Section IV

MODELLING OF BIOLOGICAL SYSTEMS

Chapter 14

Spatiotemporal Modeling and Simulation in Biology

Ivo F. Sbalzarini

1. Introduction

Describing the dynamics of processes in both space and time simultaneously is referred to as spatiotemporal modeling. This is in contrast to describing the dynamics of a system in time only as is, for example, usually done in chemical kinetics or pathway models. Solving spatiotemporal models in a computer requires spatiotemporal computer simulations. While computational data analysis allows unbiased and reproducible processing of large amounts of data from, e.g. high-throughput assays, computer simulations enable virtual experiments *in silico* that would not be possible in reality. This greatly expands the range of possible perturbations and observations. Computational experiments allow studying systems whose complexity prohibits manual analysis, and they make accessible time and length scales that cannot be reached by lab experiments. Examples of the latter include molecular dynamics (MD) studies in structural biology and studies in ecology or evolutionary biology. In virtual experiments, all variables are controllable and observable. We can thus measure everything and precisely control all influences and cross-couplings. This allows disentangling coupled effects that could not be separated in real experiments, greatly reduces or eliminates the need for indirect control experiments, and facilitates interpretation of the results. Finally, computational models do not involve living beings, thus enabling

experiments that would not be possible in reality due to ethical reasons. Although we focus on applications of spatiotemporal computer simulations in biology, the employed concepts and methods are more generally valid.

Resolving a dynamic process in space greatly increases the number of degrees of freedom (variables) that need to be tracked. Consider, for example, a biochemical heterodimerization reaction. This reaction can be modeled by its chemical kinetics using three variables: the concentrations of the two monomers and the concentration of dimers. Assume now that monomers are produced at certain locations in space and freely diffuse from there. Their concentration thus varies in space in such a way that it is higher close to the source and lower farther away, which greatly increases the number of variables we have to track in the simulation. If we are, say, interested in the local concentrations at 1000 positions, we already have to keep track of 3000 variables. Moreover, the reactions taking place at different points in space are not independent. Each local reaction can influence the others through diffusive transport of monomers and dimers. The complexity of spatiotemporal models thus rapidly increases. In fact, there is no theoretical limit to the number of points in space that we may use to resolve the spatial patterns in the concentration fields. Using infinitely many points corresponds to modeling the system as a continuum.

A number of powerful mathematical tools are available to efficiently deal with spatiotemporal models and to simulate them. While it is not possible within the scope of this chapter to describe each of them in detail, we will give an overview with references to specialized literature. We then review in detail one particular method that is well suited for applications in biology. But before we start, we revisit some of the motivations and particularities of spatiotemporal modeling in the life sciences.

In spatiotemporal modeling, nature is mostly described in four dimensions: time plus three spatial dimensions. While time and the presence of reservoirs (integrators) are essential for the existence of dynamics, three-dimensional (3D) spatial aspects also play important roles in many biological processes. Think, for example, of predators hunting their prey in a forest, of blood flowing through our arteries, of the electromagnetic fields in the brain, or of such an unpleasant phenomenon as the

epidemic spread of a disease. In all of these examples, and many others, the spatial distributions of some quantities play an essential role. Models and simulations of such systems should thus account for and resolve these distributions. When determining the location of an epileptic site in the brain, it is, for instance, of little value to know the total electric current density in the whole brain — we need to know precisely where the source is. These examples extend across all scales of biological systems, from the above-mentioned predator–prey interactions in ecosystems over morphogenesis[1–5] and intracellular processes to single molecules. Think, for example, of conformational changes in proteins. Examples at the intracellular level include virus entry[6,7] and transport,[8–10] intracellular signaling,[11,12] the diffusion of proteins in the various cellular compartments,[13–15] or the fact that such compartments exist in the first place.

Spatial organization is important, as the same protein can have different effects depending on the intracellular compartment in which it is located. The most prominent example is probably cytochrome C, which is an essential part of the cell respiration chain in the mitochondria, but triggers programmed cell death (apoptosis) when released into the cytoplasm.[16] Another example is found in the role of transmembrane signaling during morphogenesis. Differences in protein diffusion constants are not large enough to produce Turing patterns,[1] and the slow transport across intercompartment membranes is essential.[17] Examples of spatiotemporal processes at the multi-cellular level include tumor growth[18–20] and cell-cell signaling,[21] including phenomena such as bacterial quorum sensing, the microscopic mechanism underlying the macroscopic phenomenon of bioluminescence in certain squid.[22]

Given the widespread importance of spatiotemporal processes, it is not surprising that a number of large software projects for spatiotemporal simulations in biology have been initiated. Examples in computational cell biology[23,24] include E-Cell, MCell, and the Virtual Cell.

2. Properties of Biological Systems

Simulating spatially resolved processes in biological systems, such as geographically structured populations, multicellular organs, or cell organelles, provides a unique set of challenges to any mathematical

method. One often hears that this is because biological systems are "complex".

Biochemical networks, ecosystems, biological waves, heart cell synchronization, and life in general are located in the high-dimensional, nonlinear regime of the map of dynamical systems,[25] together with quantum field theory, nonlinear optics, and turbulent flows. None of these topics are completely explored. They are at the limit of our current understanding and will remain challenging for many years to come. Why is this so, and what do we mean by "complex"?

Biological systems exhibit a number of characteristics that render them difficult. These properties frequently include one or several of the following:

- high-dimensional (or infinite-dimensional in the continuum limit);
- regulated;
- delineated by complex shapes;
- nonlinear;
- coupled across scales and subsystems;
- plastic over time (time-varying dynamics); and/or
- nonequilibrium.

Due to these properties, biological systems challenge existing methods in modeling and simulation. They are thus particularly well suited to drive the development of new methods and theories. The challenges presented by spatiotemporal biological systems have to be addressed on several fronts simultaneously: numerical simulation methods, computational algorithms, and software engineering.[26] Numerical methods are needed that can deal with multi-scale systems[27–31] and topological changes in complex geometries. Computer algorithms have to be efficient enough to deal with the vast number of degrees of freedom, and software platforms must be available to effectively and robustly implement these algorithms on multiprocessor computers.[32]

2.1. *Dimensionality and Degrees of Freedom*

The large number of dimensions (degrees of freedom) is due to the fact that biological systems typically contain more compartments, components, and

interaction modes than traditional engineering applications such as electronic circuits or fluid mechanics.[26] In a direct numerical simulation, all degrees of freedom need to be explicitly tracked. In continuous systems, each point in space adds additional degrees of freedom, leading to an infinite number of dimensions. Such systems have to be discretized, i.e. the number of degrees of freedom needs to be reduced to a computationally feasible amount, which is done by selecting certain representative dimensions. Only these are then tracked in the simulation, approximating the behavior of the full, infinite-dimensional system. Discretizations must be consistent, i.e. the discretized system has to converge to the full system if the number of explicitly tracked degrees of freedom goes to infinity.

Discrete biological systems already have a finite number of degrees of freedom and can sometimes be simulated directly. If the number of degrees of freedom is too large, as is e.g. the case when tracking the motion of all atoms in a protein, we do, however, again have to reduce them in order for simulations to be feasible. This can be done by collecting several degrees of freedom into one and only tracking their collective behavior. These so-called "coarse graining" methods greatly reduce the computational cost and allow simulations of very large, high-dimensional systems such as patches of lipid bilayers with embedded proteins,[33,34] or actin filaments.[35] Coarse graining thus allows extending the capabilities of molecular simulations to time and length scales of biological interest.

2.2. *Regulation*

In biological systems, little is left to chance, which might seem surprising given the inherently stochastic nature of molecular processes, environmental influences, and phenotypic variability. These underlying fluctuations are, however, in many cases a prerequisite for adaptive deterministic behavior, as has been shown, for example, in gene regulation networks.[36] In addition to such indirect regulation mediated by bistability and stochastic fluctuations, feedback and feed-forward loops are ubiquitous in biological systems. From signal transduction pathways in single cells to Darwinian evolution, regulatory mechanisms play important roles. Results from control theory tell us that such loops can alter the dynamic behavior of a system, change its stability or robustness, or give rise to

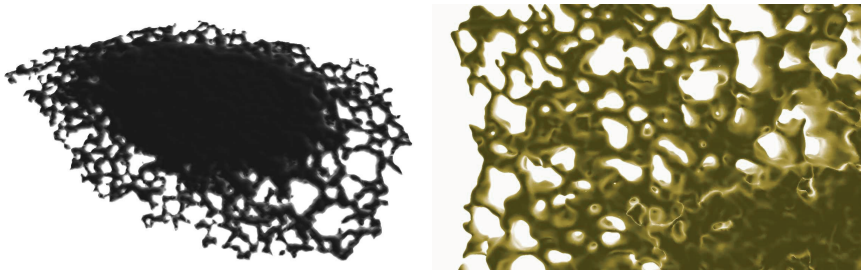
multi-stable behavior that enables adaptation to external changes and disturbances.[36] Taking all of these effects into account presents a grand challenge to simulation models not only because many of the hypothetical regulatory mechanisms are still unknown or poorly characterized.

2.3. *Geometric Complexity*

Biological systems are mostly characterized by irregular and often moving or deforming geometries. Processes on curved surfaces may be coupled to processes in enclosed spaces; and surfaces frequently change their topology, such as in fusion or fission of intracellular compartments. Examples of such complex geometries are found on all length scales and include the prefractal structures of taxonomic and phylogenetic trees,[37] regions of stable population growth in ecosystems,[38] pneumonal and arterial trees,[39] the shapes of neurons,[40] the cytoplasmic space,[41] clusters of intracellular vesicles,[42] electric currents through ion channels in cell membranes,[43] protein chain conformations,[44] and protein structures.[45] Complex geometries are not only difficult to resolve and represent in the computer, but the boundary conditions imposed by them on dynamic spatiotemporal processes may also qualitatively alter the macroscopically observed dynamics. Diffusion in complex-shaped compartments such as the endoplasmic reticulum (ER; Fig. 1) may appear anomalous, even if the underlying molecular diffusion is normal.[46–49]

2.4. *Nonlinearity*

Common biological phenomena such as interference, cooperation, and competition lead to nonlinear dynamic behavior. Many processes, from repressor interactions in gene networks over predator–prey interactions in ecosystems to calcium waves in cells, are not appropriately described by linear systems theory as predominantly used and taught in physics and engineering. Depending on the number of degrees of freedom, nonlinear systems exhibit phenomena not observed in linear systems. These phenomena include bifurcations, nonlinear oscillations, and chaos and fractals. Nonlinear models are intrinsically hard to solve. Most of them are impossible to solve analytically; and computer simulations are hampered



(a) (b)

Fig. 1. (a) Shaded view of a 3D computer reconstruction of the geometry of an endoplasmic reticulum (ER) of a live cell.[49] (b) Close-up of a reconstructed ER, illustrating the geometric complexity of this intracellular structure.

by the fact that common computational methods, such as normal mode analysis, Fourier transforms, or the superposition principle, break down in nonlinear systems because a nonlinear system is not equal to the sum of its parts.[25]

2.5. *Coupling Across Scales*

Coupling across scales means that events on the microscopic scale such as changes in molecular conformation can have significant effects on the global, macroscopic behavior of the system. This is certainly the case for many biological systems — bioluminescence due to bacterial quorum sensing[22] for example, or the effect on the behavior of a whole organism when hormones bind to their receptors. Such multi-scale systems share the property that the individual scales cannot be separated and treated independently. There is a continuous spectrum of scales with coupled interactions that impose stringent limits on the use of computer simulations. Direct numerical simulation of the complete system would require resolving it in all detail everywhere. Applied to the simulation of a living cell, this would mean resolving the dynamics of all atoms in the cell. A cell consists of about 10^{15} atoms, and biologically relevant processes such as protein folding and enzymatic reactions occur on the time scale of milliseconds. The largest molecular dynamics (MD)[50]

simulations currently done consider about 10^{10} atoms over one nanosecond. In order to model a complete cell, we would need a simulation about 100 000 times larger, running over a millionfold longer time interval. This would result in a simulation at least 10^{11} times bigger than what can currently be done. This is certainly not feasible and will remain so for many years to come. Even if one could simulate the whole system at full resolution, the results would be of questionable value. The amount of data generated by such a simulation would be vast, and the interesting macroscopic phenomena that we are looking for would mostly be masked by noise from the small scales. In order to treat coupled systems, we thus have to use multi-scale models[28–31] and formulations at the appropriate level of detail.

2.6. *Temporal Plasticity*

While the analysis of high-dimensional, nonlinear systems is already complicated as such, the systems themselves also frequently change over time in biological applications. In a mathematical model, this is reflected by jumps in the dynamic equations or by coefficients and functions that change over time. During its dynamics, the system can change its behavior or switch to a different mode. For example, the dynamics of many processes in cells depend on the cell cycle, physiological processes in organisms alter their dynamics depending on age or disease, and environmental changes affect the dynamic behavior of ecosystems. Such systems are called "plastic" or "time-varying". Dealing with time-varying systems, or equations that change their structure over time, is an open issue in numerical simulations. Consistency of the solution at the switching points must be ensured in order to prevent the simulation method from becoming unstable.

