Bioinformatics, 33(5), 2017, 710–717
doi: 10.1093/bioinformatics/btw700
Advance Access Publication Date: 6 December 2016
Original Paper

OXFORD

Systems biology

Smoldyn: particle-based simulation with rule-based modeling, improved molecular interaction and a library interface

Steven S. Andrews[1,2]

[1]Basic Sciences Division, Fred Hutchinson Cancer Research Center, Seattle, WA 98109, USA and [2]Isaac Newton Institute for Mathematical Sciences, Cambridge CB3 0EH, UK

Associate Editor: Jonathan Wren

Received on May 16, 2016; revised on October 18, 2016; editorial decision on October 19, 2016; accepted on November 3, 2016

Abstract

Motivation: Smoldyn is a spatial and stochastic biochemical simulator. It treats each molecule of interest as an individual particle in continuous space, simulating molecular diffusion, molecule-membrane interactions and chemical reactions, all with good accuracy. This article presents several new features.

Results: Smoldyn now supports two types of rule-based modeling. These are a wildcard method, which is very convenient, and the BioNetGen package with extensions for spatial simulation, which is better for complicated models. Smoldyn also includes new algorithms for simulating the diffusion of surface-bound molecules and molecules with excluded volume. Both are exact in the limit of short time steps and reasonably good with longer steps. In addition, Smoldyn supports single-molecule tracking simulations. Finally, the Smoldyn source code can be accessed through a C/C++ language library interface.

Availability and Implementation: Smoldyn software, documentation, code, and examples are at http://www.smoldyn.org.

Contact: steven.s.andrews@gmail.com

1 Introduction

Biological cells exhibit dynamic spatial organization, at least some of which is essential for cell survival. The precisely choreographed steps of cell division and the elaborate architectures of individual kinetichores are examples. The intricacy of this organization makes it difficult to study experimentally, despite recent advances in cell imaging, making computer modeling an important research component. Such modeling requires the right software tools, which in this case are programs that can accurately simulate spatial organization within cells.

Several classes of simulators have been developed for this purpose (see reviews Andrews *et al.*, 2009; ElKalaawy and Amr, 2015; Schöneberg *et al.*, 2014; Takahashi *et al.*, 2005). *Deterministic* simulators return the same results every time they are run, typically ignoring biological stochasticity. For example, the Virtual Cell software was initially written to numerically integrate the deterministic partial differential equations that describe reaction-diffusion processes (Schaff *et al.*, 1997). *Stochastic* simulators, in contrast, account for biological stochasticity using pseudo-random number generators to compute results that differ each time but that sample from the statistical distribution of results for the given model. Within this latter class, (i) *mesoscopic lattice-based* simulators, such as MesoRD (Hattne *et al.*, 2005) and URDME (Drawert *et al.*, 2012), represent space using lattices and treat molecules of interest (hereafter just molecules) as well-mixed populations within the lattice sites. They typically advance the simulation time by moving from event to event using adaptations of Gillespie's algorithms (Gillespie, 1977; Stundzia and Lumsden, 1996). They are typically most useful for models with many identical molecules ($>10^5$) and coarse spatial resolution (>10 nm). (ii) *Microscopic lattice-based* simulators, such as Spatiocyte (Arjunan and Tomita, 2010), also represent space using lattices but only allow up to one molecule per site. They offer

finer spatial resolution, making them particularly good for modeling macromolecular crowding (e.g. Saxton, 1987) and protein aggregation (e.g. Goldman *et al.*, 2004), but run more slowly and are prone to artifacts from the lattice (e.g. Andrews *et al.*, 2015). And (iii) *particle-based* simulators, such as MCell (Stiles and Bartol, 2001), ReaDDy (Schöneberg and Noé, 2013) and Smoldyn (Andrews *et al.*, 2010), represent space continuously, treat molecules as individual particles, and typically advance the simulation time with fixed steps. Their spatial resolution can be as small as the sizes of single molecules.

MCell and Smoldyn work at a similar level of detail, treating molecules as spheres and ignoring intermolecular forces. MCell's strengths include its graphical user interface, elegant graphical output and facility with complicated membrane geometries (e.g. Bartol *et al.*, 2015). It has found substantial use in the neuroscience community (e.g. Kinney *et al.*, 2013; Stiles *et al.*, 1996). Smoldyn's strengths include its high accuracy (Andrews, 2009; Andrews and Bray, 2004), fast computation, and ease of use (Andrews *et al.*, 2010; Andrews, 2012). Smoldyn has been used most often for modeling cell systems (e.g. Jilkine *et al.*, 2011; Khan *et al.*, 2012; Subburaj *et al.*, 2015) and biophysical problems (e.g. Schmidt *et al.*, 2014; Strongin *et al.*, 2014; Zavala and Marquez-Lago, 2014).

In Smoldyn's underlying reaction model, molecules diffuse with ideal Brownian motion and react upon collision. Its algorithms also follow this design, but the resulting dynamics are slightly different due to its use of finite time steps (Andrews and Bray, 2004). The Green's function reaction dynamics (GFRD) (Takahashi *et al.*, 2010; van Zon and Ten Wolde, 2005) and free-propagator reweighting (FPR) methods (Johnson and Hummer, 2014) avoid these differences, making them slightly more accurate (within the approximation that molecules are perfect spheres). However, they often run slowly; GFRD ran several orders of magnitude slower than Smoldyn in one comparison (Andrews *et al.*, 2015) and the description of the FPR method (Johnson and Hummer, 2014) shows that it uses shorter time steps and performs more computations per step than Smoldyn does.

I initially released Smoldyn in 2003 and have continued to develop it since then. This article describes recent additions to the software, including support for rule-based modeling, molecules with excluded volume, accurate on-surface diffusion, single-particle tracking and an application programming interface (API). Each is a small part of the whole program but nevertheless significant in its own right. Together with the core software (Andrews *et al.*, 2010) and other additions (Robinson *et al.*, 2015), these create a powerful cell biology modeling tool. The features described here are illustrated using example files in the S94_archive/Andrews_2016/directory of the Smoldyn download package.

