactors (NADH, FADH<sub>2</sub>) and byproducts (GTP, CO<sub>2</sub>) was ignored. The TCA cycle for *E. coli* was simulated under three different conditions: 1) the intact or wild-type cycle; 2) a disrupted TCA cycle in which succinate dehydrogenase was removed (single knockout); and 3) the TCA cycle in which both succinate dehydrogenase and fumarate reductase are removed (a double knockout). In *E. coli* it is known that fumarate reductase is an isozyme for succinate dehydrogenase, but with much less activity under aerobic conditions [Dickie and Weiner, 1979]. Therefore, one would predict that the single knockout should function reasonably closely to the wild-type strain. In the SimCell simulations we used a grid of 100 x 100 squares with a simulated starting concentration of 100 mM glycerol and starting concentration of 1 mM of each of the TCA enzymes. Time steps for this simulation were set to intervals of 4 seconds. In order to compare the simulation with the experimental results, the growth rate for the *E. coli* strains (i. e. the TCA enzyme production rate) was assumed to be 5 times slower for the knockouts than for the wild-type strain. This was consistent with optical density measurements previously made for these strains.

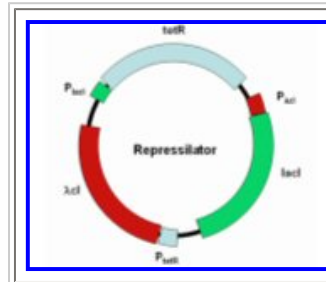


**Figure 2:** A schematic diagram of the TCA or Krebs cycle with glycerol as the primary carbon source.

To monitor the accuracy of these simulations, we compared the predicted consumption of glycerol and the predicted production of succinate to the results from a series of metabolomic experiments conducted on a set of appropriately mutated *E. coli* strains using NMR spectroscopy [Lefebvre *et al.*, 2001], see also <http://www.projectcybercell.com/>, located under Research, What's New. In these experiments, 3 different *E. coli* strains (MC4100, DW35/pH3 and DW35) with disrupted TCA cycles were grown on a glycerol minimal medium under aerobic conditions (2-l shaker flask, 37°C) for 35 hours. The MC4100 strain is a wild-type *E. coli* variant with a fully intact TCA cycle. The DW35/pH3 strain is a mutant strain in which the succinate dehydrogenase enzyme has been knocked out. The DW35 strain is a double knockout strain in which both the fumarate reductase and succinate dehydrogenase enzymes have been deactivated. Media samples (1.5 ml) were taken every hour, spun down in a micro-centrifuge (to remove the cells) and the supernatant frozen. These media samples were later thawed and then analyzed by 1D  $^1\text{H}$  NMR spectroscopy using the Eclipse metabolic analysis software package developed by Chenomx (<http://www.chenomx.com>) to identify and quantify organic components. The results of the experimental data and the SimCell simulations were compared qualitatively and quantitatively for general consistency and accuracy.

### The repressilator gene circuit

While metabolic simulations are of considerable importance, a growing number of cell modelers are turning to modeling genetic regulatory circuits involving repressors and activators, such as the *lac* and *trp* operons. A more challenging and interesting regulatory circuit is a synthetic oscillatory circuit called the repressilator [Elowitz and Leibler, 2000]. The repressilator is a 3-gene feedback circuit in which each expressed gene represses expression of a downstream gene (Figure 3). By engineering the gene products to have relatively short lifetimes, the feedback circuit enters into an oscillatory cycle in which the copy number of all three genes varies in 3 asynchronous phases. The repressilator has been successfully engineered into *E. coli* and the observed oscillatory behavior appeared to follow the predicted behavior as modeled using a stochastic differential equation [Elowitz and Leibler, 2000]. To simulate the repressilator circuit using SimCell, we constructed a grid containing 100 x 100 squares. Plasmid or DNA molecules containing the *Lacl*, *TetR* and  $\lambda\text{cl}$  genes and their corresponding operators were generated with the genes/operators placed in the same order as described in the original repressilator plasmid. Initial copy numbers of the *Lacl*, *TetR* and  $\lambda\text{cl}$  proteins were all set to zero, but with equal (linear) creation rates of 1.2 copies per second. Exponential decay rates were used to simulate concentration dependent proteolysis for the *Lacl*, *TetR* and  $\lambda\text{cl}$  proteins. The decay constants for all three protein products were set to 0.0008 per second. Binding affinities or, more particularly, off rates for the *Lacl*, *TetR* and  $\lambda\text{cl}$  repressors when bound to their respective operators were also set to 0.0008 for all three genes. A simplifying assumption in these simulations was that there was no delay in either transcription or translation to express/suppress the repressilator gene products. To investigate the effects of gene dosage and protein copy number we also increased the number of 3-gene repressilator plasmids from 1 to 10. In the 10 plasmid simulation the production rate for each plasmid was reduced to 0.12 copies per second.