2.7. *Nonequilibrium*

According to the second law of thermodynamics, entropy can only increase. Life evades this decay by feeding on negative entropy.[51] The discrepancy between life and the fundamental laws of thermodynamics has puzzled scientists for a long time. It can only be explained by assuming that living systems are not in equilibrium. Most of statistical physics and

thermodynamics has been developed for equilibrium situations and, hence, does not readily apply to living systems. Phenomena such as the establishment of cell polarity or the organization of the cell membrane can only be explained when accounting for nonequilibrium processes such as vesicular recycling.[52] Due to our incomplete knowledge of the theoretical foundations of nonequilibrium processes, they are much harder to understand. Transient computer simulations are often the sole method available for their study.

3. Spatiotemporal Modeling Techniques

Dynamic spatiotemporal systems can be described in various ways, depending on the required level of detail and fidelity. We distinguish three dimensions of description: phenomenological vs. physical, discrete vs. continuous, and deterministic vs. stochastic. The three axes are independent and all combinations are possible. Depending on the chosen system description, different modeling techniques are available. Figure 2 gives an overview of the most frequently used ones as well as examples of dynamic systems that could be described with them.

3.1. *Phenomenological vs. Physical Models*

Phenomenological models reproduce or approximate the overall behavior of a system without resolving the underlying mechanisms. Such models are useful if one is interested in analyzing the reaction of the system to a known perturbation, without requiring information about how this reaction is brought about. This is in contrast to physical models, which faithfully reproduce the mechanistic functioning of the system. Physical models thus allow predicting the system behavior in new, unseen situations, and they give information about how things work. Physical models are based on first principles or laws from physics.

3.2. *Discrete vs. Continuous Models*

The discrete vs. continuous duality relates to the spatial resolution of the model. In a discrete model, each constituent of the system is explicitly

	continuous	discrete
deterministic	PDEs	interacting particles
deterministic	diffusion	molecular dynamics
stochastic	SDEs	random events
stochastic	reaction-diffusion with low molecule numbers	population dynamics

Fig. 2. Most common modeling techniques for all combinations of continuous/discrete and deterministic/stochastic models. The techniques for physical and phenomenological models are identical, but in the former case the models are based on physical principles. Common examples of application of each technique are given in the shaded areas.

accounted for as an individual entity. Examples include MD simulations,[50] where the position and velocity of each atom are explicitly tracked and atoms are treated as individual, discrete entities. In a continuous model, a mean field average is followed in space and time. Examples of such field quantities are concentration, temperature, and charge density.

In continuous models, we distinguish two types of quantities. On the one hand, quantities whose value in a homogeneous system does not depend on the averaging volume are called "intensive". Examples include concentration or temperature. If 1 L of water at 20°C is divided into two half-liter glasses, the water in each of the two glasses will still have a temperature of 20°C, even though the volume is halved. The temperature of the water is independent of the volume of water, hence making temperature an intensive property. On the other hand, quantities whose value in a homogeneous system depends on the volume are called "extensive". These are quantities such as mass, heat, or charge. Neither of the two half-liters of water has the same mass as the original liter. Intensive and

extensive quantities come in pairs: concentration–mass, temperature–heat, charge density–charge, etc. Field quantities as considered in continuous models are always intensive, and quantities in discrete models are usually extensive. Corresponding extensive and intensive quantities are interrelated through an averaging operation. The concentration of molecules can e.g. be determined by measuring the total mass of all molecules within a given volume and dividing this mass by the volume. We imagine such an averaging volume around each point in space in order to recover a spatially resolved concentration field. If the averaging volume chosen is too small, entry and exit of individual molecules will lead to significant jumps in the average. With a growing averaging volume, the concentration may converge to a stable value. If the volume is further enlarged, the concentration may again start to vary due to macroscopic spatial gradients. This behavior is illustrated in Fig. 3. Above the continuum limit λ, the average is converged and microscopic single-particle effects are no longer significant. The value of the continuum limit is governed by the abundance of particles compared to the size of the averaging volume. If the microscopic particles are molecules such as proteins, λ is related to their mean free path. On length scales larger than the scale of field variations L, macroscopic gradients of the averaged field become apparent if the field is not homogeneous, i.e. if its value varies in space. The dimensionless ratio $\mathrm{Kn} = \lambda/L$ is called the Knudsen number.

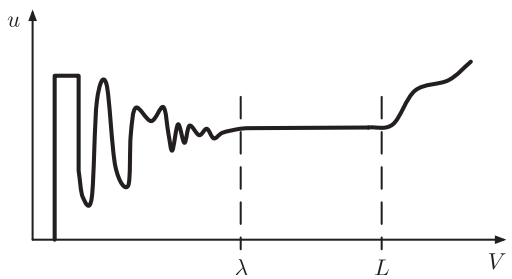


Fig. 3. The value u of a volume-averaged intensive field quantity depends on the size of the averaging volume V. For volumes smaller than the continuum limit λ, individual particles cause the average to fluctuate. In the continuum region above λ, the volume average can be stationary or vary smoothly due to macroscopic field gradients above the length scale of field variations L.

Continuous models are valid only if the microscopic and macroscopic scales are well separated, i.e. if Kn $\ll$ 1; for any spatial distribution with Kn $\gg$ 1, discrete models are the only choice since each particle is important and no continuum region exists. Between these two cases lies the realm of mesoscopic models.[53]

Continuous deterministic models are characterized by smoothly varying (on length scales $>L$) field quantities whose temporal and spatial evolution depends on some derivatives of the same or other field quantities. The fields can, for example, model concentrations, temperatures, or velocities. Such models are naturally formulated as unsteady partial differential equations (PDEs),[54,55] since derivatives relate to the existence of integrators, and hence reservoirs, in the system. The most prominent examples of continuous deterministic models in biological systems include diffusion models, advection, flow, and waves. Discrete deterministic models are characterized by discrete entities interacting over space and time according to deterministic rules. The interacting entities can, e.g. model cells in a tissue,[5] individuals in an ecosystem, or atoms in a molecule.[50] Such models can mostly be interpreted as interacting particle systems or automata. In biology, discrete deterministic models can be found in ecology or in structural biology.

3.3. *Stochastic vs. Deterministic Models*

Biological systems frequently include a certain level of randomness, as is the case for unpredictable environmental influences, fluctuations in molecule numbers upon cell division, and noise in gene expression levels. Such phenomena can be accounted for in stochastic models. In such models, the model output is not entirely predetermined by the present state of the model and its inputs, but it also depends on random fluctuations. These fluctuations are usually modeled as random numbers of a given statistical distribution. Continuous stochastic models are characterized by smoothly varying fields whose evolution in space and time depends on probability densities that are functions of some derivatives of the fields. In the simplest case, this amounts to a single noise term modeling, e.g. Gaussian or uniform fluctuations in the dynamics. Models of this kind are mostly formalized as stochastic differential equations (SDEs).[56]

These are PDEs with stochastic terms that can be used to model probabilistic processes such as the spread of epidemics, neuronal signal transduction,[57,58] or evolution theory. In discrete stochastic models, probabilistic effects mostly pertain to discrete random events. These events are characterized by their probability density functions. Examples include population dynamics (individuals have certain probabilities to be born, die, eat, or be eaten), random walks of diffusing molecules, or stochastically occurring chemical reactions. Several methods are also available for combining stochastic and deterministic models into hybrid stochastic-deterministic models.[59,60]

4. Spatiotemporal Simulation Methods

Depending on the modeling technique chosen for a given system (Fig. 2), different numerical methods exist for simulating the resulting model in a computer. While it is impossible to give an exhaustive list of all available methods, we will highlight the most important ones for each category of models. The same numerical methods can be used for both physical and phenomenological models.

4.1. *Methods for Discrete Stochastic Models*

Discrete stochastic models as formulated by events occurring according to certain probability distributions can be simulated using stochastic simulation algorithms (SSAs) such as Gillespie's algorithm[61,62] or the Gibson–Bruck algorithm.[63] While most of these algorithms were originally developed for temporal dynamics only, they have since been generalized to spatiotemporal models such as reaction-diffusion models.[64,65] Monte Carlo methods[66,67] provide the basis for most of these algorithms. Simulating probabilistic trajectories of the model thus amounts to sampling from the model probability distributions. In order to estimate the average trajectory or the standard deviation, ensemble averages over many simulations must be computed. This fundamentally limits the convergence properties of these methods to $\mathcal{O}(1/\sqrt{N})$, where N is the number of simulations performed.[68] Agent-based methods with probabilistic agents are also frequently used to

simulate discrete stochastic models. A prominent example is Brownian agents.[69]

4.2. *Methods for Discrete Deterministic Models*

Simulations of discrete deterministic models are frequently implemented using methods from the class of finite automata. The most prominent examples are cellular[a] automata[70–72] and agent-based simulations.[73] In finite automata, spatially distributed computational cells (or agents) with certain attributed properties interact according to sets of deterministic rules. These interaction rules map the state (the values of the attributed properties) of the interacting cells to certain actions, which in turn change the states of the cells. Finite automata are powerful and fascinating tools, as already small sets of simple rules can give rise to very complex nonlinear model behavior. They can be used for diverse purposes such as studying behavioral aspects of interacting individuals in ecosystems, studying artificial life, simulating interacting neurons,[71] simulating social interactions,[74] or simulating pattern-forming mechanisms in morphogenesis.[5] Another important class of discrete deterministic simulations is found in MD.[50] Here, the atomistic behavior of molecules is simulated by explicitly tracking the dynamics and positions of all atoms. Atoms in classical MD are modeled as discrete particles that interact according to deterministic mechanisms such as interatomic bonds, van der Waals forces, or electrostatics.

4.3. *Methods for Continuous Stochastic Models*

Continuous stochastic models formulated as SDEs can be numerically simulated using a variety of stochastic integration methods,[75,76] most notably Euler–Maruyama or Milstein's higher-order method.[76] It is, however, important to keep in mind that each simulation represents just one possible realization of the stochastic process. In order to estimate means and variances, many independent simulations need to be performed

[a] The word "cellular" refers to computational cells in the algorithm and does not imply any connection with biological cells.

and an ensemble average computed.[76] While the topic of SDEs may seem exotic to many computational biologists, it is more widespread than one would think. Simulating, for example, a reaction-diffusion model with stochastic reactions amounts to numerically solving an SDE.[64,65]

4.4. *Methods for Continuous Deterministic Models*

Continuous deterministic models as represented by PDEs can be solved using any of the discretization schemes from numerical analysis.[77,78] The most common ones include finite difference (FD) methods,[79] finite element (FE) methods,[80,81] and finite volume (FV) methods for conservation laws.[82,83] FD methods are based on Taylor series expansions[84] of the spatial field functions and approximation of the differential operators by difference operators such that the first few terms in the Taylor expansion are preserved. FE methods express the unknown field function in a given function space. The basis functions of this space are supported on polygonal elements that tile the computational domain. Determining the unknown field function then amounts to solving a linear system of equations for the weights of the basis functions on all elements. FV methods make use of physical conservation laws such as conservation of mass or momentum. The computational domain is subdivided into disjoint volumes, for each of which the balance equations are formulated (change of volume content equals inflow minus outflow) and numerically solved.

All of these methods have the common property that they require a computational mesh — regular or irregular — that discretizes the computational domain into simple geometric structures such as lines (FD), areas (FE), or volumes (FV) with the appropriate connectivity. For complex-shaped domains as they frequently occur in biological systems (cf. Fig. 1), it can be a daunting task to find a good connected mesh that respects the boundary conditions and has sufficient regularity to preserve the accuracy and efficiency of the numerical method. Mesh-free particle methods[85–87] relax this constraint by basing the discretization on point objects that do not require any connectivity information. While particle formulations are the natural choice for discrete models, their advantages can be transferred to the continuous domain using continuum particle

methods as described in Sec. 5; they are based on approximating the smooth field functions of a continuous model by integrals that are being discretized onto computational elements called particles. While the particles in discrete simulations represent real-world objects such as molecules, atoms, animals, or cells, particles in continuous methods are computational elements that collectively approximate a field quantity as outlined in Sec. 5.1.

4.5. *Representing Complex Geometries in the Computer*

Complex geometries and surfaces can be represented in the computer using a variety of methods,[88] which can be classified according to the connectivity information they require. Triangulated surfaces[89] are an example of connectivity-based representations, as they require each triangle to know which other triangles it is connected to. Establishing this connectivity information on complex-shaped surfaces is computationally expensive, so these representations are preferably used in conjunction with numerical methods that operate on the same connectivity. This is the case when using FE methods with triangular elements in simulations involving triangulated surfaces,[3,90] or FD methods in conjunction with pixelated surface representations.[91] An example of a complex triangulated surface is shown in Fig. 1(a).