2 Results and discussion

2.1 Species groups and wildcards

One aspect of subcellular organization concerns the structures of protein complexes, such as the kinetochores mentioned earlier. More generally, many proteins have variable post-translational *modifications*, such as phosphate groups, nucleotide cofactors and conformational states, and/or form transient multimeric complexes (Mayer *et al.*, 2009).

Most simulators represent these modifications and complexes with *narrowly defined species*, in which each variant is treated as a separate species. This is conceptually simple, but the number of species increases combinatorially with the possible modifications and complexations, so it can be impractical to enumerate them by hand. To alleviate this, *rule-based modeling* methods automatically generate the possible species and reactions from simple *reaction rules* (see Stefan *et al.*, 2014). The BioNetGen software (Blinov *et al.*, 2004), Moleculizer software (Lok and Brent, 2005), and κ-calculus (Danos and Laneve, 2004) represent three approaches. However, if the entire reaction network is generated before the simulation, in the *generate-first* approach, then it can still become too large; e.g. polymerization reactions lead to an infinite number of species (Chylek *et al.*, 2014; Blinov *et al.*, 2005). This problem is alleviated by the *on-the-fly* approach, in which the reactions for a species are only generated when a molecule of that species has arisen in the simulation.

Modifications and complexes can also be represented with *broadly defined species*, in which each molecule carries binding information with it, often including a graph its subunits and each of their modifications. In the limit, even the concept of a species, meaning a class of molecules that all behave identically, becomes unnecessary (e.g. Bray and Lay, 1997; Sneddon *et al.*, 2011). Spatial simulators often use approaches in between these extremes (see Gruenert *et al.*, 2010; Tolle and Le Novère, 2010). For example, MCell supports 'slots' for molecules, with which users can create modifications or enable complexation (Stefan *et al.*, 2014).

Smoldyn supported rule-based modeling using the Libmoleculizer module (Andrews *et al.*, 2010) but this did not work satisfactorily. Thus, I replaced it with new methods, described here. I kept Smoldyn's species narrowly defined: molecules of a species can only differ in their surface-binding *state*, meaning whether the molecule is in solution, is bound to the front or back of a surface, or spans the surface in an 'up' or 'down' orientation. I also kept molecules as always being spherical.

Smoldyn now supports *species groups*, which are user-definable sets of species. For example, a modeler can define a species group for a specific protein with individual species within the group representing the possible modification states. The modeler can then define diffusion coefficients or graphical display parameters for an entire family of species at once while also making exceptions as needed.

These species groups are most easily defined using wildcards in species names, much as computer users use wildcards to specify groups of files. For example, suppose a model includes proteins Fus3, Fus3p and Fus3pp, where each 'p' represents a phosphate. Using the '*' wildcard, which matches to any 0 or more characters, Fus3* denotes the group of all three species. Table 1 lists the wildcards that Smoldyn supports. They include the standard '*', '?' and '[...]' operating system wildcards, the less common '|', which is an OR operator, and '&', which is a new permutation operator.

Smoldyn also supports wildcards in chemical reactions, using the method that wildcards in reaction products get substituted with the matching text in the reactants. For example, suppose a model includes the protein Ste5, which can bind to Fus3 in any phosphorylation state. Binding between them could then be defined with the reaction rule Ste5 + Fus3* $\rightarrow$ Ste5Fus3*. These reaction rules can produce new species names. If this happens, and if the user permits it, Smoldyn includes the new species in the model. These new species can then be reactants in the reaction rules, leading to further network expansion, making this is a form of rule-based modeling. Smoldyn supports it with both the generate-first and on-the-fly methods (Andrews, 2016).

Smoldyn defines the diffusion coefficients, surface interaction rates, and graphical display parameters for generated species from those of the reactants. These definitions are based on the assumptions that the reactants diffuse as though they are roughly spherical

Table 1. Smoldyn wildcards

Symbol	Meaning	Matching example	Reaction example
?	any 1 character	A? matches AB, AC, etc.	A? → B?
*	0 or more characters	A* matches A, AB, etc.	A* → B*
[...]	1 listed character	A[a-c] matches Aa, Ab, Ac	A[u,p] → B[0,1]
|	OR operator	A|B|C matches A, B, C	A|B → a|b
&	permutation	A&B matches AB, BA	A&B → a&b
{...}	grouping	A{B|C} matches AB, AC	A{b|c}→ A{c|b}

and their volumes add upon binding, so the effective radius of a molecule scales as the cube root of its volume. Thus, for the generic reaction A + B → AB, the product radius is

$$r_{AB} = (r_A^3 + r_B^3)^{1/3} \quad (1)$$

where r_A and r_B are the reactant radii. Smoldyn computes the product's graphical display radius from this equation. Using the further assumption that diffusion coefficients scale as the inverse of a molecule's radius, from the Stokes-Einstein equation, which is reasonably accurate even within cells (Andrews, 2012; Dix and Verkman, 2008), Smoldyn computes product diffusion coefficients as

$$D_{AB} = (D_A^{-3} + D_B^{-3})^{-1/3} \quad (2)$$

where D_A and D_B are the reactant diffusion coefficients (see Tolle and Le Novère, 2010). Smoldyn computes the product's display color using a weighted average of the reactant colors. For each of the red, green and blue brightness values, Smoldyn computes the product brightness using

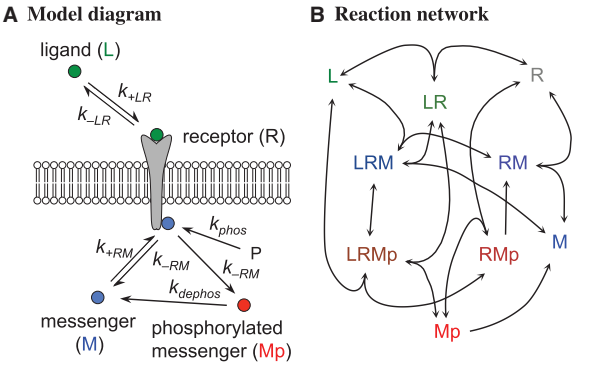
$$v_{AB} = \frac{r_A v_A + r_B v_B}{r_A + r_B} \quad (3)$$

where v_A and v_B are the reactant brightness values. Finally, Smoldyn determines surface interactions for products using the method that the new species behaves like the reactant that has the 'greater action', where the possible actions are ordered with increasing value as: transmission, reflection, absorption and porting (which is for hybrid simulations). For example, if a surface reflects reactant A and transmits reactant B, then reflection is the greater action, so the surface reflects product AB. All of these definitions can be overridden by the user.