**Figure 3:** A schematic diagram of the repressilator plasmid.  $\lambda cl$ , TetR and LacI are all gene repressing proteins that bind to their respective promoter/operator elements (marked as  $P_{xyz}$  in the diagram). From this diagram we see that LacI represses TetR production, TetR represses  $\lambda cl$  production and  $\lambda cl$  represses LacI production.

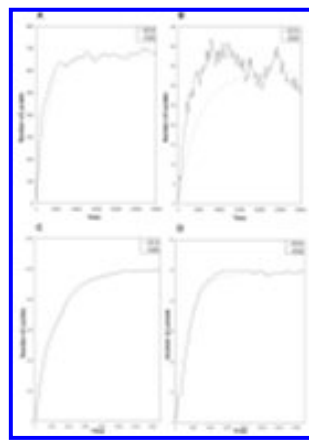
All simulations described here were performed on SimCell (version 1.0) and run on either a 1.2 GHz single processor PC with 512 MB RAM or a 1.0 GHz single processor Macintosh laptop with 256 MB RAM. Both the backend and front end of SimCell are written in Java. The program should be compatible with any computer operating system equipped with a Sun Microsystems Java compiler (version 1.4.2 or higher). SimCell is available at <http://wishart.biology.ualberta.ca/SimCell/>.

## Results

To summarize, SimCell was run on 4 very different cellular simulation scenarios including: 1) the modeling of simple enzyme kinetics; 2) the modeling of macromolecular diffusion in different environments; 3) the modeling of metabolism and metabolic mutants in the TCA cycle; and 4) the modeling of the repressilator oscillatory gene circuit [Elowitz and Leibler, 2000]. Here we present the results for each of these simulations along with comparisons to experimental results or competing simulation methods.

### Enzyme kinetics

Figure 4 summarizes the results of simulating the enzyme kinetics of lactate dehydrogenase (LDH) under various conditions. Four enzyme kinetic or enzyme progress graphs are shown. Figure 4A plots the kinetics of lactate synthesis as modeled with SimCell (solid line) and a standard best-fit exponential function (dashed line) using a starting concentration of 55 mM pyruvate and 55  $\mu$ M enzyme. A quick comparison between the two enzyme progress curves indicates that the SimCell kinetic curve displays a measurable degree of stochasticity or noise relative to the one calculated by solving the ODE. This noise has to do with the randomness of enzyme-substrate collisions that would be expected to occur when working with a particle-based method, such as SimCell. The stochasticity also arises from the relatively high level of reversibility for this reaction. If we reduce the starting concentration of pyruvate to below saturating levels (down to 2.7 mM), and run the same reaction in SimCell we see that the resulting enzyme progress curve (Fig. 4B) exhibits considerably more noise. As we might expect this noise arises purely from the effects of working with small numbers of interacting molecules. Clearly this kind of stochastic (but realistic) enzyme progress curve could not be generated using standard ODE's.



**Figure 4:** Enzyme progress curves for lactate dehydrogenase (LDH) plotted under various starting conditions. A) Enzyme progress curve calculated with a starting concentration of 1000 pyruvate molecules (55 mM) and 1 LDH molecule (55  $\mu$ M) showing the results from the SimCell simulation (dashed line) and the best-fit curve (solid line) where the forward reaction is 2x faster than the reverse reaction. B) Enzyme progress curve calculated with a starting concentration of 50 pyruvate molecules (2.7 mM) and 1 LDH molecule (55  $\mu$ M) showing the results from the SimCell simulation (dashed line) and the best-fit curve (solid line) where the forward reaction is 2x faster than the reverse reaction. C) Enzyme progress curve calculated and plotted using the same starting conditions as A, but with the forward reaction being 100x faster than the reverse reaction. D) Enzyme progress curve calculated and plotted using the same starting conditions as B, but with the forward reaction being 100x faster than the reverse reaction.

Interestingly, if we significantly reduce the reversibility of this reaction by making the forward reaction 100x faster than the reverse, we see that SimCell generates a nearly smooth kinetic curve (Figure 4C) that fits very closely to the dashed curve generated by a time-dependent exponential function. At these high levels of reactants and low degree of reversibility, the particulate behavior of the system begins to disappear, and the kinetic curve begins to resemble that modeled via continuum (ODE) methods. At reduced particle numbers (Figure 4D), we again see the effects of stochasticity or collisional noise. Overall, these results for lactate synthesis from LDH indicate that SimCell and related DCA methods can reproduce the classic enzymatic behavior seen in ODE's. We can also see that under dilute conditions, DCA models generate much noisier enzyme progress curves – which would be expected if experimental data were collected under these conditions. Indeed, these noisy kinetic plots appear to closely resemble the curves actually measured by enzymologists [Duggleby, 1995; Straathof, 2001] (<http://www.bt.tudelft.nl/content/download/encora.html>) as well as those generated by solving stochastic differential equations [Oksendal, 2002]. These results clearly suggest that DCA methods can be used as an alternative, non-mathematical route to "solve" both simple ODE's and stochastic DE's.