Connectivity-less surface representations include scattered point clouds[92] and implicit surface representations such as level sets.[93] In level set methods, the geometry is implicitly represented as an isosurface of a higher-dimensional level function. Level sets are well suited to be used in combination with particle methods because the level function can directly be represented on the same set of computational particles.[94–96] This allows treating arbitrarily complex geometries at constant computational cost, and simulating moving and deforming geometries with no linear stability limit. The ER shown in Fig. 1(b) was, e.g. represented in the computer as a level set in order to simulate diffusion processes on its surface using particle methods.[96]

5. Introduction to Continuum Particle Methods

Particle methods are point-based spatiotemporal simulation methods that exhibit a number of favorable properties, which help address the complications of spatiotemporal biological systems (cf. Sec. 2):

- They are the most universal simulation method. Particle methods can be used for all types of models in Fig. 2, whereas most other numerical methods are limited to one or two types of models.
- They are intimately linked to the physical or biological process they represent, since particles correspond to real-world entities (in discrete models) or approximate field quantities (in continuous models). The interactions between the particles can mostly be intuitively understood as forces or exchange of mass. This prevents spurious, unphysical modes from showing up in the simulation results, a capability that has recently also been developed for FE methods.[97]
- They are well suited for simulations in complex geometries, such as the ER shown in Fig. 1,[49] and for simulations on complex curved surfaces such as intracellular membranes.[96] No computational mesh needs to be generated and no connectivity constraints satisfied. This effectively avoids the increased algorithmic complexity of mesh-based methods in complex geometries due to loss of the "nice" structure of the matrix.
- Due to their inherent regularity (particles have a finite size that defines the resolution of the method; cf. Sec. 5.1), particle methods can easily handle topological changes such as fusion and fission in the simulated geometry. Mesh-based methods need special regularization so as not to become unstable when two fusing or separating objects touch in exactly one point.
- They are inherently adaptive, as particles are only required where the represented quantity is present, and the motion of the particles automatically tracks these regions. This constitutes an important computational advantage compared to mesh-based methods, where a mesh is required throughout the computational domain.

- Particle methods are not subject to any linear convective stability condition (CFL condition[98,99]).[27,100] When simulating flows or waves with mesh-based methods, the CFL condition imposes a time step limit such that the flow or wave can never travel more than a certain number of mesh cells per time step. In particle methods, convection simply amounts to moving the particles with the local velocity field, and no limit on how far they can move is imposed as long as their trajectories do not intersect.
- Thanks to advancements in fast algorithms for N-body interactions (cf. Sec. 6), particle methods are as computationally efficient as mesh-based methods. In addition, the data structures and operators in particle simulations can be distributed across many processors, enabling highly efficient parallel simulations.[32]

Since continuum applications of particle methods are far less known than discrete ones, we focus on deterministic continuous models. For reasons of simplicity, however, we will not cover the most recent extensions of particle methods to multi-resolution and multi-scale[27–31] problems using concepts from adaptive mesh refinement,[101] adaptive global mappings,[101] or wavelets.[100]

In continuum particle methods, a particle p occupies a certain position x_p and carries a physical quantity $\boldsymbol{\omega}_p$, referred to as its strength. These particle attributes — strength and location — evolve so as to satisfy the underlying governing equations in a Lagrangian frame of reference.[27] Simulating a model thus amounts to tracking the dynamics of all N computational particles carrying the physical properties of the system being simulated. The dynamics of the N particles are governed by sets of ordinary differential equations (ODEs) that determine the trajectories of the particles p and the evolution of their properties $\boldsymbol{\omega}$ over time. Thus,

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}_p}{\mathrm{d}t} &= \boldsymbol{v}_p(t) = \sum_{q=1}^{N} \boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{x}_q; \boldsymbol{\omega}_p, \boldsymbol{\omega}_q) \qquad p = 1, 2, \ldots, N \\ \frac{\mathrm{d}\boldsymbol{\omega}_p}{\mathrm{d}t} &= \sum_{q=1}^{N} \boldsymbol{F}(\boldsymbol{x}_p, \boldsymbol{x}_q; \boldsymbol{\omega}_p, \boldsymbol{\omega}_q) \qquad p = 1, 2, \ldots, N, \end{aligned} \tag{1}$$

where $\boldsymbol{v}_p(t)$ is the velocity of particle p at time t. The dynamics of the particles are completely defined by the functions $\boldsymbol{K}$ and $\boldsymbol{F}$, which represent the model being simulated. In pure particle methods, $\boldsymbol{K}$ and $\boldsymbol{F}$ emerge from integral approximations of differential operators (cf. Sec. 5.2.1); in hybrid particle-mesh (PM) methods, they entail solutions of field equations that are discretized on a superimposed mesh (cf. Sec. 5.2.2). The sums on the right-hand side of Eq. (1) correspond to quadrature[84] (numerical integration) of some functions.

In order to situate continuum particle methods on the map of numerical analysis, we consider the different strategies to numerically solve a differential equation as outlined in Fig. 4 for the example of a simple PDE, the Poisson equation, which we wish to solve for the intensive field quantity u. One way consists of discretizing the equation onto a computational mesh with resolution h using FD, FE, or FV, and then numerically solving the resulting system $\mathbf{A}\boldsymbol{u} = \boldsymbol{f}$ of linear algebraic equations. The discretization needs to be done consistently in order to ensure that the discretized equations model the same system as the original PDE, and the numerical solution of the resulting linear system is subject to stability criteria. An alternative route is to solve the PDE analytically

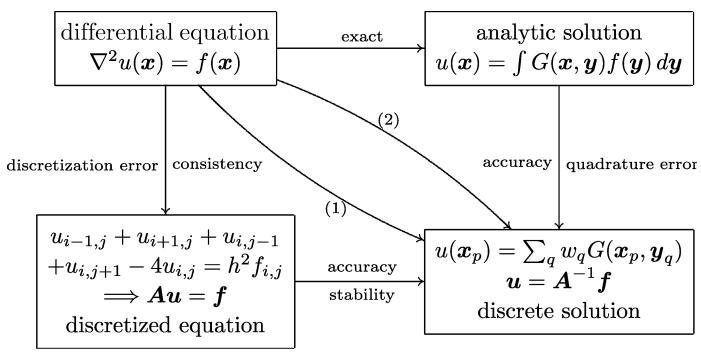


Fig. 4. Strategies to numerically solve a differential equation, illustrated on the example of the two-dimensional (2D) Poisson equation: (1) discretization of the PDE on a mesh with resolution h, followed by numerical solution of the discretized equations for the intensive property u; or (2) integral solution for the extensive property that is numerically approximated by quadrature.

using Green's function[55] $G(\boldsymbol{x}, \boldsymbol{y})$.[b] The resulting integral defines an extensive quantity that is then discretized and computed by a quadrature[84] with some weights w. The values of the weights depend on the particular quadrature rule used. For midpoint quadrature[102] and the example of Fig. 4, they would be $w_q = f(\boldsymbol{y}_q)\mathrm{d}\boldsymbol{y}$. This defines the right-hand side of Eq. (1) for this example. The advantages of the latter procedure are that the integral solution is always consistent (even analytically exact), and that numerical quadrature is always stable. The only property that remains to be concerned about is the solution's accuracy. The first way of solution is sometimes referred to as the "intensive method", and the second as the "extensive method".

5.1. *Function Approximation by Particles*

The approximation of a continuous field function $u(\boldsymbol{x})$ by particles in d-dimensional space can be developed in three steps:

- *Step 1: integral representation*. Using the Dirac δ-function identity, the function u can be expressed in integral form as

$$u(\boldsymbol{x}) = \int u(\boldsymbol{y})\, \delta(\boldsymbol{x} - \boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{2}$$

  In point particle methods such as random walk (cf. Sec. 7.1), this integral is directly discretized on the set of particles using a quadrature rule with the particle locations as quadrature points. Such a discretization, however, does not allow recovering the function values at locations other than those occupied by the particles.
- *Step 2*: *integral mollification*. Smooth particle methods relax this limitation by regularizing the δ-function by a mollification kernel $\zeta_\epsilon = \epsilon^{-\mathrm{d}}\zeta(\boldsymbol{x}/\epsilon)$, with $\lim_{\epsilon\to 0}\, \zeta_\epsilon = \delta$, that conserves the first $r - 1$ moments of the δ-function identity.[85] The kernel ζ_ϵ can be thought of as a cloud or blob of strength, centered at the particle location,

[b] Note that Green's function always exists, even though it may not be known in closed form in most cases.

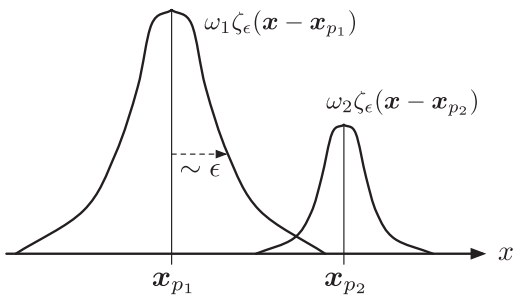


Fig. 5. Two particles of strengths ω_1 and ω_2, carrying mollification kernels ζ_ϵ.

as illustrated in Fig. 5. The core size ϵ defines the characteristic width of the kernel and thus the spatial resolution of the method. The regularized function approximation is defined as

$$u_\epsilon(\boldsymbol{x}) = \int u(\boldsymbol{y})\zeta_\epsilon(\boldsymbol{x}-\boldsymbol{y})\mathrm{d}\boldsymbol{y} \tag{3}$$

and can be used to recover the function values at arbitrary locations $\boldsymbol{x}$. The approximation error is of order ϵ^r, hence

$$u_\epsilon(\boldsymbol{x}) = u(\boldsymbol{x}) + \mathcal{O}(\epsilon^r). \tag{4}$$

As introduced above, r is the first nonvanishing moment of the mollification kernel.[27,85] For positive symmetric kernels, such as a Gaussian, $r = 2$.

- *Step 3: mollified integral discretization*. The regularized integral in Eq. (3) is discretized over N particles using the quadrature rule

$$u_\epsilon^h(\boldsymbol{x}) = \sum_{p=1}^{N} \omega_p^h \zeta_\epsilon\left(\boldsymbol{x}-\boldsymbol{x}_p^h\right), \tag{5}$$

where $\boldsymbol{x}_p^h$ and ω_p^h are the numerical solutions of the particle positions and strengths, determined by discretizing the ODEs in Eq. (1) in time. The quadrature weights ω_p are the particle strengths and depend on the particular quadrature rule used. The most frequent choice is to use midpoint quadrature,[102] thus setting $\omega_p = u(\boldsymbol{x}_p)V_p$,

where V_p is the volume of particle p. Using this discretization, we obtain the function approximation

$$u_\epsilon^h(\boldsymbol{x}) = u_\epsilon(\boldsymbol{x}) + \mathcal{O}\left(\frac{h}{\epsilon}\right)^s = u(\boldsymbol{x}) + \mathcal{O}(\epsilon^r) + \mathcal{O}\left(\frac{h}{\epsilon}\right)^s, \tag{6}$$

where s depends on the number of continuous derivatives of the mollification kernel ζ_ϵ,[27,85] and h is the interparticle distance. For a Gaussian, $s \to \infty$.

From the approximation error in Eq. (6), we see that it is imperative that the distance h between any two particles is always less than the kernel core size ϵ, thus maintaining

$$\frac{h}{\epsilon} < 1 \tag{7}$$

at all times. If this "particle overlap" condition is violated, the approximation error becomes arbitrarily large, and the method ceases to be well posed.

5.2. *Operator Approximation*

Two strategies are distinguished to evaluate differential operators on particles: pure particle methods and hybrid PM methods.

5.2.1. *Pure particle methods*

In pure particle methods, differential operators on functions that are represented on particles are approximated by integral operators. The sums on the right-hand side of Eq. (1) thus represent the discretized versions of these integral operators. If we are interested in diffusion processes, for example, the relevant differential operators are ∇^2 and $\nabla \cdot (\boldsymbol{D}\nabla)$ (cf. Sec. 7). Both of them can be approximated by integral operators that allow consistent evaluation on scattered particle locations and conserve mass exactly[103,104] (cf. Sec. 7.2). The concept of this approximation method has been extended to a general, systematic framework for approximating

any differential operator by a corresponding integral.[105] Following this framework, a differential operator L^β of order β applied to a continuous function $u(\boldsymbol{x})$ is equivalent to the integral operator

$$L^\beta u(\boldsymbol{x}) = \frac{1}{\epsilon^{|\beta|}} \int (u(\boldsymbol{y}) \pm u(\boldsymbol{x})) \eta_\epsilon^\beta (\boldsymbol{x} - \boldsymbol{y}) \mathrm{d}\boldsymbol{y} \tag{8}$$

with a suitable scaled kernel $\eta_\epsilon^\beta(\boldsymbol{x}) = \epsilon^{-d} \eta^\beta(\boldsymbol{x}/\epsilon)$ of core size ϵ. This integral operator is then discretized over the particles using, e.g., midpoint quadrature[102] of resolution h, yielding

$$L_h^\beta u(\boldsymbol{x}_p) = \frac{1}{\epsilon^{|\beta|}} \sum_q V_q (u(\boldsymbol{x}_q) \pm u(\boldsymbol{x}_p)) \eta_\epsilon^\beta (\boldsymbol{x}_p - \boldsymbol{x}_q). \tag{9}$$

Pure particle methods thus amount to evaluating direct particle–particle (PP) interactions, which means that for each particle $p = 1, 2, \ldots, N$ we have to compute a sum over all particles $q = 1, 2, \ldots, N$ [cf. Eq. (1)]. The computational complexity of this N-body problem thus nominally scales as $\mathcal{O}(N^2)$. Efficient algorithms do, however, exist to reduce it to $\mathcal{O}(N)$ in all practical cases. These algorithms will be outlined in Sec. 6. Alternatively, hybrid PM methods, as described next, can be used.