Smoldyn performs on-the-fly simulation by tagging species when a molecule of that species first arises in the simulation. At the end of each time step, Smoldyn inserts the names of these tagged species in the reaction rules to generate the corresponding reactions and products.

If Smoldyn encounters multiple reactions that form the same products, which often happens with symmetric species, Smoldyn adds the reaction rates. For example, consider a model for a polymer of 'A' subunits, where species A is a monomer, AA is a dimer, AAA is a trimer etc. If the reaction rule for this model is '* + * ↔ **', where the forward reaction indicates that any two polymers can associate and the reverse reaction indicates that a polymer can break at any position (this needs to be simulated on-the-fly because the reaction network is infinite), then dissociation of AAA can produce the A and AA products in either of two ways, which are for dissociation of the left-most and right-most subunits. In this case, Smoldyn adds the two elementary reaction rates to create the simulated reaction rate.

I validated Smoldyn's wildcard rule-based modeling using several models, in each case expanding them manually, using BioNetGen (see below), and with wildcards. The model shown in



C Model rules using wildcards

```
L(fsoln) + R*(up) <-> LR*(up) KLRON KLROFF
*R(up) + M*(bsoln) <-> *RM*(up) KRMON KRMOFF
LRM(up) -> LRMp(up) KPHOS
Mp(soln) -> M(soln) KUNPHOS
```

D Model rules using BioNetGen

```
L(l2r) + R(r2l) <-> L(l2r!1).R(r2l!1) krlon,krloff
R(r2m) + M(m2r) <-> R(r2m!1).M(m2r!1) krmon,krmoff
R(r2l!+,r2m!1).M(m2r!1,psite~u) -> \
          R(r2l!+,r2m!1).M(m2r!1,psite~p) kphos
M(psite~p) -> M(psite~u) kunphos
```

Fig. 1. Model of second messenger signaling. **(A)** Model cartoon, showing that membrane-bound receptors, R, can bind ligands, L and/or messenger proteins, M; the latter get phosphorylated when both are bound and dephosphorylate spontaneously. **(B)** The complete reaction network, comprising 9 species and 18 reactions. **(C)** Rules for model generation using wildcards. Surface-bound molecules have 'up' states, while reactants or products on the front or back side of a membrane have 'fsoln' or 'bsoln' indicators, respectively. **(D)** Reaction rules for model generation using BioNetGen; other rules are not shown (Color version of this figure is available at *Bioinformatics* online.)

Figure 1 tested the generated species properties, reactant and product states, and simple reaction generation. A model in which A and B species can bind together to form the A-B-B-A complex tested the method's accuracy with symmetric complexes. Finally, two polymer growth and disassembly models validated Smoldyn's on-the-fly network generation and its performance with large reaction networks. In all cases, the different network expansion methods produced the same results. See Andrews (2016) for details.

2.2 Rule-based modeling in Smoldyn with BioNetGen

Rule-based modeling using wildcards is convenient but its dependence on informal notation can lead to unintentional results in complicated models. I addressed this by integrating Smoldyn with BioNetGen, which reads the formal BNGL language (Blinov *et al.*, 2004; Chylek *et al.*, 2014). In a process that is relatively seamless to the user, Smoldyn passes BNGL rules to BioNetGen, BioNetGen

generates and saves the reaction network, and Smoldyn reads the reaction network (see Fig. 1D).

Smoldyn defines the properties of these generated species, and also the reactant and product states in reactions, by first extracting the monomer composition of the complexes from the species names that BioNetGen generates. From these monomer compositions, Smoldyn creates simplified species names. Then, Smoldyn computes the display size, diffusion coefficient, and color for a species using the following equations, which are essentially the same as equations (1–3):

$$r_{\text{complex}} = \left(\sum\nolimits_i r_i^3\right)^{1/3} \tag{4}$$

$$D_{\text{complex}} = \left(\sum\nolimits_i D_i^{-3}\right)^{-1/3} \tag{5}$$

$$v_{\text{complex}} = \frac{\sum_i r_i v_i}{\sum_i r_i} \tag{6}$$

The i subscripts index the monomers that compose the complex. Smoldyn defines surface interaction rates using the same 'greatest action' approach as before, now using the surface interactions of all component monomers. Finally, Smoldyn assigns states to reactants and products based upon any 'monomer states' that the user entered, in each case choosing the highest priority monomer state of the component monomers, where surface-bound states have higher priority than solution states. In the model shown in Figure 1, for example, ligand and receptor monomer states are 'solution' and 'up', respectively (assigned in a portion of the input file that is not shown), so Smoldyn chooses the higher priority 'up' state for the ligand-receptor complex. I validated Smoldyn's BioNetGen rule-based modeling with the same models as for wildcards, described earlier.

These approaches to rule-based modeling are complimentary, with wildcards being better for simple models and BioNetGen for complicated models. They can also be used together, such as by generating a reaction network in BioNetGen, which is less vulnerable to mistakes, and then directing model observation using wildcards, which are more convenient.

Keeping species narrowly defined in both approaches simplified software design, cleanly separating the tasks that address reaction network expansion with those that perform the simulation. It also means that the core simulation algorithms work with simpler molecule data structures, which reduces their workload and speeds up simulations. Finally, provided that there are many more molecules than species, this approach reduces computer memory use because less information gets stored with each molecule. This enables the use of faster memory types, again speeding up simulations (e.g. Andrews *et al.*, 2010).