### Macromolecular diffusion

The application of DCA methods to analyzing enzyme kinetics nicely illustrates the quantitative and analytical utility of this approach. However, these relatively simple kinetic problems do not really do the method justice. In this next example we show how DCA methods can be used to address more difficult problems, such as those relating to the effects of excluded volume and transient binding on molecular diffusion. Heterogeneous diffusion phenomena like these are generally quite difficult to model as the particulate nature of these problems makes them quite intractable to solution by differential equations. However, these problems are ideal for DCA methods. Table 2 presents the results calculated by SimCell for modeling the diffusion of a protein using a variety of interaction and movement rules.

**Table 2:** Observed mean velocities versus calculated mean velocities for macromolecules under various movement rules (A-F) and varying levels of molecular crowding (0-50%).

Movement rule	# Moves	Collision	Sticky	Viscosity	Average diffusion rate (nm/ms)		
					0%	25%	50%

A	8	No	No	No	30.4	30.4	30.2
B	9	No	No	No	26.9	26.9	26.7
C	9	No	No	Yes	18.8	18.8	18.6
D	9	Yes	No	No	26.7	20.1	13.4
E	9	Yes	Yes	No	26.7	6.4	1.6
F	9	Yes	Yes	Yes	18.7	4.5	1.1
Observed					27.0	3.0	1.0

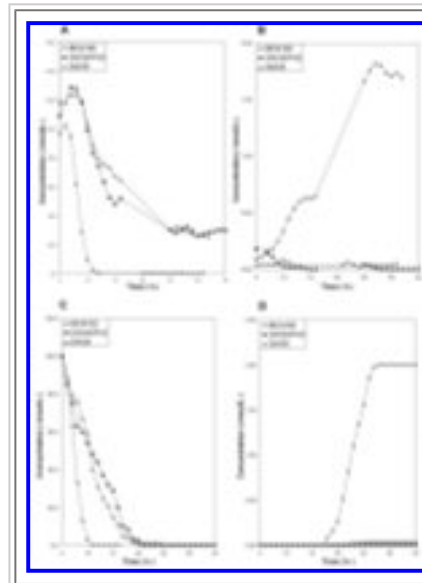
As indicated earlier we were specifically interested in modeling the effects of collision, binding or stickiness and microviscosity on protein diffusion rates in 3 different environments: pure water (0% crowding), the cytoplasm (25% crowding) and the lumen of the endoplasmic reticulum (50% crowding). For this particular simulation a total of 18 different movement rules or conditions were tested. This involved selecting a set of movement and interaction rules and then calculating the velocity of a tracking molecule over a defined number of DCA cycles. The calculated velocity of the tracking molecule in pure water (no viscosity, collision or sticking) was then calibrated to experimentally measured molecular velocity to see if the method could generate accurate predictions for protein diffusion rates or velocities in the other environments. From previously reported data [Jacobson and Wojcieszyn, 1984; Dayel *et al.*, 1999] it is known that macromolecules in free, aqueous conditions will move at a mean velocity of 27 nm/ms. This value was used to calibrate the mean protein velocity using the 9 set Brownian movement cycle in SimCell (rule B, 0% crowding in Table 2).

As seen from this table, when the movement rules require that the molecule cannot stay in the same square, the diffusion rate increases minimally by a factor of 9/8 or 1.125. Similarly, if the microviscosity of the solvent is increased by 30%, the velocity of the molecules drop by 30%. This simply reflects the 30% reduction in the probability of movement during any given movement step. However, when macromolecular interactions are included in the simulation we see much more interesting and dramatic effects on macromolecular diffusion. If collision effects are included, the tracking molecule is seen to slow down by an additional 25% (at 25% molecular density) and an additional 50% (at 50% molecular density). If binding or sticking effects are added to this we see the diffusion rates are slowed by an additional factor of 3.1 (at 25% molecular density) and by 8.4 (at 50% molecular density). Therefore the SimCell model predicts that if microviscosity (30%), crowding and binding are included, the mean velocity of a protein in the cytoplasm would drop from 27 nm/ms to 4.5 nm/ms. When compared the experimental measurements of Jacobson and Wojcieszyn, 1984, who report a velocity of 3.5 nm/ms for cytoplasmic diffusion, we can see the agreement between model and experiment is quite good. Likewise, the SimCell diffusion model predicts that when macromolecular concentrations reach 50% (as in the ER), the mean velocity drops to 1.1 nm/ms. If we compare this result to the experimental measurements of Dayel *et al.*, 1999, who report a value of 1.0 nm/ms for the mean velocity of green fluorescent protein in the endoplasmic reticulum, we again see the agreement is very good. It is interesting to note that SimCell predicts that of the three sources of diffusional slow-down (originally identified by Kao *et al.*, 1993) that macromolecular binding/sticking appears to have the greatest effect. This is a result that neither Kao *et al.* nor Dayel *et al.* were able to derive from their experimental measurements, although both groups speculated on what the major contributor might be.