5.2.2. *Hybrid particle-mesh (PM) methods*

In hybrid PM methods, as pioneered by Harlow,[106] some (but not all) of the differential operators are evaluated on a superimposed regular Cartesian mesh.[107] This amounts to splitting the operators into separate short-range and long-range contributions. The short-range interactions are directly evaluated on the particles, whereas the long-range contributions are evaluated on the mesh. Using direct PP interactions for the short-range part allows better resolving local phenomena and retaining the favorable stability properties of particle methods in the case of convection (moving the particles is a local operation). Prominent examples of hybrid PM methods can be found in fluid dynamics and electrostatics. Both applications involve long-range interactions in order to compute the velocities (fluid dynamics) or

forces (electrostatics). In a hybrid PM approach, such long-range interactions are modeled by a corresponding field equation that is then solved on the mesh. In many applications, the fields to be discretized are gradient fields, such that the corresponding long-range operator is the Laplace operator and the field equation hence is the Poisson equation (cf. Fig. 4). This equation can efficiently be solved using, e.g. FD[79] implemented in a multigrid algorithm,[108] or Poisson solvers based on fast Fourier transforms. In hybrid PM methods, the functions $\boldsymbol{K}$ and $\boldsymbol{F}$ in Eq. (1) may thus contain contributions corresponding to the solution of the field equation on the mesh. Therefore, hybrid methods require

- interpolation of the $\boldsymbol{\omega}_p$ carried by the particles from the irregular particle locations $\boldsymbol{x}_p$ onto the M regular mesh points ($\boldsymbol{\omega}_m$):

$$\boldsymbol{\omega}_m^h = \sum_{p=1}^{N} Q(\boldsymbol{x}_m - \boldsymbol{x}_p^h)\boldsymbol{\omega}_p^h \qquad m = 1, 2, \ldots, M; \tag{10}$$

- and interpolation of the field solution $\boldsymbol{F}_m$ (and, similarly, $\boldsymbol{K}_m$ if present) from the mesh to the (not necessarily same) particle locations ($\boldsymbol{F}_p$):

$$\boldsymbol{F}_p^h = \sum_{m=1}^{M} R(\boldsymbol{x}_p^h - \boldsymbol{x}_m)\boldsymbol{F}_m^h \qquad p = 1, 2, \ldots, N. \tag{11}$$

The accuracy of the method depends on the smoothness of $\boldsymbol{K}$ and $\boldsymbol{F}$, on the interpolation functions Q and R, and on the mesh-based discretization scheme employed for the solution of the field equations. In order to achieve high accuracy, the interpolation functions Q and R must be smooth to minimize local errors, and conserve the moments of the interpolated quantity to minimize far-field errors.[27] In addition, it is necessary that Q is at least of the same order of accuracy as R in order to avoid spurious contributions to $\boldsymbol{F}_p^h$.[107] This can easily be achieved by selecting the same interpolation function, W, for both operations: $Q = R = W$. Accurate interpolation functions that conserve the moments of the interpolated

quantity up to a certain order can be constructed systematically.[109] Conservation of moments is an important property in the simulation of biological systems, where the laws of physics require quantities such as mass (zero-order moment), impulse (first-order moment), and angular impulse (second-order moment) to be conserved. Building this conservation right into the method constitutes an obvious advantage. One of the most commonly used moment-conserving interpolation kernels (but not the only one) is the M_4' function,[109] given by

$$M_4'(s) = \begin{cases} 1 - \frac{1}{2}(5s^2 + 3s^3) & \text{if } 0 \le s < 1 \\ \frac{1}{2}(2-s)^2(1-s) & \text{if } 1 \le s \le 2 \\ 0 & \text{if } s > 2, \end{cases} \tag{12}$$

where $s = |\boldsymbol{x}|/h = |\boldsymbol{x}_p - \boldsymbol{x}_m|/h$. Hereby, h denotes the mesh spacing and x is the distance from the particle to the respective mesh node, as illustrated in Fig. 6. The M_4' kernel is third-order accurate, exactly conserving moments up to and including the second moment. For each particle–node pair, we compute one weight $0 \le W_{pm} \le 1$ and the portion $\boldsymbol{\omega}_m = W_{pm}\boldsymbol{\omega}_p$ of the strength of particle p is attributed to mesh node

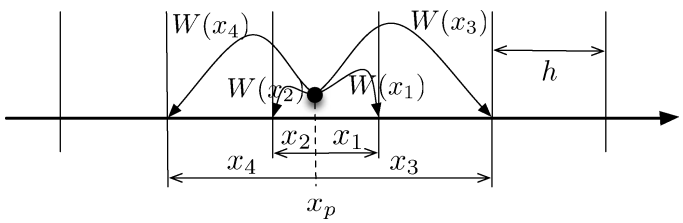


Fig. 6. Particle-to-mesh interpolation in one dimension. The interpolation weight is computed from the mesh spacing h and the distance x between the particle and the mesh node. For each particle–node pair, a different weight is computed. The particle strength is then multiplied by these weights and assigned to the mesh nodes. Mesh-to-particle interpolation works analogously and uses the same interpolation kernels.

m (Fig. 6). This is done independently for each particle and can efficiently be parallelized on vector and multi-processor computers.[32] In higher dimensions, the kernels are tensorial products of the one-dimensional (1D) kernels. Their values can thus be computed independently in each spatial direction and then multiplied to form the final interpolation weight for a given particle and mesh node: $W(x,y,z) = W_x(x)\,W_y(y)\,W_z(z)$.

Meshes are used not only to accelerate the computation of long-range interactions in hybrid PM schemes, but also to periodically reinitialize the particle locations to regular positions in order to maintain the overlap condition of Eq. (7). Reinitialization using a mesh is needed if particles tend to accumulate in certain areas of the computational domain and to disperse in others. In such cases, the function approximation would cease to be well posed as soon as the condition in Eq. (7) is violated. This can be prevented by periodically resetting the particle positions to regular locations by interpolating the particle properties to the nodes of a regular Cartesian mesh as outlined above, discarding the present set of particles, and generating new particles at the locations of the mesh nodes. This procedure is called remeshing.[85]

6. Efficient Algorithms for Particle Methods

The evaluation of PP interactions is a key component of particle methods and PM algorithms. Equation (1), however, defines an N-body problem, which is of potentially $\mathcal{O}(N^2)$ complexity to solve. It is this high computational cost that has long prevented the use of particle methods in computational science. Fortunately, this can be circumvented and the complexity can be reduced to $\mathcal{O}(N)$ in all practical cases. Together with efficient implementations on parallel computers,[32] this makes particle methods a competitive alternative to mesh-based methods.

If the functions $\boldsymbol{K}$ and $\boldsymbol{F}$ in Eq. (1) are local (but not necessarily compact), the algorithmic complexity of the sums in Eq. (1) naturally reduces to $\mathcal{O}(N)$ by considering only interactions within a certain cut-off radius r_c around each particle. This corresponds to short-range interactions where only nearby neighbors of a given particle significantly contribute. The specific value of r_c depends on the interaction law, i.e. the kernel

functions $\boldsymbol{K}$ and $\boldsymbol{F}$ in Eq. (1), and has to be chosen to meet the desired simulation accuracy. The most conservative choice of r_c is given by the radius where the interaction contributions fall below the machine epsilon of the computer[84] and hence become insignificant.

For long-range interactions whose value decays as $\mathcal{O}(1/r^2)$ or slower with increasing interparticle distance r, cut-offs are not appropriate and we have to consider the full N-body problem of Eq. (1). Examples of such interactions include Coulomb forces, gravitation, or the Biot–Savart law in electromagnetism and fluid dynamics. Fast algorithms such as multipole expansions[110] (cf. Sec. 6.2) are, however, available to reduce the complexity of the corresponding pure particle method to $\mathcal{O}(N)$ also in these cases, albeit with a large prefactor. This large prefactor typically causes pure particle implementations of long-range interactions to be several orders of magnitude slower than the corresponding hybrid PM algorithm. Nevertheless, fast N-body methods are appealing from a conceptual point of view.

6.1. *Fast Algorithms for Short-Range Interactions*

Considering only the interactions within an r_c-neighborhood naturally reduces the algorithmic complexity of the PP evaluation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ with a prefactor that depends on the value of r_c and the local particle density. This requires, however, that the set of neighbors to interact with is known or can be determined with at most $\mathcal{O}(N)$ cost. Since particle methods do not use any connectivity information (cf. Sec. 4.4), neighborhood information is not explicitly available and it changes over time if particles move. Finding the neighbors of each particle by searching over all other particles would again render the complexity of the algorithm $\mathcal{O}(N^2)$, annihilating all benefits of a finite cut-off radius r_c. Two standard methods are available to find the interaction partners in $\mathcal{O}(N)$ time: cell lists and Verlet lists.

In cell lists, particles are sorted into equisized cubic cells whose size corresponds to the interaction cut-off r_c. Each cell contains a (linked) list of the particles residing in it. Interactions are then computed by sweeping through these lists. If particle p is to interact with all of its neighbors closer than r_c, this involves considering all other particles in the same cell

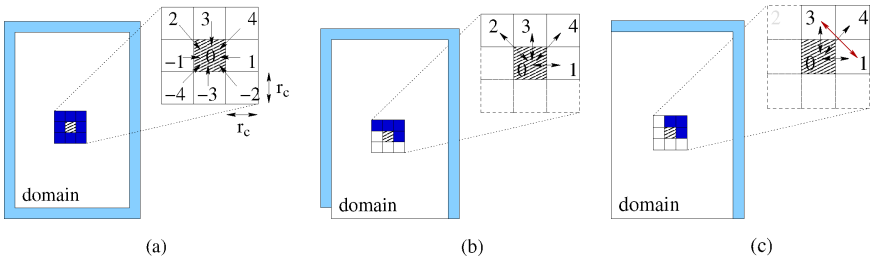


Fig. 7. Cell–cell interactions in cell list algorithms. (a) For asymmetric PP interactions, all adjacent cells have to be considered and the interactions are one-sided. (b) In traditional symmetric cell list algorithms, interactions are required on all but one boundary. (c) Introducing diagonal interactions (1–3), the cell layers for the boundary conditions (light blue; cf. Sec. 7.2.2) also become symmetric. This reduces the memory overhead and improves the efficiency of parallel implementations by reducing the communication volume. The 2D case is depicted. See text for interactions in the 3D case.

as particle p (center cell) as well as all particles in all immediately adjacent cells [Fig. 7(a)]. The shaded areas around the computational domain in Fig. 7 are needed to satisfy the boundary conditions using the method of images as outlined in Sec. 7.2.2.

For spherically symmetric interactions in 3D, cell lists contain up to $27/(4\pi/3) \approx 6$ times more particles than actually needed. Verlet lists[111] are available to reduce this overhead. For each particle p, they consist of an explicit list of all other particles with which it has to interact. This list contains the indices of all particles within a sphere around $\boldsymbol{x}_p$. The radius of this Verlet sphere has to be at least r_c, but is usually enlarged by a certain margin (skin) in order for the Verlet lists to be valid over several simulation time steps. The Verlet lists need to be rebuilt as soon as any particle has moved farther than the skin margin. Choosing the skin size is a trade-off between minimizing the lengths of the lists (and hence the number of interactions to be computed) and maximizing the time between list updates.[112] In the 3D case, Verlet list algorithms are at most $81/(4\pi(1 + \text{skin})^3)$ times faster than cell list algorithms. In order to ensure overall $\mathcal{O}(N)$ scaling, Verlet lists are constructed using intermediate cell lists.