2.3 Diffusion of surface-bound molecules

Smoldyn has been simulating the diffusion of a surface-bound molecule over a single time step with the *single projection* method. In it, Smoldyn computes a 3D Gaussian displacement for the molecule with root mean square (rms) step length $s = \sqrt{2D\Delta t}$, where D is the diffusion coefficient and Δt is the time step, and projects the displaced molecule to the surface along the surface normal vector at the new molecule position (Andrews *et al.*, 2010). This is exact for planar surfaces due to the independence of diffusion on each axis. However, Smoldyn surfaces can also include regions that are not planar, but spherical or cylindrical, so I investigated the quality of this algorithm for these shapes. More specifically, users define surfaces in Smoldyn by constructing them from 'panels', each of which can be a rectangle, triangle, sphere, cylinder, hemisphere or disk, so these tests investigated diffusion accuracy on each panel shape. I also investigated the accuracy of transitions between panels, described below.

I compared the single projection method with the *double projection* method, in which the molecule is projected into the plane tangent to the panel at the molecule's starting point, creating a 2D diffusive step, and then projected to the panel using the local surface normal (e.g. Hołyst *et al.*, 1999) (Fig. 2A). I implemented both algorithms in Mathematica and ran them over one time step for 10^5 non-interacting molecules that all started at the same points on a sphere or cylinder surface. The molecule rms step lengths were $0.5R$, where R is the sphere or cylinder radius. Comparison of the resulting distributions to the exact ones (from Ghosh *et al.*, 2012) showed that the single projection method captures the distribution more

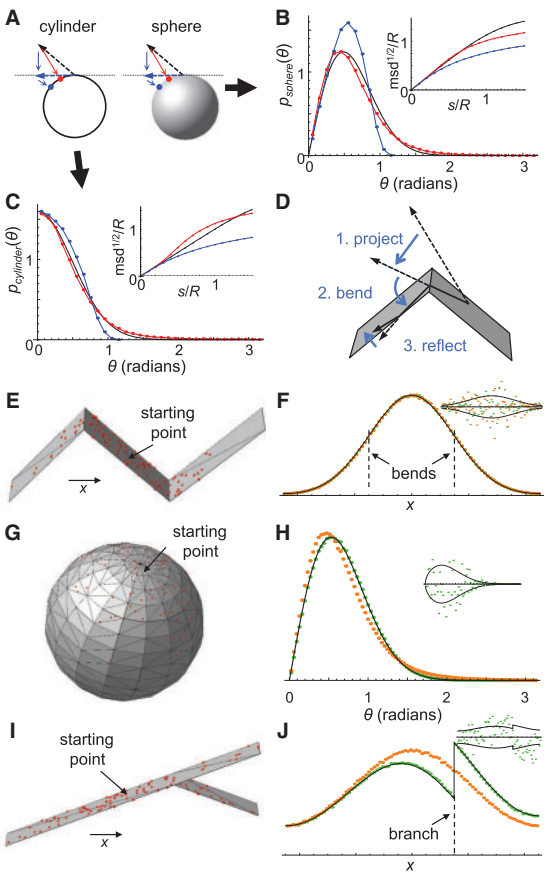


Fig. 2. Surface-bound diffusion. **(A)** Single (red) and double (blue) projection methods. **(B, C)** Molecular distribution resulting from single (red) and double (blue) projection, compared with exact results (black); panel B is for a sphere and C a cylinder. Insets show square root of mean square displacements, measured along the curved surfaces, as functions of the rms step length; both axes are normalized with respect to the radius of curvature. **(D)** Algorithm for simulating diffusion across multiple panels. **(E, F)** Diffusion on a surface with sharp bends. **(G, H)** Diffusion on a sphere, composed from 480 triangles. **(I, J)** Diffusion on a branched surface. In panels F, H and J, the black lines represent exact results, the green points arose from simulations that used 300 time steps that were each 1 ms long, and the orange points arose from simulations that used one 300 ms time step. These simulations used 10^6 non-interacting molecules that started at the marked 'starting points'. Insets to these panels show deviation of simulation results about exact theory; dashed lines represent one standard deviation for the expected statistical deviation that would arise due to the finite number of simulated particles (Color version of this figure is available at *Bioinformatics* online.)

accurately for both shapes (Fig. 2B and C). I also computed the square root of the mean squared displacement measured along the sphere or cylinder surface (Fig. 2B and C insets), which is directly proportional to the simulated diffusion coefficient and thus shows whether simulated diffusion has the correct rate. For the sphere, the simulated diffusion coefficient was within 10% of the exact result for all rms step lengths below the radius of curvature for the single projection method but deviated by up to 32% for the double projection method. For the cylinder, the single projection method was in error by up to 24% while the double projection method was in error by up to 33%. These results show that the single projection method is superior, so I kept the existing Smoldyn algorithm.

Smoldyn also used the single projection method when a molecule diffused beyond its initial panel, projecting it to the adjacent neighboring panel (if there were multiple neighbors, then Smoldyn chose one of them randomly). However, this was inaccurate at bends, did not allow diffusion across multiple panels in one step, and created errors with branched surfaces (where one surface splits into two or more surfaces). Thus, I replaced it with a new design (Fig. 2D). As before, Smoldyn starts by computing a 3D Gaussian displacement for a molecule. Smoldyn projects it to the infinite version of the molecule's current panel, meaning that flat panels are treated as infinite planes, cylindrical panels as infinite cylinders, and hemispherical panels as spheres. If this places the molecule beyond the edge of its current panel, Smoldyn determines which neighbor the trajectory continues onto, choosing one at random if the surface branches there, and bends the trajectory at the panel boundary into the local plane of the new panel. If the trajectory reaches an edge that does not have a neighbor, Smoldyn reflects the trajectory using ballistic reflection. Smoldyn repeats this, bending or reflecting the trajectory as appropriate, until the trajectory is used up.