### Metabolic simulation

These days most cellular simulations are generally less focused on explaining enzyme kinetics or macromolecular diffusion and more focused on describing metabolic networks and genetic circuits. These latter problems are intrinsically more difficult as they require solving coupled sets of ODE's or

PDE's to explain the temporal and/or spatial effects that are being measured. To investigate the utility of DCA methods for handling the complex metabolic simulations that are popular today, we used SimCell to model the TCA cycle in *E. coli*. Specifically we investigated the influence that single and double gene knockouts in the succinate/fumarate conversion steps would have on the consumption of glycerol and the production of succinate. Graphs comparing the consumption of glycerol (predicted and experimentally observed) of these metabolites over a 30 hour period for all three *E. coli* strains are shown in [Figure 5A](#) and [5B](#). Note that the first 5 hours are not plotted in these graphs as the cell densities were insufficient to have any measurable impact on the metabolite concentrations in the growth media. As can be seen in [Figure 5A](#), the observed consumption of glycerol by the wild type MC4100 grows exponentially with most of the glycerol being consumed within 5 hours. For the other *E. coli* strains, the glycerol consumption was much slower, reflecting their 5-fold slower rate of growth. The SimCell model shown in [Figure 5c](#) predicts a similar exponential drop in glycerol concentration for the wild type MC4100 strain that matches closely with the observed data. When we look at the single knock-out (DW35/pH3) and double knock-out (DW35) strains in [Figure 5C](#) we see that SimCell predicts a significantly slower consumption of glycerol, in line with the overall lower abundance of TCA enzymes. Comparison to the corresponding glycerol consumptions curves shown in [Figure 5A](#) indicates that the SimCell predictions match reasonably well with those seen in our metabolomic experiments. Given the many simplifications and abstractions used in the SimCell model, this result is quite encouraging.



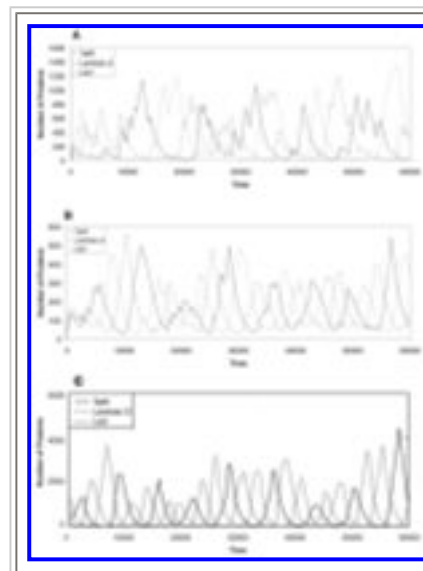
**Figure 5:** Comparison between SimCell calculated and experimentally observed glycerol consumption and succinate production curves for TCA cycle mutants. A) Plot of experimentally measured consumption of glycerol for MC4100, DW35/pH3 (single knockout) and DW35 (double knockout) as measured over 30 hours. B) Plot of experimentally measured succinate production for the same 3 *E. coli* strains. C) Plot of SimCell calculated glycerol consumption for the same cell strains assuming a doubling time of 1 hour for MC4100 and a doubling time of 5 hours for DW35/pH3 and DW35. D) Plot of SimCell calculated succinate production for the same 3 *E. coli* strains with the same growth rate assumptions.

If we look at the experimental results obtained for succinate production ([Figure 5B](#)) we see that both the wildtype (MC4100) and the single knockout (DW35/pH3) strains are able to process succinate completely. The single knockout strain is able to do so because the remaining fumarate reductase enzyme is an isozyme (38% sequence identity) to succinate dehydrogenase and is able to perform the "reverse" reaction (succinate to fumarate) with moderate efficiency. On the other hand, for the double knockout strain (DW35), succinate begins to accumulate in the media very early on and continues to rise to an abnormally high concentration of 1.8 mM. This is because the cell has no capacity to process succinate, either through succinate dehydrogenase or fumarate reductase. If we look at the SimCell results shown in [Figure 5D](#), we see largely the same phenomena. The wildtype (MC4100) and single knockout (DW35/pH3) are predicted to process succinate fully while the double knockout (DW35) is predicted to continually produce succinate until it tops out after approximately 25 hours. Note that in modeling the single knockout strain we reduced the efficiency of the fumarate reductase enzyme by 50% to reflect its reduced succinate dehydrogenase activity [[Dickie and Weiner, 1979](#)].