Another point of possible optimization concerns the symmetry of PP interactions. By construction of the kernel-based interactions, the effect of a particle p on another particle q is the same (with a possible sign change) as the effect of particle q on p. Looping over all particles and computing the interactions with all neighbors within the cut-off radius thus considers every interaction twice. The computational cost can be reduced by a factor of (at most) two if interactions are evaluated symmetrically. We then only loop over half of the neighbors and attribute the interaction contributions to both participating particles at once. How, then, is it possible to make sure that all interactions are considered exactly once? In cell lists, it is sufficient to loop over only those particles q in the center cell for which $q > p$, as well as over all particles in half of the neighboring cells [Fig. 7(b)]. In Fig. 7(c), diagonal interactions are introduced in order to further reduce the memory overhead for the boundary layers by 33% in the 2D case and 40% in the 3D case.[32] In parallel implementations, the diagonal interaction scheme moreover has the advantage of lower communication overhead. If the cells are numbered in ascending x,y,(z), starting from the center cell with number 0, the symmetric cell–cell interactions are[32] 0–0, 0–1, 0–3, 0–4, and 1–3 in the 2D case; and 0–0, 0–1, 0–3, 0–4, 0–9, 0–10, 0–12, 0–13, 1–3, 1–9, 1–12, 3–9, 3–10, and 4–9 in the 3D case. Verlet list algorithms remain unchanged in the symmetric case, as the Verlet lists are constructed using intermediate symmetric cell lists and hence only contain unique interactions in the first place.

6.2. *Fast Algorithms for Long-Range Interactions*

In 1986, Joshua Barnes and Piet Hut[113] introduced a fast hierarchical $\mathcal{O}(N \log N)$ algorithm for N-body problems. The Barnes–Hut algorithm divides the domain into a tree of regular cuboidal cells. Each cell in the tree has half the edge length of its parent cell, and stores information about the center of mass and the total strength of all particles inside. The tree is then traversed for each particle p for which the interactions are to be evaluated. Direct PP interactions are only computed for nearby interaction partners q. If the partners are sufficiently far away, they are collectively

approximated by the center of mass and the total strength of the largest possible cell that satisfies the closeness criterion

$$\frac{d}{\Delta} < \theta, \tag{13}$$

where d is the diagonal of the cell currently being considered, Δ is the distance of particle p from the center of mass of that cell, and θ is a fixed accuracy parameter ~1. This amounts to coarse-graining clusters of remote particles to single particles.

Based on the Barnes–Hut algorithm, Leslie Greengard and Vladimir Rokhlin presented the fast multipole method (FMM).[110,114,115] Their formulation uses a finite series expansion of the interaction kernel and direct cell–cell interactions in the tree. Compared to the Barnes–Hut algorithm, this further reduces the algorithmic complexity to $\mathcal{O}(N)$.

7. Particle Methods for the Simulation of Diffusion Processes

We consider the simulation of continuous spatial diffusion processes as a simple example of biological relevance.[116] Physically, the macroscopic phenomenon of diffusion is created by the collective behavior of a large (in theory, infinite) number of microscopic particles, such as molecules, undergoing Brownian motion.[116–118] From continuum theory,[119] we can define a concentration field as the mean mass of particles per unit volume at every point in space (cf. Sec. 3.2). For abundant diffusing particles, this allows formulating a continuous deterministic model for the spatiotemporal evolution of the concentration field $u(\boldsymbol{x}, t)$ in a closed, bounded domain Ω. This model is formulated as the PDE

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \nabla \cdot (\boldsymbol{D}(\boldsymbol{x},t)\nabla u(\boldsymbol{x},t)) \quad \text{for } \boldsymbol{x} \in \{\Omega/\partial\Omega\}, \quad 0 < t \leq T. \tag{14}$$

In this diffusion equation, $\boldsymbol{D}(\boldsymbol{x},t)$ denotes the diffusion tensor, ∇ the Nabla operator, and $\partial\Omega$ the boundary of the domain Ω.

Terminology classifies diffusion processes based on the structure of the diffusion tensor:

- If $\boldsymbol{D}$ is constant everywhere in Ω, diffusion is called "homogeneous". A $\boldsymbol{D}$ that varies in space defines "inhomogeneous diffusion".
- If $\boldsymbol{D}$ is proportional to the identity matrix, $\boldsymbol{D} = D\mathbb{1}$, diffusion is called "isotropic"; otherwise, "anisotropic". Isotropic diffusion is characterized by a flux whose magnitude does not depend on its direction, and it can be described using a scalar diffusion constant D. For isotropic, homogeneous diffusion, the diffusion equation simplifies to

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = D\nabla^2 u(\boldsymbol{x},t) \quad \text{for } \boldsymbol{x} \in \{\Omega/\partial\Omega\}, \;\; 0 < t \leq T, \tag{15}$$

where ∇^2 is the Laplace operator.

At $t = 0$, the concentration field is specified by an initial condition

$$u(\boldsymbol{x}, t = 0) = u_0(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega.$$

The model is completed by problem-specific boundary conditions prescribing the behavior of u along $\partial\Omega$. The most frequently used types of boundary conditions are Neumann and Dirichlet conditions. A Neumann boundary condition fixes the diffusive flux through the boundary to a prescribed value f_N ($\boldsymbol{n}$ is the outer unit normal on the boundary):

$$\frac{\partial u}{\partial \boldsymbol{n}} = \nabla u(\boldsymbol{x},t)\cdot\boldsymbol{n} = f_N(\boldsymbol{x},t) \quad \text{for } \boldsymbol{x} \in \partial\Omega, \;\; 0 < t \leq T;$$

whereas a Dirichlet condition prescribes the concentration f_D at the boundary:

$$u(\boldsymbol{x},t) = f_D(\boldsymbol{x},t) \quad \text{for } \boldsymbol{x} \in \partial\Omega, \;\; 0 < t \leq T.$$

If the boundary function f is 0 everywhere on $\partial\Omega$, the boundary condition is called "homogeneous".

In the framework of pure particle methods, continuous diffusion models can be simulated using particles carrying mass as their extensive strength ω and collectively representing the intensive concentration field u. In the following, we review the stochastic method of random walk (RW) and the deterministic particle strength exchange (PSE) method. Using a 1D test problem, we then compare the accuracy and the convergence behavior of the two methods.

7.1. *The Method of Random Walk (RW)*

The Random Walk (RW)[116,120] method is based on the stochastic interpretation of Green's function solution[55] (cf. Fig. 4) of the diffusion equation:

$$u(\boldsymbol{x},t) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y},t)u_0(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{16}$$

In the case of d-dimensional isotropic homogeneous free-space diffusion, i.e. $\boldsymbol{D} = D\mathbb{1}$ and $\Omega = \mathbb{R}^d$, Green's function is explicitly known to be[121]

$$G(\boldsymbol{x},\boldsymbol{y},t) = \frac{1}{(4\pi Dt)^{d/2}}\exp\left[-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{4Dt}\right]. \tag{17}$$

The RW method interprets this function as the transition density of a stochastic process.[122] In d dimensions, the method starts by either uniformly or randomly placing N particles p at initial locations $\boldsymbol{x}_p^0$, $p = 1, 2,\ldots,N$. Each particle is assigned a strength of $\omega_p = V_p u_0(\boldsymbol{x}_p^0)$, where V_p is the particle volume. This defines a point particle function approximation (cf. Sec. 5.1) to the initial concentration field $u_0(\boldsymbol{x})$. The particles then undergo a random walk by changing their positions at each positive-integer time step n according to the transition density in Eq. (17):

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \mathcal{N}_p^n(0, 2D\delta t), \tag{18}$$

where $\mathcal{N}_p^n(0, 2D\delta t)$ is a vector of i.i.d. Gaussian random numbers with each component having a mean of zero and a variance of $2D\delta t$; δt is the simulation time step size. Moving the particles according to Eq. (18) creates a concentration field that, for $N \to \infty$, converges to the exact solution of the diffusion equation as given in Eq. (16). Homogeneous Neumann boundary conditions can be satisfied by reflecting the particles at the boundary. Drawing the step displacements in Eq. (18) from a multivariate Gaussian distribution readily extends the RW method to anisotropic diffusion processes.

RW is a stochastic simulation method. This Monte Carlo[66,67] character limits its convergence capabilities (cf. Sec. 4.1), since the variance of the mean of N i.i.d. random variables is given by $1/\sqrt{N}$ times the individual variance of a single random variable[68] (cf. Sec. 7.3). Moreover, the solution deteriorates with increasing diffusion constant D as the variance of the random variables becomes larger. In the case of small D ($\ll \delta t$), the motion of the particles can be masked by the sampling noise. RW thus works best for an intermediate range of diffusion constants.

7.2. *Particle Strength Exchange (PSE)*

The Particle Strength Exchange (PSE)[103,104] method is a deterministic pure particle method to simulate continuous diffusion processes in space. It is based on approximating the diffusion operator by a mass-conserving integral operator that can be consistently evaluated on the particle locations (cf. Sec. 5.2.1). The PSE scheme has been devised by Degond and Mas-Gallic for both isotropic[103] and anisotropic[104] diffusion. We illustrate the concept in the isotropic case. Anisotropic PSE is analogous and follows a similar derivation.[104]

7.2.1. *PSE for isotropic diffusion*

In free space, i.e. $\Omega = \mathbb{R}^d$, the isotropic PSE method[103] obtains an integral approximation to the Laplace operator in Eq. (15) by considering the

concentration at a location $\boldsymbol{y}$ and expanding it into a Taylor series[84] around $\boldsymbol{x}$:

$$u(\boldsymbol{y}) = u(\boldsymbol{x}) + \sum_{i=1}^{r+1}\left[\frac{1}{i!}\Big((\boldsymbol{y}-\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}'}\Big)^i u(\boldsymbol{x}')\right]_{\boldsymbol{x}'=\boldsymbol{x}} + \mathcal{O}\Big(\|\boldsymbol{y}-\boldsymbol{x}\|_2^{r+2}\,\|u\|_\infty\Big). \tag{19}$$

Subtracting $u(\boldsymbol{x})$ on both sides, multiplying the whole equation by a scaled kernel function $\eta_\epsilon(\boldsymbol{x}) = \epsilon^{-d}\eta(\boldsymbol{x}/\epsilon)$ of core size $\epsilon > 0$, and integrating over $\boldsymbol{y}$ yields

$$\int_{\mathbb{R}^d}(u(\boldsymbol{y})-u(\boldsymbol{x}))\eta_\epsilon(\boldsymbol{x}-\boldsymbol{y})\mathrm{d}\boldsymbol{y} = \sum_{i=1}^{r+1}\frac{1}{i!}\int_{\mathbb{R}^d}\left[\Big((\boldsymbol{y}-\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}'}\Big)^i u(\boldsymbol{x}')\right]_{\boldsymbol{x}'=\boldsymbol{x}}\eta_\epsilon(\boldsymbol{x}-\boldsymbol{y})\mathrm{d}\boldsymbol{y} + \|u\|_\infty\,\mathcal{O}\Big(\int_{\mathbb{R}^d}\|\boldsymbol{y}-\boldsymbol{x}\|_2^{r+2}\,\eta_\epsilon(\boldsymbol{x}-\boldsymbol{y})\mathrm{d}\boldsymbol{y}\Big). \tag{20}$$

For the approximation to be consistent, we have to ask the following requirement for the kernel function η[103]:

$$\int_{\mathbb{R}^d}\prod_{i=1}^{d}x_i^{\alpha_i}\eta(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \begin{cases}0, & \forall\boldsymbol{\alpha}\in\mathbb{N}^{\mathrm{d}},\ \ \boldsymbol{\alpha}\neq 2\boldsymbol{e}_i,\ \ 1\le\sum_{i=1}^{d}\alpha_i\le r+1\\ 2, & \text{if } \boldsymbol{\alpha}=2\boldsymbol{e}_i,\ \ i\in\{1,2,\ldots,d\}.\end{cases} \tag{21}$$

r is the order of the approximation and $\boldsymbol{x} = (x_1, x_2,\ldots, x_d) \in \mathbb{R}^d$. $\boldsymbol{\alpha} = (\alpha_1, \alpha_2,\ldots,\alpha_d) \in \mathbb{N}^{\mathrm{d}}$ is a d-dimensional index and $(\boldsymbol{e}_1, \boldsymbol{e}_2,\ldots,\boldsymbol{e}_d)$ is the canonical basis of $\mathbb{R}^d$. In the 3D case, the above requirement can be expressed as

$$\int_{\mathbb{R}^3} x_i x_j \eta(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 2\delta_{ij} \qquad \text{for } i,j = 1,2,3 \tag{22}$$

$$\int_{\mathbb{R}^3} x_1^{i_1} x_2^{i_2} x_3^{i_3}\, \eta(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 0 \quad \text{if } i_1 + i_2 + i_3 = 1 \text{ or } 3 \le i_1 + i_2 + i_3 \le r + 1 \tag{23}$$

$$\int_{\mathbb{R}^3} \|\boldsymbol{x}\|_2^{r+2} \left|\eta(\boldsymbol{x})\right| \mathrm{d}\boldsymbol{x} < \infty \tag{24}$$

for $i_1, i_2, i_3 \in \mathbb{N}_0^+$, and $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The first condition normalizes the kernel function. The second one requires all moments up to order $r + 1$ to vanish, and the third one is required in order to bound the truncation error. Using the requirement in Eq. (21), the only remaining terms in Eq. (20) are

$$\epsilon^{-2} \int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\eta_\epsilon(\boldsymbol{x} - \boldsymbol{y})\mathrm{d}\boldsymbol{y} = \nabla^2 u(\boldsymbol{x}) + \mathcal{O}(\epsilon^r), \tag{25}$$

and the integral operator that approximates the Laplacian is found as

$$\nabla_\epsilon^2 u(\boldsymbol{x}) = \epsilon^{-2} \int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))\eta_\epsilon(\boldsymbol{x} - \boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{26}$$