These algorithms are exact for multiple planar panels that bend along a single axis (Fig. 2E and F) and become exact as simulation time steps are reduced towards zero (green points in Fig. 2F, H and J; more precisely, they become exact as molecule rms step lengths become much smaller than the local radii of curvature). They are inexact for curved panels (Fig. 2B and C), multiple planar panels that curve on two axes (Fig 2G and H), and branched surfaces (Fig. 2I and J). In this last case, the probability of staying on the original panel is too high and of transitioning onto the branches is too low, which is difficult to avoid because the branch affects the diffusion of all nearby molecules, including those that never encountered the branch. Biological membranes can be highly convoluted, such as during endocytosis, typically with curvature on two axes. From above, Smoldyn is inexact in these cases but will be reasonably accurate if rms step lengths are smaller than the radii of curvature. As with other algorithms, this accuracy can be tested by varying the simulation time step and observing if simulation results change (Andrews, 2012). Comparison with Smoldyn's prior algorithms, from Smoldyn version 2.39, showed that diffusion of surface-bound molecules now runs about 3.5 times faster when using the same size time step (using the model shown in Fig. 2E) and that reactions between surface-bound molecules exhibit better accuracy (tested with reactions between molecules bound to a sphere and a bacillus).

2.4 Excluded volume interactions

Smoldyn can simulate molecules as point-like particles that do not occupy volume (Andrews and Bray, 2004). This is fast and typically valid when the molecules occupy a small fraction of the system volume. Smoldyn can also simulate molecules as spheres that do occupy volume to support research on the effects of macromolecular crowding or molecular interactions in confined spaces (e.g. Andrews *et al.*, 2015; Marquez-Lago *et al.*, 2012).

Smoldyn represents excluded volume interactions by simulating diffusion with Gaussian distributed displacements, as usual, and then increasing the separation between any molecules that ended up closer than the sum of their excluded volume radii, which is called the binding radius. Although this repositioning can create additional collisions with yet other molecules, Smoldyn ignores them for computational efficiency and numerical stability. Smoldyn has been assigning new molecule positions using the *overlap method*: it computes the 'overlap' as the difference between the binding radius and the molecular separation and then computes the new separation as the binding radius plus the overlap; the molecules stay on the axis that included the molecule centers (Fig. 3A).

I investigated whether it would be more accurate if molecules were repositioned using the *reflection method*, in which molecules bounce off of each other according to their straight-line trajectories (Fig. 3B). Here, Smoldyn computes the collision time as the time when the molecule surfaces exactly touch, computes the reflection plane as the plane tangent to the molecules' contacting points at the collision time, and reflects the molecule trajectories off of this plane. In contrast to simulations of billiard ball collisions, which are otherwise similar, Smoldyn conserves trajectory lengths rather than momenta because doing so is closer to the physics of diffusing particles, where momentum is meaningless (Purcell, 1977).

Both algorithms are exact in the limit of short time steps because, in this limit, molecule surfaces are effectively flat on the size scale of the rms step length and both algorithms become equivalent to reflection off of a flat surface. Reflection is an exact approach for simulating diffusion near an impermeable planar surface (Edelstein and

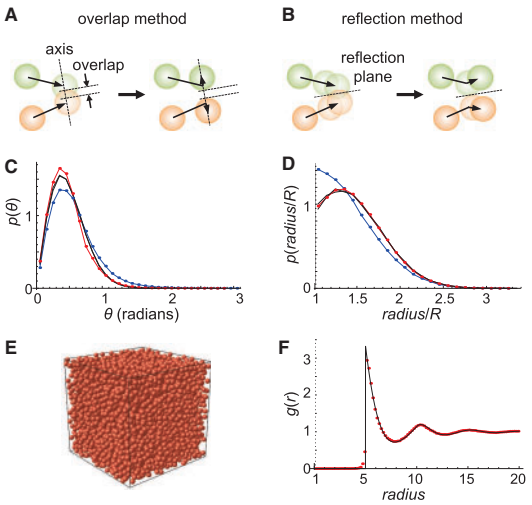


Fig. 3. Excluded volume. **(A)** The overlap method, in which colliding molecules are moved to make the space between them equal to their prior overlap. **(B)** The reflection method, in which molecules reflect off each other. **(C, D)** Angular and radial distributions of particles about a sphere, with essentially exact results in black, the overlap method in blue, and the reflection method in red. **(E)** A hard-sphere fluid simulation; volume is $(100\,\text{nm})^3$ with periodic boundaries, there are 6112 molecules with diameters of 5 nm and diffusion coefficients of 65 $\mu m^2/s$, and time ran for 6 μs in steps of 0.1 ns. Simulation took 11 minutes on a 2013 MacBook Pro (and 1 min when excluded volume reactions were ignored). **(F)** Radial distribution function for simulation in panel E, showing theory from Chang and Sandler (1994) as a black line and simulation data with red dots (Color version of this figure is available at *Bioinformatics* online.)

Agmon, 1993). This exact limit does not depend on the relative molecule radii or diffusion coefficients.

I compared these algorithms by simulating them with Mathematica. Each simulation included a sphere at the origin with radius R to represent one molecule and a particle that started a short distance away from the sphere along the z-axis to represent the other molecule. The particle diffused and interacted with the sphere using one of the two algorithms. Mathematica ran the simulation for one step using an rms step length $0.5R$ or 20 steps with an rms step length of $0.5R/\sqrt{20}$, which simulated the effects of long and short time steps with the same diffusion coefficient. Figure 3B and C show the angular and radial distributions of particles about the sphere for the two algorithms, for 10^5 particles that started $0.1R$ away from the sphere surface. Using short time steps, both algorithms produced essentially the same distributions (black lines), in agreement with their being exact in the limit of short time steps. Comparison of the single step results with the black lines shows that the reflection method produced more accurate angular and radial distributions than the overlap method. I also investigated the first, second, and third moments of the particle distribution along the z-coordinate. Again, the two methods led to essentially identical results with short time steps (mean, standard deviation, and skewness, divided by R, for 20 short steps and particle starting $0.1R$ away from sphere: 1.30, 0.38 and 0.35). With single steps, the reflection method results were much closer to these ideal values (overlap method: 1.17, 0.45 and −0.42; reflection method: 1.30, 0.39 and 0.13). Results were similar for particles that started 20.2 and 0.5R away from the sphere. Together, these results show that the reflection method produces more accurate particle distributions.