Overall, we can see that SimCell and its DCA modeling approach are able to qualitatively match the observed metabolic outcomes of this real-life metabolic experiment. As with the enzyme kinetic model, we see that all the predictions generated by SimCell have a characteristic noise or stochasticity to them. As before, this noise arises because of the particulate or discrete nature of the DCA simulation technique. Interestingly, this predicted noise seems to mirror that seen in the experimental data. No doubt other metabolic modeling techniques could have generated very similar predictions, but largely without the stochastic noise seen in these SimCell graphs. Indeed, even without doing the full-scale simulation, a modestly trained biochemist could have predicted what most of the metabolic outcomes would be in this experiment. However, it is important to emphasize that metabolic simulation shown here is intended to be an illustrative example only. Indeed, this example's primary purpose is not to predict something novel, but to show that DCA methods are fully capable of analyzing or processing complex metabolic networks in the same way that more established approaches can.

### The repressilator

So far we have seen how DCA methods can be used to model problems of varying complexity in enzyme kinetics, diffusion and metabolism. Here we show how DCA methods can be used to model a genetic circuit – specifically the repressilator. This elegant *E. coli* gene circuit has been well-studied, both experimentally and computationally [Elowitz and Leibler, 2000]. The fact that it generates a triphasic oscillatory output makes this genetic circuit particularly compelling to model. Elowitz and Leibler have already demonstrated how the repressilator can be adequately modeled using either a simple ODE or a stochastic ODE approach. As we show in Figure 6 it is also possible to model the repressilator using DCA methods. Specifically, in Figure 6A we show that SimCell can be used to model the oscillatory behavior for the *λcl*, *lacI* and *tetR* genes using a single copy of the three-gene repressilator plasmid. Note that the oscillations have a distinct "sawtooth" shape to them. This arises because of the abstractions used in SimCell to have linear protein creation rates and exponential protein decay rates. The linear creation rates give a linear slope on the left of each "tooth" while the exponential decay rate gives a curvilinear slope on the right of each tooth.



**Figure 6:** Comparison between SimCell calculated expression levels for the repressilator gene circuit and the expression levels originally calculated by Elowitz and Leibler using a stochastic differential equation. A) SimCell calculated expression levels for a repressilator gene circuit with 1 plasmid. B) SimCell calculated expression levels for a repressilator gene circuit with 10 plasmids. C) Calculated expression levels for repressilator gene circuit as modeled by a stochastic differential equation (figure adapted from Elowitz and Leibler, 2000). The time is given in seconds.

In Figure 6B, the same model is constructed using 10 plasmid copies. By increasing the gene dosage (and protein copy numbers) we see two things. First, the noisy oscillations in Figure 6a are somewhat smoothed out, giving a more pleasing sinusoidal appearance to the curves in 6b. Second, the

oscillations have a less sawtoothed shape and a more sinusoidal appearance. This results from the summing or averaging of multiple creation and decay processes happening in different repressilator plasmids located throughout the cell. Note that in both [Fig. 6A](#) and [6B](#), the oscillations in protein copy number are not perfectly periodic and that they occasionally break down into chaotic or disorganized patterns. However, in every case, the oscillatory behavior eventually recovers and it is typically maintained for very long periods of time (data not shown). In [Figure 6C](#) we illustrate the gene expression oscillations predicted by Elowitz and Leibler via a stochastic differential equation. Note the overall similarity in the curves, phase and period between [Figs. 6B](#) and [6C](#). Also note that the noise seen in the graphs generated by SimCell are generally greater than those seen in the stochastic DE graphs generated by Elowitz and Leibler. Interestingly, visual comparison with the actual experimental data presented by [Elowitz and Leibler, 2000](#), suggests that the true behavior of repressilator cells is more akin to that generated by the DCA model. Nevertheless, the key point here is to emphasize the fact that DCA methods can be used to model complex genetic circuits and that their performance is as good, and perhaps better, than other conventional modeling approaches. It is also important to point out that DCA methods are capable of generating elegant temporal and spatial patterns. These complex patterns can arise from a disordered set of randomly moving particles following a simple set of pairwise interaction rules. The fact that these rules are biologically based and that the interactions encoded by these require no special "intelligence" suggests that other kinds of interesting patterns may be conceived of or modeled by DCA methods.

---

## Discussion

As seen from the results presented here, we have used DCA methods to successfully model 4 fundamentally different but important cellular phenomena for which experimental data are available. These include: 1) the modeling of simple enzyme kinetics; 2) the modeling of macromolecular diffusion in different environments; 3) the modeling of metabolism and metabolic mutants in the TCA cycle; and 4) the modeling of the repressilator oscillatory gene circuit. These models effectively span the range of most kinds of modern cellular simulations, from simple chemical kinetics to more complex metabolism and genetic circuits. In many respects the results described here are comparable those that could be obtained by solving coupled sets of ODE's or stochastic ODE's. However, this ignores the fact that the DCA approach also offers important spatial and temporal information that could not be obtained by solving ODE's or PDE's. Specifically, each simulation produced by SimCell generates a record of the time dependent distribution of all the molecular components in the system. Indeed the ability to generate, visualize and analyze molecular "movies" is perhaps one of the most useful features of DCA.