While this operator is not the only possibility of discretizing the Laplacian onto particles, it has the big advantage of conserving mass exactly.[85] The approximation error is $\mathcal{O}(\epsilon^r)$, with r being the largest integer for which the condition in Eq. (21) is fulfilled.[85] Equation (26) is discretized using the particle locations as quadrature points. Thus,

$$\nabla_{\epsilon,h}^2 u^h(\boldsymbol{x}_p^h) = \epsilon^{-2} \sum_{q=1}^{N} (V_q u_q^h - V_q u_p^h)\eta_\epsilon(\boldsymbol{x}_p^h - \boldsymbol{x}_q^h), \tag{27}$$

where V_q is the volume of particle q. Inserting this discretized operator into Eq. (15), the final PSE scheme for isotropic diffusion reads

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{x}_p^h}{\mathrm{d}t} &= 0\\ \frac{\mathrm{d}\boldsymbol{\omega}_p^h}{\mathrm{d}t} &= V_p D\epsilon^{-2}\sum_{q=1}^{N}(V_q u_q^h - V_q u_p^h)\eta_\epsilon(\boldsymbol{x}_p^h - \boldsymbol{x}_q^h)\\ p &= 1,2,\ldots,N.\end{aligned} \tag{28}$$

The PSE kernel η_ϵ is local and the fast algorithms described in Sec. 6.1 can be used to reduce the computational complexity to $\mathcal{O}(N)$. In order to simulate diffusion using PSE, the strengths of all the particles change (i.e. they exchange mass) while their locations remain constant (i.e. they do not move). This is dual to the method of RW, where the particles conserve their mass but move in space. In PSE, all geometry and boundary condition processing thus only needs to be done once when initializing the particles. Combined convection-diffusion problems can be simulated by moving the particles with the convective velocity field instead of keeping them fixed.

Besides the obvious choice of using a Gaussian [cf. Eq. (17)] as the PSE kernel η, various algebraic kernels have also been derived. Algebraic kernels are computationally more efficient, since evaluating the exponential function on the floating-point unit of a computer processor takes several tens of clock cycles. In the 3D case, the following second-order accurate kernel as proposed by G.-H. Cottet (private communication, 1999) can, for example, be used:

$$\eta(\boldsymbol{x}) = \frac{15}{\pi^2}\frac{1}{|\boldsymbol{x}|^{10}+1}. \tag{29}$$

7.2.2. *Boundary conditions in PSE*

The PSE algorithm as described so far only applies to infinite domains or to particles farther away from the boundary than r_c. For particles within an r_c-neighborhood from the boundary, we need to modify the PSE scheme in order to account for the prescribed boundary conditions.

For homogeneous boundary conditions in the case of flat (compared to the core size ϵ of the mollification kernel) boundaries, a straightforward method consists of placing mirror particles in an r_c-neighborhood outside of the simulation domain (Fig. 7). In the resulting method of images, the integral operator becomes

$$\epsilon^{-2} \int_{\mathbb{R}^d} (u(\boldsymbol{y}) - u(\boldsymbol{x}))(\eta_\epsilon(\boldsymbol{x} - \boldsymbol{y}) \pm \eta_\epsilon(\boldsymbol{x} + \boldsymbol{y}))\mathrm{d}\boldsymbol{y} + \mathcal{O}(\epsilon^r). \quad (30)$$

The final scheme is thus represented as

$$\frac{\mathrm{d}\boldsymbol{\omega}_p^h}{\mathrm{d}t} = V_p D\epsilon^{-2} \sum_{q=1}^{N} (V_q u_q^h - V_q u_p^h)(\eta_\epsilon(\boldsymbol{x}_p^h - \boldsymbol{x}_q^h) \pm \eta_\epsilon(\boldsymbol{x}_p^h + \boldsymbol{x}_q^h)). \quad (31)$$

The positive sign between the two kernel functions applies for homogeneous Neumann boundary conditions, whereas the negative sign is to be used in the case of homogeneous Dirichlet boundary conditions. The method of images is restricted to the case of homogeneous boundary conditions. For inhomogeneous boundary conditions, the particle strengths need to be adjusted in the vicinity of the boundary.[123]

7.3. *Comparison of PSE and RW*

The accuracy of the RW and PSE methods is illustrated by using a benchmark case of isotropic homogeneous diffusion on the 1D ($d = 1$) ray $\Omega = [0, \infty)$, subject to the following initial and boundary conditions:

$$\begin{cases} u(x, t = 0) = u_0(x) = xe^{-x^2} & x \in [0, \infty), \quad t = 0 \\ u(x = 0, t) = 0 & x = 0, \quad 0 < t \le T. \end{cases} \quad (32)$$

The exact analytic solution of this problem is

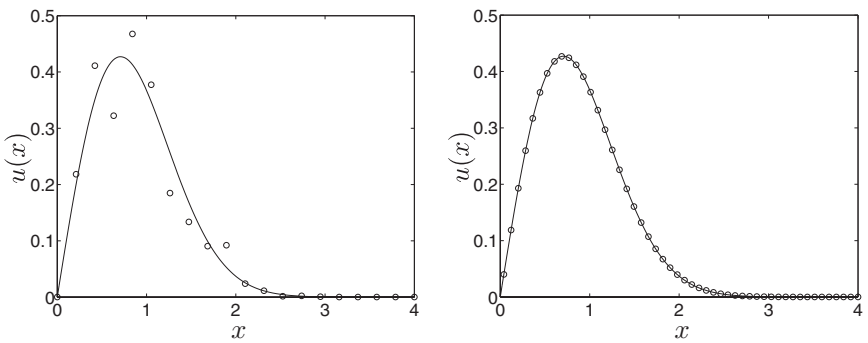
$$u_{\mathrm{ex}}(x, t) = \frac{x}{(1 + 4Dt)^{3/2}} e^{-x^2/(1+4Dt)}. \quad (33)$$

Both RW and PSE simulations of this benchmark case are performed with varying numbers of particles in order to study spatial convergence.[49] The boundary condition at $x = 0$ is satisfied using the method of images as introduced in Sec. 7.2.2.

Figure 8 shows the RW and PSE solutions in comparison to the exact solution at a final time of $T = 10$ for $N = 50$ particles and a diffusion constant of $D = 10^{-4}$. The accuracy of the simulations for different numbers of particles is assessed by computing the final L_2 error

$$L_2 = \left[\frac{1}{N} \sum_{p=1}^{N} (u_{\text{ex}}(x_p, T) - u(x_p, T))^2 \right]^{1/2} \tag{34}$$

for each N. The resulting convergence curves are shown in Fig. 9. For RW, we observe the characteristic slow convergence of $\mathcal{O}(1/\sqrt{N})$.[68] For PSE, a convergence of $\mathcal{O}(1/N^2)$ is observed, in agreement with the employed second-order kernel function. Below an error of 10^{-6}, machine precision is reached. It can be seen that the error of a PSE simulation



(a) RW (b) PSE

Fig. 8. Comparison of (a) RW and (b) PSE solutions of the benchmark case. The solutions at time $T = 10$ are shown (circles) along with the exact analytic solution [solid line; Eq. (33)]. For both methods, $N = 50$ particles, a time step of $\delta t = 0.1$, and $D = 10^{-4}$ are used. The RW solution is sampled in 20 intervals of width $\delta x = 0.2$. For the PSE, a core size of $\epsilon = h$ is used.

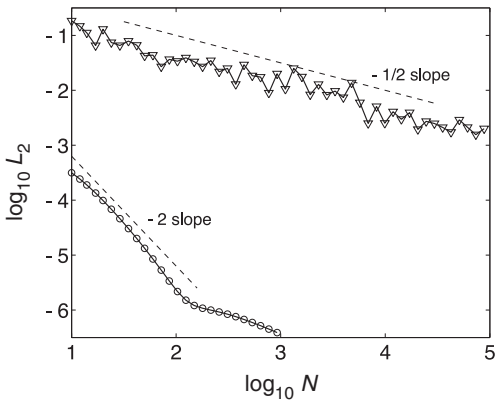


Fig. 9. Convergence curves for RW and PSE. The L_2 error versus the number of particles for the RW (triangles) and the PSE (circles) solutions of the benchmark case at time $T = 10$ are shown. For both methods, a time step of $\delta t = 0.1$ and $D = 10^{-4}$ are used. The RW solution is sampled in 20 intervals of width $\delta x = 0.2$; and for the PSE, a core size of $\epsilon = h$ is used. The machine epsilon of the computer is 10^{-6}.

is several orders of magnitude lower than the one of the corresponding RW simulation with the same number of particles. Using only 100 particles, PSE is already close to machine precision. It is evident from these results that large numbers of particles are necessary to achieve reasonable accuracy using RW in complex-shaped domains.

8. Reaction-Diffusion Processes

The reviewed particle methods for the simulation of diffusion processes can be extended to account for spatially resolved (bio)chemical reactions using either deterministic reaction kinetics or stochastic models. In the following, we consider reaction-diffusion systems governed by equations of the Fisher–KPP[124,125] type:

$$\frac{\partial u_i}{\partial t} - \nabla \cdot (\boldsymbol{D}_i \nabla u_i) = f_i(\boldsymbol{u}). \tag{35}$$

The concentration vector $\boldsymbol{u}$ contains one entry u_i per chemical species i, and the diffusion tensors $\boldsymbol{D}_i$ are allowed to vary among species and

in space. All chemical reactions are described by the source terms f_i.

8.1. *Previous Approaches and Applications in Biology*

Coupled reaction-diffusion systems exhibit nontrivial stability properties that can give rise to the formation of stable concentration patterns called Turing patterns,[1] or traveling waves called Fisher waves.[126]

Twenty years after the seminal work of Turing,[1] Gierer and Meinhardt used reaction-diffusion systems to formulate their theory of pattern formation in biology.[127] They introduced the Gierer–Meinhardt model, which has become one of the most widely used pattern formation models in biology, with later applications also in computer graphics.[128]

The first biological applications of reaction-diffusion models considered morphogenesis,[1] following the idea that reaction-diffusion patterns of growth factors could explain the geometries and shapes found in nature. Computer simulations linking pattern formation to growth and morphogenesis were done using, e.g. hybrid cellular automata–PDE simulations to explain stalk formation and cell differentiation in slime mold.[2]

Simulations of reaction-diffusion patterns on surfaces first considered spherical objects such as globular tumors.[18] Morphogenesis of more complex surfaces was simulated using an FE method to solve the reaction-diffusion equation on triangulated surfaces.[3] This method allowed treating shapes as complex as branched unicellular algae. Moving-mesh FE techniques were later used to directly couple the motion of the boundary to reaction-diffusion patterns on continuously deforming 2D domains.[129] A different approach uses the solution of an interior Poisson problem to evolve the surface shape.[20]

Besides morphogenesis, reaction-diffusion models also have important applications in cell motility,[130] cell modeling,[131] and cell culture pattern formation.[132] Emerging applications also include spatiotemporal simulations of cell signaling pathways.[133] Since the first ODE model of the chemotaxis pathway in *Escherichia coli* was published by Bray *et al.* in 1993,[134] computer simulations have become increasingly more sophisticated in resolving spatial phenomena. A recent model explicitly includes

diffusion of the key signal transduction molecule in the cytoplasmic space.[12] Other examples of reaction-diffusion signaling models include a sporulation control network model[11] and plant shoot meristem simulations.[135]

8.2. *Reaction-Diffusion in Particle Methods*

Extending the simulation scheme outlined in Sec. 7 to reaction-diffusion problems as governed by Eq. (35), all components of the concentration vector $\boldsymbol{u}$ are represented on the same set of computational particles, supporting property vectors $\boldsymbol{\omega}_p^b$.

Evaluating the reaction terms f_i amounts to a purely local exchange of strength among species in the same particle. Reactions are thus evaluated independently for each particle. The rate of exchange between different u_i is directly given by the reaction kinetics that are evaluated using either a deterministic method based on kinetic ODEs or a stochastic method such as Gillespie's SSA algorithm.[61,62] The latter is possible because individual particles constitute homogeneous reaction spaces with no spatial gradients present inside a single particle. The deterministic solver uses the same time integrator as the diffusion part, and is thus restricted by the time step stability limit; while the stochastic solver directly operates on (scaled) molecule numbers, and is used outside of the time integrator's right-hand side.

8.2.1. *An example with moving reaction fronts*

As an illustrative example, we consider the reaction $a + b \rightarrow 2a$ with rate constant k. Both species a and b diffuse with the same isotropic diffusion constant D. We use a combination of PSE[103] and SSA[61,62] to simulate the example system on the surface of a sphere with diameter ℓ. The initial condition is such that one half of the sphere contains only a, and the other half only b. Since a and b are initially unmixed, diffusion is required in order to bring them together and allow the reaction to start. As b is "eaten up" by a, the reaction front separating the two species moves into the region where b initially was and thus forms a traveling Fisher wave, which propagates in the direction orthogonal to the reaction front.