As a further test, I used Smoldyn to simulate a hard-sphere fluid, a widely used model system (e.g. Speedy, 1987; Thiele, 1963). I scaled the parameters to make them comparable to hemoglobin in a red blood cell, using a 5 nm molecular diameter and a 65 $\mu m^2/s$ diffusion coefficient in dilute solution (see Andrews, 2012; Krueger and Nossal, 1988; Muramatsu and Minton, 1988). Hemoglobin occupy about 25% of the cell cytoplasm volume (Krueger and Nossal, 1988), but I assumed 40% occupancy to make the test more demanding. Figure 3E shows a snapshot of the simulation and Figure 3F shows the radial distribution function of the proteins, which represents the distribution of separations between protein pairs. The simulated data agree essentially perfectly with the theoretical result for hard spheres (from Chang and Sandler, 1994). The simulation also showed that crowding decreased the hemoglobin diffusion coefficient about 25%, to 49.8 $\mu m^2/s$. Yet more tests showed that Smoldyn also performed well when molecules had different radii and diffusion coefficients (available in the S94_archive/Andrews_2016 Smoldyn example directory).

These excluded volume algorithms are independent of Smoldyn's treatment of molecule-surface interactions, in which Smoldyn only considers molecule center positions and ignores molecule radii. Moving the surface towards the molecules by one molecule radius can effectively account for excluded volume, if desired (e.g. Andrews *et al.*, 2015). The algorithms are also independent of Smoldyn's treatment of chemical reactions, meaning that Smoldyn can account for both excluded volume and chemical reactions for a pair of molecules, but it ignores interactions between these processes. Thus, it is possible to run reaction-diffusion simulations in which all molecules have excluded volumes, but the simulated reaction rate constants may not equal the user's requested values.

This approach to simulating excluded volume using hard spheres maintains the design philosophy used throughout the software in which Smoldyn simulates a well-defined model system and becomes exact in the limit of short time steps. On the other hand, if models require more realistic representations at the single-molecule size scale, then software tools that represent excluded volume interactions using softer potentials and/or non-spherical molecules, including ReaDDy (Schöneberg and Noé, 2013) and SpringSaLaD (Michalski and Loew, 2016) may be more appropriate (however, SpringSaLaD's excluded volume algorithm ignores diffusion that occurred after molecular collision, making it less accurate than the ones described here).

2.5 Single-particle tracking

Single-particle tracking studies, in which one observes the motion of individual proteins or other particles (see Saxton, 2009; Hoze *et al.*, 2012), provide more information than ensemble measurements because they provide information about the distributions of behaviors. They can also show when and where particles interact with other structures or particles.

Smoldyn has not been ideal for modeling single particle tracking because its molecules get replaced with new ones when they undergo reactions, making it difficult to track a single molecule through its bindings and unbindings. Using the fact that every simulated molecule in Smoldyn has a unique serial number, I addressed this problem by adding several options for reading and assigning serial numbers. A modeler can now (i) specify that a reaction product should have the same serial number as one of the reactants, (ii) log the reactions and track the positions of molecules with specific serial numbers and (iii) print out serial numbers along with other molecule information. These enable collections of serial numbers to be tracked throughout a simulation much as fluorescent labels can be tracked with microscopy.

2.6 Libsmoldyn API

Nearly all cell biology behaviors arise from elementary processes working together. These processes include molecular diffusion, chemical reactions, and protein-membrane interactions, all of which Smoldyn can simulate. They also include filament dynamics, bulk flow, and dynamics of extended complexes, which Smoldyn cannot simulate currently. Yet undiscovered processes may be important, too. Because these processes need to work together, Smoldyn is inadequate for modeling many biological systems.

I developed the Libsmoldyn API to alleviate this problem. It is a C/C++ language interface to the Smoldyn source code which enables researchers to link Smoldyn to other software. With Libsmoldyn, it is relatively straightforward to simulate a model by using Smoldyn for the parts of the model that it can handle and to write custom code (or use other 'solvers') for the other parts.

Libsmoldyn also helps with multi-scale simulators, where the Smoldyn code simulates system portions that require single-molecule resolution and other code simulates other portions. The MOOSE (Multiscale Object-Oriented Simulation Environment) neural simulator (Ray *et al.*, 2008), for example, simulates dynamics within neural spines using Libsmoldyn and signaling across multiple cells using ordinary differential equations. Also, Virtual Cell (Cowan *et al.*, 2012) runs particle-based simulations using links to the Smoldyn code, although not through Libsmoldyn.

Libsmoldyn includes about 100 functions for setting and getting Smoldyn parameters. For example, one function instructs Smoldyn to read a text input file and prepare the data structures. Other functions can create a Smoldyn model purely through function calls, without a text input file. Yet others can be called to run the simulation for one or more time steps. The API includes a header file that

defines the Smoldyn data structures, enabling linked code to read any aspects of the Smoldyn simulation, including the simulation state (linked code can also write to the data structures, although this is not advised).

Linking to Smoldyn through the Libsmoldyn API is generally preferable to working directly with the Smoldyn source code. It is more convenient because the API has functions that are specifically designed to handle most interactions that are likely to be necessary. Also, it is safer because all API functions that manipulate the model address the required modifications in all of the data structures at once, thereby reducing the likelihood of creating internal inconsistencies. In addition, the API functions check for errors in the input parameters, returning errors if they would cause problems. Finally, I plan to maintain the current API so that software that links to it will continue to work, even as others and I continue to extend Smoldyn's internal code.