The fact that DCA methods are particularly good at generating accurate spatial models is a key point that is somewhat under-emphasized in the 4 examples given in this paper. Many cellular phenomena, ranging from simple processes such as reaction-diffusion processes to septum formation, cell division, chemotaxis, calcium waves, excitation potentials, and embryogenesis are fundamentally spatial phenomena. SimCell was specifically designed to accommodate these spatially dependent processes (primarily in 2 dimensions) and we plan to present additional results showing how SimCell can be used simulate a number of complex spatial phenomena in a future publication. It is also worth noting that DCA methods are not restricted to spatial modeling in 2 dimensions. Indeed, DCA methods are fully compatible with 3-dimensional grids and 3-dimensional modeling. Recently, 3-dimensional DCA methods with subnanometer grid sizes were used to model such spatially dependent phenomena as membrane growth and blastula formation. They were also used to investigate osmotic shock and to accurately calculate Young's modulus for cell membranes [Ellison, personal communication].

Another key feature that must be emphasized is the fact that DCA methods are highly scalable. This scalability arises primarily from the fact that Brownian motion – which is fundamental to the DCA

concept – is highly scalable too ([http://math.mit.edu/~kang/bm/bm\\_harmonic\\_measure.htm](http://math.mit.edu/~kang/bm/bm_harmonic_measure.htm)). Brownian motion, whether traced at the nanometer level or the millimeter level, or whether analyzed in 1, 2 or 3 dimensions is remarkably similar (i. e. scale invariant) at all levels or dimensions. This means that a DCA simulation run with a presumptive grid size of 3 nm and a time step of 1 ms, should also mimic phenomena where the grid size was 3  $\mu\text{m}$  and the time step was 1 s, or phenomena where the grid size was 3  $\text{\AA}$  and the time step was 100  $\mu\text{s}$  (or smaller). This scale invariance means that DCA methods can potentially be used to model phenomena ranging in size from Angstroms to meters and time scales from picoseconds to days. In fact, this kind of temporal and spatial rescaling was actually done in several simulations presented here, including the TCA cycle and repressilator simulations where processes were modeled over the equivalent of hours or even days in less than 1 CPU hour.

DCA methods not only allow rescaling of time and space, but they also allow scaling of particle numbers. For instance one can rescale grid occupancy so that more than 100 macromolecules occupy any given grid square. Alternately one can shrink the grid occupancy to as little as 0.1 molecule/square. This occupancy or particle scaling can be done either in terms of system dilution or in terms of grid size rescaling. Obviously as the grid sizes and time steps are increased (i. e. the granularity is expanded), there is bound to be some loss of precision in the model. Similarly, as the grid sizes and time steps are decreased (fine-grained simulations), one is likely to gain precision – but usually with a concomitant loss in computational speed or efficiency. Finding the right balance between precision and computer time is perhaps one of the more awkward and least refined aspects of the DCA method.

The DCA method draws from several pre-existing computer simulation concepts, including cellular automata (CA), agent-based modeling and Brownian dynamics. Whether DCA is completely distinct from these 3 concepts is somewhat debatable. However, we believe DCA methods differ from traditional CA models in at least 3 ways. First, in DCA approaches, not all objects need to be identical in size, shape or assigned properties as they are in classical CA. This kind of particle heterogeneity means that a DCA system will not easily fall into a steady state – as a CA system invariably does. Second, in DCA methods the system is neither expected nor is it required to evolve to a steady state as is seen in conventional CA approaches. In particular, DCA is not solely limited to generating or modeling interesting steady-state spatial patterns. In fact, DCA approaches can be used to model much more, including non-steady state temporal, spatial and spatio-temporal patterns or phenomena. In other words, DCA methods can be used to model unstable, continuously changing variables and events. Third, and perhaps most importantly, the movement of objects in a DCA is expected to mimic realistic Brownian or random walk motion. This is not the case with CA where the placement of objects with each time step does not have to follow any physical laws. The result is that DCA methods have the capacity to generate complex, continuously evolving temporal and spatial patterns, instead of just steady spatial patterns as is found with CA. DCA methods also appear to differ from most multi-agent systems [Wooldridge, 2002]. Like DCA methods, multi-agent simulation systems permit multiple different agents or objects to move and interact in a visually realistic fashion. However, in agent-based approaches, the objects are given some "intelligence" or capacity to communicate and learn. Consequently multi-agent systems can often be used to simulate flocking behavior in birds, schooling in fish or food-finding behavior in ants [Reeves, 1983; Reynolds, 1987] (<http://www.red3d.com/cwr/boids/>). Clearly we cannot ascribe any intelligence, communication ability or learning capacity to macromolecules. Consequently, in DCA we are limited to describing or acting on events that arise only through physical contact or bumping. This may seem like a small distinction, however, it is important. In fact, it serves to emphasize the fact that DCA modeling is limited to modeling non-intelligent, non-communicating entities interacting under relatively strict physical laws, while multi-agent systems are targeted to modeling intelligent, fully communicating entities that do not have to adhere to any known physical laws.