The wave stops as soon as the sphere contains only a and all of b has been consumed. If we identify the concentration of a with $[a] = u$ and normalize the total concentration to 1 everywhere, mass action kinetics gives the reaction term $f(u) = ku(1 - u)$. Inserting this into Eq. (35) yields a nonlinear PDE. Such systems can exhibit bifurcations (cf. Sec. 2), which is the case in the present example as Fisher waves only exist for speeds $s \geq s^* = 2\sqrt{f'(0)} = 2\sqrt{k}$.[136,137] For $s < s^*$, no moving front exists.

For the stochastic simulations, we denote by X_p the total number of molecules contained in particle p, and define an analog to Avogadro's number, viz. M, the number of molecules per unit mass. Gillespie introduced the product $\hbar c$ as the expected number of reactions per unit time. For the binary reaction here, it is $\hbar = X_p^a X_p^b$. The rate constant k and the reaction parameter c are related as $k = MVc$.[61] We interpret c as the probability that two molecules of species a and b react, provided they meet in space and time. This probability is independent of the particle volume.[c]

Figure 10(a) shows the total mass, integrated over the surface of the sphere, of a and b as they evolve in time. The front position is also shown, defined as the location where $[a] = [b] = 0.5$. As long as reactions occur, $[a]$ and $[b]$ change and the wave travels at a more or less constant speed s, given by the slope of the dashed curve in Fig. 10(a). If the dimensionless front speed is plotted against the dimensionless reaction parameter, the curves for different diffusion constants D collapse as shown in Fig. 10(b). We also observe that the theoretical scaling[136,137] is well approximated, particularly if the reaction is significantly faster than the diffusion.

9. Conclusions

Spatiotemporal models and numerical computer simulations are of rapidly growing importance in almost all areas of biology. Besides computational data analysis, including image processing, gene and protein sequence analysis, clustering, and machine learning, modeling and simulation are

[c] The probability of an encounter to occur, on the other hand, does depend on the particle volume. This is, however, already accounted for in $\hbar$, as the number of molecules X_p per particle is an extensive property.

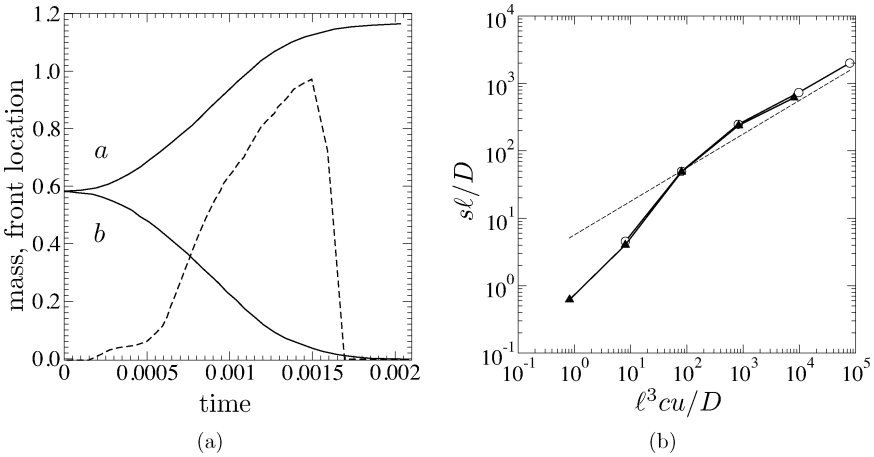


Fig. 10. (a) Evolution of the total mass of *a* and *b* (solid lines) for $M = 10$ molecules per unit mass. See text for problem description. The location of the wave front is shown by the dashed curve. The reaction front moves into the region of *b* until all of *b* is consumed. (b) Dependence of the front speed *s* on the reaction parameter *c*. In dimensionless representation, the two curves for $D = 0.1$ (open circles) and $D = 1.0$ (filled triangles) collapse to one. The theoretical scaling[136,137] is indicated by the dashed line.

important tools in modern biology. "Virtual experiments" *in silico* enable control over all variables and influence factors, allowing us to dissect biological systems, formulate physical models for their constituents, and disentangle coupled processes that are not separable in classical experiments. Moreover, spatiotemporal modeling makes accessible time and length scales unreachable by experiments. This allows studying systems as small as the atoms in an individual protein or as large as complete ecosystems.

The characteristic properties of biological systems frequently complicate the formulation and simulation of spatiotemporal models. Regulation mechanisms, geometric shape complexity, nonlinearity, and couplings across scales increase model complexity, and require powerful and flexible numerical methods as well as efficient high-performance software implementations on supercomputers.[26,32]

We have surveyed some of the most common modeling techniques and categorized them along the dimensions of phenomenological vs.

physical, discrete vs. continuous, and deterministic vs. stochastic. We summarized for each of these model classes some of the available numerical simulation methods with pointers to specialized literature. We then motivated the use of point-based particle methods that do not require any connectivity-based discretization and allow simulating both discrete and continuous systems. While their application to discrete systems is standard, the use of continuum particle methods is less widespread. We have reviewed continuum particle methods in more detail and demonstrated their application to diffusion and reaction-diffusion problems. This was intended as an easy-to-follow introduction. Applications to other phenomena such as flows and waves are well documented in the literature.

Notwithstanding the inherent complexity of biological systems, a wealth of methods and tools are available that enable highly accurate and predictive spatiotemporal simulations. At the same time, biology can serve as an important technology driver to stimulate the development of new methods and further advances in numerical analysis, computational science, and high-performance computing. Numerical methods that efficiently handle multi-scale systems[27–31] and topological changes in complex geometries are at the forefront of research in computational science. In parallel, computer algorithms have to be efficient enough to deal with the vast number of degrees of freedom, and software platforms must be available to effectively and robustly implement these algorithms on multi-processor supercomputers.[32] Just as fluid dynamics — in particular, the nonlinear, multi-scale problem of turbulence — has driven the development of numerical methods in the past, the major scientific goal of modeling an entire cell[138] might continue to do so in the future.

References

1. Turing AM. (1952) The chemical basis of morphogenesis. *Phil Trans R Soc Lond B* **237**: 37–72.
2. Marée AFM. (2000) *From Pattern Formation to Morphogenesis.* PhD thesis. University of Utrecht, Utrecht, The Netherlands.
3. Harrison LG, Wehner S, Holloway DM. (2001) Complex morphogenesis of surfaces: theory and experiment on coupling of reaction-diffusion patterning to growth. *Faraday Discuss* **120**: 277–94.

4. Kaandorp JA, Sloot PMA, Merks RMH, *et al.* (2005) Morphogenesis of the branching reef coral *Madracis mirabilis*. *Proc R Soc Lond B Biol Sci* **272**: 127–33.
5. Grant MR, Mostov KE, Tlsty TD, Hunt CA. (2006) Simulating properties of *in vitro* epithelial cell morphogenesis. *PLoS Comput Biol* **2**: e129.
6. Smith AE, Helenius A. (2004) How viruses enter animal cells. *Science* **304**: 237–42.
7. Dimitrov DS. (2004) Virus entry: molecular mechanisms and biomedical applications. *Nat Rev Microbiol* **2**: 109–22.
8. Dinh AT, Theofanous T, Mitragotri S. (2005) A model for intracellular trafficking of adenoviral vectors. *Biophys J* **89**: 1574–88.
9. Greber UF, Way M. (2006) A superhighway to virus infection. *Cell* **124**: 741–54.
10. Marsh M, Helenius A. (2006) Virus entry: open sesame. *Cell* **124**: 729–40.
11. Marwan W. (2003) Theory of time-resolved complementation and its use to explore the sporulation control network in *Physarum polycephalum*. *Genetics* **164**: 105–15.
12. Lipkow K, Andrews SS, Bray D. (2005) Simulated diffusion of phosphorylated CheY through the cytoplasm of *Escherichia coli*. *J Bacteriol* **187**: 45–53.
13. Dayel MJ, Hom EF, Verkman AS. (1999) Diffusion of green fluorescent protein in the aqueous-phase lumen of the endoplasmic reticulum. *Biophys J* **76**: 2843–51.
14. Lippincott-Schwartz J, Snapp E, Kenworthy A. (2001) Studying protein dynamics in living cells. *Nat Rev Mol Cell Biol* **2**: 444–56.
15. Verkman AS. (2002) Solute and macromolecule diffusion in cellular aqueous compartments. *Trends Biochem Sci* **27**: 27–33.
16. Alberts B, Bray D, Johnson A, *et al.* (1997) *Essential Cell Biology*. New York, NY: Garland Publication, Inc.
17. Rauch EM, Millonas MM. (2004) The role of trans-membrane signal transduction in Turing-type cellular pattern formation. *J Theor Biol* **226**: 401–7.
18. Chaplain MAJ, Ganesh M, Graham IG. (2001) Spatio-temporal pattern formation on spherical surfaces: numerical simulation and application to solid tumour growth. *J Math Biol* **42**: 387–423.
19. Jiang Y, Pjesivac-Grbovic J, Cantrell C, Freyer JP. (2005) A multiscale model for avascular tumor growth. *Biophys J* **89**: 3884–94.
20. Macklin P, Lowengrub J. (2005) Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth. *J Comput Phys* **203**: 191–220.
21. Hense BA, Kuttler C, Müller J, *et al.* (2007) Does efficiency sensing unify diffusion and quorum sensing? *Nat Rev Microbiol* **5**: 230–9.
22. Müller J, Kuttler C, Hense BA, *et al.* (2006) Cell-cell communication by quorum sensing and dimension-reduction. *J Math Biol* **53**: 672–702.

23. Slepchenko BM, Schaff JC, Carson JH, Loew LM. (2002) Computational cell biology: spatiotemporal simulation of cellular events. *Annu Rev Biophys Biomol Struct* **31**: 423–41.
24. Fall CP, Marland ES, Wagner JM, Tyson JJ. (2002) *Computational Cell Biology*. Interdisciplinary Applied Mathematics, Vol. 20. New York, NY: Springer.
25. Strogatz SH. (2000) *Nonlinear Dynamics and Chaos*. Boulder, CO: Westview Press.
26. Takahashi K, Yugi K, Hashimoto K, *et al.* (2002) Computational challenges in cell simulation: a software engineering approach. *IEEE Intell Syst* **17**: 64–71.
27. Koumoutsakos P. (2005) Multiscale flow simulations using particles. *Annu Rev Fluid Mech* **37**: 457–87.
28. Mielke A. (2006) *Analysis, Modeling and Simulation of Multiscale Problems*. Heidelberg, Germany: Springer.
29. Alt W, Chaplain M, Griebel M, Lenz J. (2008) *Polymer and Cell Dynamics: Multiscale Modeling and Numerical Simulations*. Basel, Switerland: Birkhäuser.
30. Kovalenko A, Gusarov S, Stepanova M. (2009) *Multiscale Modeling: Concepts, Theories, and Applications*. Boca Raton, Florida: CRC Press.
31. Chauviere A, Preziosi L, Verdier C. (2009) *Cell Mechanics: From Single Scale-based Models to Multiscale Modeling*. Boca Raton, Florida: Chapman & Hall.
32. Sbalzarini IF, Walther JH, Bergdorf M, *et al.* (2006) PPM — a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J Comput Phys* **215**: 566–88.
33. Reynwar BJ, Gregoria I, Harmandaris VA, *et al.* (2007) Aggregation and vesiculation of membrane proteins by curvaturemediated interactions. *Nature* **447**: 461–4.
34. Periole X, Huber T, Marrink SJ, Sakmar TP. (2007) G protein-coupled receptors self-assemble in dynamics simulations of model bilayers. *J Am Chem Soc* **129**: 10126–32.
35. Chu JW, Voth GA. (2006) Coarse-grained modeling of the actin filament derived from atomistic-scale simulations. *Biophys J* **90**: 1572–82.
36. Kashiwagi A, Urabe I, Kaneko K, Yomo T. (2006) Adaptive response of a gene network to environmental changes by fitness-induced attractor selection. *PLoS ONE* **1**: e49.
37. Burlando B. (1993) The fractal geometry of evolution. *J Theor Biol* **163**: 161–72.
38. Newman TJ, Antonovics J, Wilbur HM. (2002) Population dynamics with a refuge: fractal basins and the suppression of chaos. *Theor Popul Biol* **62**: 121–8.
39. Grasman J, Brascamp JW, Van Leeuwen JL, Van Putten B. (2003) The multifractal structure of arterial trees. *J Theor Biol* **220**: 75–82.
40. Smith TG Jr, Marks WB, Lang GD, *et al.* (1989) A fractal analysis of cell images. *J Neurosci Methods* **27**: 173–80.