3 Conclusions

Smoldyn has proven to be a useful simulation tool for numerous research projects. In the course of many of them, researchers have needed software additions or identified algorithms that required improvement. This paper reports on those additions and improvements. It describes two types of rule-based modeling, improved methods for simulating the diffusion of surface-bound molecules and excluded volume interactions, support for single-molecule tracking, and a library interface. These additions work well with the other software components and with each other. The focus has been on the Smoldyn software in particular, but these developments should have broader relevance because each involves new approaches.

These extensions have not reduced Smoldyn's functionality for models that it ran previously. It still runs input files that were written for older versions and gives either the same or more accurate results. In 2010, a model of a Michaelis-Menten reaction comprising 10 000 molecules, which ran for 10 s of simulated time in 1 ms time steps, required 47 s to run (Andrews *et al.*, 2010). On a 2013 Mac Pro laptop, the same model using the essentially same software ran in 21 s, because of the faster computer. The latest Smoldyn version (2.45) also ran this model in 21 s, showing that the new features have not slowed the software.

Acknowledgements

I thank Upi Bhalla, Weiren Cui, Jim Faeder, André Leier, Karen Lipkow, François Nédélec, Masoud Nickaeen, Martin Robinson, Karin Sasaki, Hugo Schmidt, Boris Slepchenko and Gerard Weatherby for useful discussions. Much of this work was performed at the Isaac Newton Institute for Mathematical Sciences during the program *Stochastic Dynamical Systems in Biology: Numerical Methods and Applications*. I thank Radek Erban, David Holcman, Sam Isaacson, and Konstantinos Zygalakis, who were the program organizers, and the Institute staff. I thank Roger Brent, Erick Matsen and Harlan Robbins for providing space at the FHCRC.

Funding

This work was supported by a Simons Foundation grant awarded to S.S.A. and by EPSRC grant EP/K032208/1 awarded to the Isaac Newton Institute.

Conflict of Interest: none declared.

References

Andrews,S.S. (2009) Accurate particle-based simulation of adsorption, desorption and partial transmission. *Phys. Biol.*, **6**, 046015.

Andrews,S.S. (2012). Spatial and stochastic cellular modeling with the Smoldyn simulator. In: van Helden, *et al.* (eds) *Bacterial Molecular Networks: Methods and Protocols. Methods for Molecular Biology*, **804**, 519–542.

Andrews,S.S. (2016) Rule-based modeling using wildcards. *Methods Mol. Biol.*, submitted.

Andrews,S.S., and Bray,D. (2004) Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Phys. Biol.*, **1**, 137.

Andrews,S.S. *et al.* (2009). Stochastic models of biological processes. In: Meyers, R. (ed). *Encyclopedia of Complexity and Systems Science*, 8730–8749. Springer, New York.

Andrews,S.S. *et al.* (2010) Detailed simulations of cell biology with Smoldyn 2.1. *PLoS Comput. Biol.*, **6**, e1000705.

Andrews,S.S. *et al.* (2015). Simulating macromolecular crowding with particle and lattice-based methods (Team 3). In: Gilbert, D., Heiner, M., Takahashi, K. and U. A. M. (eds). *Multiscale Spatial Computational Systems Biology*, 170–187.

Arjunan,S.N.V., and Tomita,M. (2010) A new multicompartmental reaction-diffusion modeling method links transient membrane attachment of *E. coli* MinE to E-ring formation. *Syst. Synth. Biol.*, **4**, 35–53.

Bartol,T.M. *et al.* (2015) Computational reconstitution of spine calcium transients from individual proteins. *Front. Synaptic Neurosci.*, **7**,

Blinov,M.L. *et al.* (2004) BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, **20**, 3289–3291.

Blinov,M.L. *et al.* (2005) 'On-the-fly' or 'generate-first' modeling?. *Nat. Biotechnol.*, **23**, 1344–1345.

Bray,D., and Lay,S. (1997) Computer-based analysis of the binding steps in protein complex formation. *Proc. Natl. Acad. Sci. USA*, **94**, 13493–13498.

Chang,J., and Sandler,S.I. (1994) A real function representation for the structure of the hard-sphere fluid. *Mol. Phys.*, **81**, 735–744.

Chylek,L.A. *et al.* (2014) Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdiscip. Rev. Syst. Biol. Med.*, **6**, 13–36.

Cowan,A.E. *et al.* (2012) Spatial modeling of cell signaling networks. *Methods Cell Biol.*, **110**, 195.

Danos,V., and Laneve,C. (2004) Formal molecular biology. *Theor. Comput. Sci.*, **325**, 69–110.

Dix,J.A., and Verkman,A. (2008) Crowding effects on diffusion in solutions and cells. *Annu. Rev. Biophys.*, **37**, 247–263.

Drawert,B. *et al.* (2012) URDME: a modular framework for stochastic simulation of reaction-transport processes in complex geometries. *BMC Syst. Biol.*, **6**, 1.

Edelstein,A.L., and Agmon,N. (1993) Brownian dynamics simulations of reversible reactions in one dimension. *J. Chem. Phys.*, **99**, 5396–5404.

ElKalaawy,N., and Amr,W. (2015) Methodologies for the modeling and simulation of biochemical networks, illustrated for signal transduction pathways: a primer. *Biosystems*, **129**, 1–18.

Ghosh,A. *et al.* (2012) A "Gaussian" for diffusion on the sphere. *Europhys. Lett.*, **98**, 30003.

Gillespie,D.T. (1977) Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, **81**, 2340–2361.

Goldman,J. *et al.* (2004) Size and composition of membrane protein clusters predicted by monte carlo analysis. *Eur. Biophys. J.*, **33**, 506–512.

Gruenert,G. *et al.* (2010) Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics*, **11**, 307.

Hattne,J. *et al.* (2005) Stochastic reaction-diffusion simulation with MesoRD. *Bioinformatics*, **21**,

Hołyst,R. *et al.* (1999) Diffusion on curved, periodic surfaces. *Phys. Rev. E*, **60**, 302.