## DCA limitations

While there are many positive features about dynamic cellular automata, it is important to note that DCA methods are not without their limitations. For instance, one inherent shortcoming is the fact that most DCA objects are inherently simple (squares, cubes, circles, spheres) and relatively featureless. While some surface features or sidedness can be ascribed to DCA entities, it is relatively difficult to create a DCA object that would match the atomic spatial complexity of something like an enzyme. Obviously DCA methods should not be used to predict atomic or molecular properties, such as binding sites, active sites,  $K_{cat}$ ,  $K_d$ ,  $pI$  or  $\Delta G$ . Rather, these properties need to be calculated, measured or predicted via experimental observation or MD simulations. The fact that individual molecular properties are not predictable from DCA methods is one obvious shortcoming. On the other hand, DCA methods are particularly good at predicting the collective properties of a system, once all the individual properties of the components are known. The capacity to predict or model collective properties from individual properties or components is perhaps the most useful or compelling feature of DCA.

Another curious shortcoming to DCA lies in the fact that it often takes proportionately more work or CPU time to model something simple than it does to model something complex. For instance, it is trivial to solve the ODE's for simple enzyme kinetics using a pencil and a piece of paper. Typically one can write down the solution and plot the correct curve by hand in a minute or two. Alternately one can let a computer do it in just a few seconds. However, if one tried to model some simple enzyme kinetics via DCA (as we did here), it could take up to 5 minutes of CPU time to generate a useful result. On the other hand, at least 2 of the more complex DCA simulations shown in this paper, including those involving cytoplasmic diffusion and the TCA cycle took either substantially less CPU time or substantially less set-up time than the simple enzyme kinetics simulation. Clearly, one should be selective in choosing when to use DCA methods and when to use alternative approaches, such as ODE's or SOE's. Typically, more detailed high-resolution models (like DCA) tend to require additional computation but also deliver much higher resolution. This should be understood when comparing analytical solutions to ODE devoid of spatial resolution with high resolution DCA. The modeling technique is generally selected to suit the problem, with simplified problems (i. e. well-stirred continuum assumptions) being most efficiently handled by less complex models.

Because DCA methods generate a complete movie of all particles and their motions, there is always the possibility that DCA simulations are generating too much information or too much data that is potentially irrelevant to the problem at hand. This "information overload" can make DCA methods computationally inefficient – both in terms of the simulation time and the effort needed to analyze the data. Obviously, if one is attempting to simulate something where spatial dependence is relatively unimportant or one is modeling something that could be easily described by ODE's or SOE's, then DCA methods are probably not the best choice. However, if one is uncertain about the mathematics of the problem or if the spatial dependence is of interest, then DCA approaches are probably a good choice.

---

## Conclusion

Dynamic cellular automata (DCA) appear to offer a simple, but powerful approach to cellular simulation. We have described a freely available program (SimCell) that implements many DCA principles and have demonstrated that this approach is able to qualitatively and quantitatively model a diverse range of cellular phenomena. There are a number of features about DCA which may be particularly appealing both to biologists and computer scientists. These include: their simplicity, their intuitive and relatively non-mathematical structure, their scalability, their visual or animated quality, their amenability to modeling many different and complex phenomena, their robustness, their capacity to deal with

discontinuities, irregular geometries and state changes, their capacity to capture stochasticity or randomness, their ability to capture or model non-steady state phenomena, and their amenability to computational parallelism. Overall, we believe DCA methods may offer an attractive alternative or may serve as useful complement to current methods being used in cellular simulations and systems biology.

---

## Acknowledgements

The authors wish to thank Jasmine Cruz for her assistance in preparing [Figure 6](#). Financial support from PENCE Inc., CIHR-Rx&D and from Genome Prairie (a division of Genome Canada) is gratefully acknowledged. We also wish to thank Joel Weiner, Gordon Broderick and Michael Ellison for helpful discussions.

---

## References

- 
- [Ayton, G. S., Tepper, H. L., Mirijanian, D. T. and Voth, G. A. \(2004\). A new perspective on the coarse-grained dynamics of fluids. \*J. Chem. Phys.\* \*\*120\*\*, 4074-4088.](#)

---

  - [Dayel, M. J., Hom, E. F. and Verkman, A. S. \(1999\). Diffusion of green fluorescent protein in the aqueous-phase lumen of endoplasmic reticulum. \*Biophys. J.\* \*\*76\*\*, 2843-2851.](#)

---

  - [Dickie, P. and Weiner, J. H. \(1979\). Purification and characterization of membrane-bound fumarate reductase from anaerobically grown \*Escherichia coli\*. \*Can. J. Biochem.\* \*\*57\*\*, 813-821.](#)

---

  - [Duggleby, R. G. \(1995\). Analysis of enzyme reaction progress curves by nonlinear regression. \*Methods Enzymol.\* \*\*249\*\*, 60-91.](#)

---

  - [Elowitz, M. B. and Leibler, S. \(2000\). A synthetic oscillatory network of transcriptional regulators. \*Nature\* \*\*403\*\*, 335-338.](#)

---

  - [Endy, D. and Brent, R. \(2001\). Modelling cellular behaviour. \*Nature\* \*\*409\*\*, 391-395.](#)

---

  - [Endy, D., You, L., Yin, J. and Molineux, I. J. \(2000\). Computation, prediction, and experimental tests of fitness for bacteriophage T7 mutants with permuted genomes. \*Proc. Natl. Acad. Sci. USA\* \*\*97\*\*, 5375-5380.](#)

---

  - [Ermentrout, G. B. and Edelstein-Keshet, L. \(1993\). Cellular automata approaches to biological modeling. \*J. theor. Biol.\* \*\*160\*\*, 97-133.](#)

---

  - [Gardner, M. \(1970\). Mathematical Games. \*Sci. Am.\* \*\*223\*\*, 120-123.](#)

---

  - [Ishii, N., Robert, M., Nakayama, Y., Kanai, A. and Tomita, M. \(2004\). Toward large-scale modeling of the microbial cell for computer simulation. \*J. Biotechnol.\* \*\*113\*\*, 281-294.](#)

---

  - [Jacobson, K. and Wojcieszyn, J. \(1984\). The translational mobility of substances within the cytoplasmic matrix. \*Proc. Natl. Acad. Sci. USA\* \*\*81\*\*, 6747-6751.](#)

---

  - [Javed, M. D., Azimuddin, S. M. I., Hussain, A. N., Ahmed, A. and Ishaq, M. \(1997\) Purification and characterization of lactate dehydrogenase from \*Varanus\* liver. \*Exp. Mol. Med.\* \*\*29\*\*, 25-30.](#)

---

- Kao, H. P., Abney, J. R. and Verkman, A. S. (1993). Determinants of the translational diffusion of a small solute in cytoplasm. *J. Cell Biol.* **120**, 175-184.

---

- Karplus, M. and McCammon, J. A. (2002). Molecular dynamics simulations of biomolecules. *Nat. Struct. Biol.* **9**, 646-652.

---

- Lefebvre, B. A., Abera, A., Epstein, N., Rothery, R., Weiner, J. A. and Wishart, D. S. (2001). Metabolomics of *E. coli* using High Throughput Quantitative <sup>1</sup>H NMR Spectroscopy. Proceedings of the 1st annual Cambridge Healthtech Conference on Metabolic Profiling, Chapel Hill, NC, USA.

---

- McCammon, J. A. (1987). Computer-aided molecular design. *Science* **238**, 486-491.

---

- Meir, E., von Dassow, G., Munro, E. and Odell, G. M. (2002). Robustness, flexibility, and the role of lateral inhibition in the neurogenic network. *Curr. Biol.* **12**, 778-786.

---

- Oksendal, B. K. (2002) Stochastic Differential Equations: An Introduction with Applications, 5th edition. Springer-Verlag, Berlin.

---

- Qian, H. (2002). From discrete protein kinetics to continuous Brownian dynamics: a new perspective. *Protein Sci.* **11**, 1-5.

---

- Reeves, W. (1983). Particle Systems - A Technique for Modeling a Class of Fuzzy Objects. *In: SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, Tanner, P. P. (ed.), ACM Press, NY, USA, pp. 359-376.

---

- Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *In: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, Stone, M. C. (ed.), ACM Press, NY, USA, pp. 25-34.

---

- Shen, M. Y. and Freed, K. F. (2002). Long time dynamics of Met-enkephalin: comparison of explicit and implicit solvent models. *Biophys. J.* **82**, 1791-1808.

---

- Slepchenko, B. M., Schaff, J. C., Macara, I. and Loew, L. M. (2003). Quantitative cell biology with the Virtual Cell. *Trends Cell Biol.* **13**, 570-576.

---

- Straathof, A. J. J. (2001). Development of a computer program for analysis of enzyme kinetics by progress curve fitting. *J. Mol. Catal. B: Enzym.* **11**, 991-998.

---

- Sundararaj, S., Guo, A., Habibi-Nazhad, B., Rouani, M., Stothard, P., Ellison, M. and Wishart, D. S. (2004). The CyberCell Database (CCDB): a comprehensive, self-updating, relational database to coordinate and facilitate *in silico* modeling of *Escherichia coli*. *Nucleic Acids Res.* **32**, D293-D295.

---

- Sveiczler, A., Tyson, J. J. and Novak, B. (2004). Modelling the fission yeast cell cycle. *Brief. Funct. Genomic. Proteomic.* **2**, 298-307.

---

- Warshel, A. (2002). Molecular dynamics simulations of biological reactions. *Acc. Chem. Res.* **35**, 385-395.

---

- Wolfram, S. (2002). A New Kind of Science. Wolfram Media, Champaign, Ill.

---

- Wooldridge, M. (2002). An Introduction to Multiagent Systems. John Wiley and Sons Press, Chichester, UK.