Hoze,N. *et al.* (2012) Heterogeneity of AMPA receptor trafficking and molecular interactions revealed by superresolution analysis of live cell imaging. *Proc. Natl. Acad. Sci. USA*, **109**, 17052–17057.

Jilkine,A. *et al.* (2011) A density-dependent switch drives stochastic clustering and polarization of signaling molecules. *PLoS Comp. Biol.*, **7**, e1002271.

Johnson,M.E., and Hummer,G. (2014) Free-propagator reweighting integrator for single-particle dynamics in reaction-diffusion models of heterogeneous protein-protein interaction systems. *Phys. Rev. X*, **4**, 031037.

Khan,S. *et al.* (2012) Spatiotemporal maps of CaMKII in dendritic spines. *J. Comput. Neurosci.*, **33**, 123–139.

Kinney,J.P. *et al.* (2013) Extracellular sheets and tunnels modulate glutamate diffusion in hippocampal neuropil. *J. Comp. Neurol.*, **521**, 448–464.

Krueger,S., and Nossal,R. (1988) SANS studies of interacting hemoglobin in intact erythrocytes. *Biophys. J.*, **53**, 97.

Lok,L., and Brent,R. (2005) Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nat. Biotechnol.*, **23**, 131–136.

Marquez-Lago,T. *et al.* (2012) Anomalous diffusion and multifractional brownian motion: simulating molecular crowding and physical obstacles in systems biology. *Syst. Biol., IET*, **6**, 134–142.

Mayer,B.J. *et al.* (2009) Molecular machines or pleiomorphic ensembles: signaling complexes revisited. *J. Biol.*, **8**, 1.

Michalski,P.J., and Loew,L.M. (2016) SpringSaLaD: a spatial, particle-based biochemical simulation platform with excluded volume. *Biophys. J.*, **110**, 523–529.

Muramatsu,N., and Minton,A.P. (1988) Tracer diffusion of globular proteins in concentrated protein solutions. *Proc. Natl. Acad. Sci. USA*, **85**, 2984–2988.

Purcell,E.M. (1977) Life at low Reynolds number. *Am. J. Phys.*, **45**, 3–11.

Ray,S. *et al.* (2008) A general biological simulator: the multiscale object oriented simulation environment, MOOSE. *BMC Neurosci.*, **9(Suppl 1)**, P93.

Robinson,M. *et al.* (2015) Multiscale reaction-diffusion simulations with Smoldyn. *Bioinformatics*, **31**, 2406–2408.

Saxton,M.J. (1987) Lateral diffusion in an archipelago. the effect of mobile obstacles. *Biophys. J.*, **52**, 989–997.

Saxton,M.J. (2009). Single particle tracking. In: Jue, T. (ed). *Fundamental Concepts in Biophysics*: *Volume 1*. Handbook of Modern Biophysics, Humana Press, New York, pp. 147–179.

Schaff,J. *et al.* (1997) A general computational framework for modeling cellular structure and function. *Biophys. J.*, **73**, 1135.

Schmidt,H.G. *et al.* (2014) An integrated model of transcription factor diffusion shows the importance of intersegmental transfer and quaternary protein structure for target site finding. *PLoS One*, **9**, e108575.

Schöneberg,J., and Noé,F. (2013) ReaDDy-a software for particle-based reaction-diffusion dynamics in crowded cellular environments. *PloS One*, **8**, e74261.

Schöneberg,J. *et al.* (2014) Simulation tools for particle-based reaction-diffusion dynamics in continuous space. *BMC Biophys.*, **7**, 1.

Sneddon,M.W. *et al.* (2011) Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nat. Methods*, **8**, 177–183.

Speedy,R.J. (1987) Diffusion in the hard sphere fluid. *Mol. Phys.*, **62**, 509–515.

Stefan,M.I. *et al.* (2014) Multi-state modeling of biomolecules. *PLOS Comput. Biol.*, **10**, e1003844.

Stiles,J.R., and Bartol,T.M. (2001). Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. In: De Schutter, E. (ed). *Computational Neuroscience, Realistic Modeling for Experimentalists*, chapter 4. CRC Press, Boca Raton, FL, 87–127.

Stiles,J.R. *et al.* (1996) Miniature endplate current rise times less than 100 microseconds from improved dual recordings can be modeled with passive acetylcholine diffusion from a synaptic vesicle. *Proc. Natl. Acad. Sci. USA*, **93**, 5747–5752.

Strongin,D.E. *et al.* (2014) Nucleolar tethering mediates pairing between the *IgH* and *Myc* loci. *Nucleus*, **5**, 474–481.

Stundzia,A.B., and Lumsden,C.J. (1996) Stochastic simulation of coupled reaction–diffusion processes. *J. Comput. Phys.*, **127**, 196–207.

Subburaj,Y. *et al.* (2015) Bax monomers form dimer units in the membrane that further self-assemble into multiple oligomeric species. *Nat. Commun.*, **6**,

Takahashi,K. *et al.* (2005) Space in systems biology of signaling pathways–towards intracellular molecular crowding in silico. *FEBS Lett.*, **579**, 1783–1788.

Takahashi,K. *et al.* (2010) Spatio-temporal correlations can drastically change the response of a MAPK pathway. *Proc. Natl. Acad. Sci. USA*, **107**, 2473–2478.

Thiele,E. (1963) Equation of state for hard spheres. *J. Chem. Phys.*, **39**, 474–479.

Tolle,D.P., and Le Novère,N. (2010) Meredys, a multi-compartment reaction-diffusion simulator using multistate realistic molecular complexes. *BMC Syst. Biol.*, **4**, 24.

van Zon,J.S., and Ten Wolde,P.R. (2005) Simulating biochemical networks at the particle level and in time and space: Green's function reaction dynamics. *Phys. Rev. Lett.*, **94**, 128103.

Zavala,E., and Marquez-Lago,T.T. (2014) The long and viscous road: uncovering nuclear diffusion barriers in closed mitosis. *PLoS Comput. Biol.*, **10**, e1003725.