41. Aon M, Cortassa S. (1994) On the fractal nature of cytoplasm. *FEBS Lett* **344**: 1–4.
42. Lahiri T, Chakrabarti A, Dasgupta AK. (1998) Multilamellar vesicular clusters of phosphatidylcholine and their sensitivity to spectrin: a study by fractal analysis. *J Struct Biol* **123**: 179–86.
43. Liebovitch LS, Scheurle D, Rusek M, Zochowski M. (2001) Fractal methods to analyze ion channel kinetics. *Methods* **24**: 359–75.
44. Li H, Li Y, Zhao H. (1990) Fractal analysis of protein chain conformation. *Int J Biol Macromol* **12**: 6–8.
45. Li H, Chen S, Zhao H. (1991) Fat fractal and multifractals for protein and enzyme surfaces. *Int J Biol Macromol* **13**: 210–6.
46. Saxton MJ. (1994) Anomalous diffusion due to obstacles: a Monte Carlo study. *Biophys J* **66**: 394–401.
47. Saxton MJ. (2001) Anomalous subdiffusion in fluorescence photobleaching recovery: a Monte Carlo study. *Biophys J* **81**: 2226–40.
48. Saxton MJ. (2007) A biological interpretation of transient anomalous subdiffusion. I. Qualitative model. *Biophys J* **92**: 1178–91.
49. Sbalzarini IF, Mezzacasa A, Helenius A, Koumoutsakos P. (2005) Effects of organelle shape on fluorescence recovery after photobleaching. *Biophys J* **89**: 1482–92.
50. Frenkel D, Smit B. (2002) *Understanding Molecular Simulation*: *From Algorithms to Applications*. San Diego, CA: Academic Press.
51. Schrödinger E. (1948) *What Is Life*? *The Physical Aspect of the Living Cell*. Cambridge, UK: Cambridge University Press.
52. Marco E, Wedlich-Soldner R, Li R, *et al*. (2007) Endocytosis optimizes the dynamic localization of membrane proteins that regulate cortical polarity. *Cell* **129**: 411–22.
53. González-Segredo N. (2007) Amphiphilic fluids: mesoscopic modelling and computer simulations. In: Dillon KT (ed.), *Soft Condensed Matter*: *New Research*. New York, NY: Nova Science, pp. 125–56.
54. Farlow SJ. (1993) *Partial Differential Equations for Scientists and Engineers*. New York, NY: Dover Publications.
55. Strauss WA. (2007) *Partial Differential Equations: An Introduction*, 2nd ed. New York, NY: Wiley.
56. Øksendal BK. (2003) *Stochastic Differential Equations*, 6th ed. Berlin, Germany: Springer.
57. Manninen T, Linne ML, Ruohonen K. (2006) Developing Itô stochastic differential equation models for neuronal signal transduction pathways. *Comput Biol Chem* **30**: 280–91.
58. Saarinen A, Linne ML, Yli-Harja O. (2008) Stochastic differential equation model for cerebellar granule cell excitability. *PLoS Comput Biol* **4**: e1000004.

59. Alfonsi A, Cancès E, Turinici G, *et al.* (2005) Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. In: Cancès E, Gerbeau JF (eds.). *ESAIM: Proceedings*, Vol. 14, pp. 1–13.
60. Salis H, Sotiropoulos V, Kaznessis YN. (2006) Multiscale Hy3S: hybrid stochastic simulation for supercomputers. *BMC Bioinformatics* **7**: 93–112.
61. Gillespie DT. (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comput Phys* **22**: 403–34.
62. Gillespie DT. (1977) Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* **81**: 2340–61.
63. Gibson MA, Bruck J. (2000) Efficient exact stochastic simulation of chemical systems with many species and many channels. *J Phys Chem A* **104**: 1876–89.
64. Stundzia A, Lumsden C. (1996) Stochastic simulation of coupled reaction-diffusion processes. *J Comput Phys* **127**: 196–207.
65. Isaacson SA, Peskin CS. (2006) Incorporating diffusion in complex geometries into stochastic chemical kinetics simulations. *SIAM J Sci Comput* **28**: 47–74.
66. Liu JS. (2001) *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. New York, NY: Springer.
67. Rubinstein RY, Kroese DP. (2007) *Simulation and the Monte Carlo Method*, 2nd ed. New York, NY: Wiley.
68. Milizzano F, Saffman PG. (1977) The calculation of large Reynolds number two-dimensional flow using discrete vortices with random walk. *J Comput Phys* **23**: 380–92.
69. Schweitzer F. (2003) *Brownian Agents and Active Particles. On the Emergence of Complex Behavior in the Natural and Social Sciences*. Springer Series in Synergetics. Berlin, Germany, Springer.
70. Gaylord RJ, Nishidate K. (1996) *Modeling Nature: Cellular Automata Simulations with Mathematica*. New York, NY: Springer.
71. Ilachinski A. (2001) *Cellular Automata*. Singapore: World Scientific Publishing.
72. Schiff JL. (2007) *Cellular Automata*: *A Discrete View of the World*. Wiley Series in Discrete Mathematics and Optimization. Hoboken, NJ: Wiley Interscience.
73. Terano T, Kita H, Kaneda T, *et al.* (2005) *Agent-based Simulation*: *From Modeling Methodologies to Real-World Applications*. Tokyo, Japan: Springer.
74. Mosler HJ, Schwarz K, Ammann F, Gutscher H. (2001) Computer simulation as a method of further developing a theory: simulating the elaboration likelihood model (ELM). *Pers Soc Psychol Rev* **5**: 201–15.
75. Kloeden PE, Platen E. (1992) *Numerical Solution of Stochastic Differential Equations*. Stochastic Modeling and Applied Probability. Berlin, Germany: Springer.
76. Higham DJ. (2001) An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Rev* **43**: 525–46.

77. Evans G, Blackledge J, Yardley P. (1999) *Numerical Methods for Partial Differential Equations*. Berlin, Germany: Springer.
78. Grossmann C, Roos HG, Stynes M. (2007) *Numerical Treatment of Partial Differential Equations*. Berlin, Germany: Springer.
79. Smith GD. (1985) *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, 3rd ed. Oxford, UK: Clarendon Press.
80. Reddy JN. (1993) *An Introduction to the Finite Element Method*, 2nd ed. New York, NY: McGraw-Hill.
81. Zienkiewicz OC, Taylor RL. (2005) *The Finite Element Method*, 6th ed. Oxford, UK: Butterworth-Heinemann.
82. Benkhaldoun F, Vilsmeier R. (1999) *Finite Volumes for Complex Applications*, Vols. 1 and 2. Paris, France: Hermes Science Publications.
83. LeVeque RJ. (2002) *Finite Volume Methods for Hyperbolic Problems*. Cambridge, UK: Cambridge University Press.
84. Kincaid DR, Cheney EW. (2001) *Numerical Analysis: Mathematics of Scientific Computing*, 3rd ed. Pacific Grove, CA: Brooks/Cole.
85. Cottet GH, Koumoutsakos P. (2000) *Vortex Methods — Theory and Practice*. New York, NY: Cambridge University Press.
86. Li S, Liu WK. (2004) *Meshfree Particle Methods*. Berlin, Germany: Springer.
87. Liu GR, Gu YT. (2005) *An Introduction to Meshfree Methods and Their Programming*. Berlin, Germany: Springer.
88. de Berg M, van Krefeld M, Overmars M, Schwarzkopf O. (2000) *Computational Geometry: Algorithms and Applications*, 2nd ed. Heidelberg, Germany: Springer.
89. Schumaker LL. (1987) Triangulation methods. In: Chui CK, Schumaker LL, Utreras FI (eds.). *Topics in Multivariate Approximation*. New York, NY: Academic Press, pp. 219–32.
90. Bänsch E, Morin P, Nochetto RH. (2005) A finite element method for surface diffusion: the parametric case. *J Comput Phys* **203**: 321–43.
91. Novak IL, Gao F, Choi YS, *et al.* (2007) Diffusion on a curved surface coupled to diffusion in the volume: application to cell biology. *J Comput Phys* **226**: 1271–90.
92. Haber J, Zeilfelder F, Davydov O, Seidel HP. (2001) Smooth approximation and rendering of large scattered data sets. In: *Proc IEEE Visualization*, pp. 341–7.
93. Sethian JA. (1999) *Level Set Methods and Fast Marching Methods*. Cambridge, UK: Cambridge University Press.
94. Enright D, Fedkiw R, Ferziger J, Mitchell I. (2002) A hybrid particle level set method for improved interface capturing. *J Comput Phys* **183**: 83–116.
95. Hieber SE, Koumoutsakos P. (2005) A Lagrangian particle level set method. *J Comput Phys* **210**: 342–67.
96. Sbalzarini IF, Hayer A, Helenius A, Koumoutsakos P. (2006) Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophys J* **90**: 878–85.

97. Ahusborde E, Gruber R, Azaiez M, Sawley ML. (2007) Physics-conforming constraints-oriented numerical method. *Phys Rev E Stat Nonlin Soft Matter Phys* **75**: 056704.
98. Courant R, Friedrichs K, Lewy H. (1928) Über die partiellen Differenzengleichungen der mathematischen Physik. *Math Ann* **100**: 32–74.
99. Courant R, Friedrichs K, Lewy H. (1967) On the partial difference equations of mathematical physics. *IBM Res Dev* **11**: 215–34.
100. Bergdorf M, Koumoutsakos P. (2006) A Lagrangian particle-wavelet method. *Multiscale Model Simul* **5**: 980–95.
101. Bergdorf M, Cottet GH, Koumoutsakos P. (2005) Multilevel adaptive particle methods for convection-diffusion equations. *Multiscale Model Simul* 4: 328–57.
102. Haber S. (1967) Midpoint quadrature formulas. *Math Comput* **21**: 719–21.
103. Degond P, Mas-Gallic S. (1989) The weighted particle method for convection-diffusion equations. Part 1: the case of an isotropic viscosity. *Math Comput* **53**: 485–507.
104. Degond P, Mas-Gallic S. (1989) The weighted particle method for convection-diffusion equations. Part 2: the anisotropic case. *Math Comput* **53**: 509–25.
105. Eldredge JD, Leonard A, Colonius T. (2002) A general deterministic treatment of derivatives in particle methods. *J Comput Phys* **180**: 686–709.
106. Harlow FH. (1964) Particle-in-cell computing method for fluid dynamics. *Methods Comput Phys* **3**: 319–43.
107. Hockney RW, Eastwood JW. (1988) *Computer Simulation Using Particles*. Bristol, UK: Institute of Physics Publishing.
108. Trottenberg U, Oosterlee C, Schueller A. (2001) *Multigrid*. San Diego, CA: Academic Press.
109. Monaghan JJ. (1985) Extrapolating B splines for interpolation. *J Comput Phys* **60**: 253–62.
110. Greengard L, Rokhlin V. (1988) The rapid evaluation of potential fields in three dimensions. In: Anderson C, Greengard C (eds.). *Vortex Methods*. Lecture Notes in Mathematics, Vol. 136. Berlin, Germany: Springer-Verlag, pp. 121–41.
111. Verlet L. (1967) Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys Rev* **159**: 98–103.
112. Sutmann G, Stegailov V. (2006) Optimization of neighbor list techniques in liquid matter simulations. *J Mol Liq* **125**: 197–203.
113. Barnes J, Hut P. (1986) A hierarchical O(Nlog$N$) force-calculation algorithm. *Nature* **324**: 446–9.
114. Greengard L, Rokhlin V. (1987) A fast algorithm for particle simulations. *J Comput Phys* **73**: 325–48.
115. Cheng H, Greengard L, Rokhlin V. (1999) A fast adaptive multipole algorithm in three dimensions. *J Comput Phys* **155**: 468–98.

116. Berg H. (1993) *Random Walks in Biology*. Princeton, NJ: Princeton University Press.
117. Brown R. (1828) A brief account of microscopical observations made in the months of June, July, and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *Philos Mag* **4**: 161–73.
118. Einstein A. (1905) Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Ann Phys* **17**: 549–60.
119. Nadler SB Jr. (1992) *Continuum Theory*. Basel, Switzerland: M. Dekker.
120. Chorin AJ. (1973) Numerical study of slightly viscous flow. *J Fluid Mech* **57**: 785–96.
121. Chandrasekhar S. (1943) Stochastic problems in physics and astronomy. *Rev Mod Phys* **15**: 1–89.
122. Van Kampen NG. (2001) *Stochastic Processes in Physics and Chemistry*, 2nd ed. Amsterdam, The Netherlands: North Holland.
123. Koumoutsakos P, Leonard A, Pépin F. (1994) Boundary conditions for viscous vortex methods. *J Comput Phys* **113**: 52–61.
124. Fisher RA. (1937) The advance of advantageous genes. *Ann Eugen* **7**: 335–69.
125. Kolmogorov AN, Petrovsky IG, Piskunov NS. (1937) Étude de l'équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique. *Bull Univ d'Etat Moscou* (*Bjul Moskowskogo Gos Univ*), Sér Int A **1**: 1–26.
126. Riordan J, Doering CR, ben Avraham D. (1995) Fluctuations and stability of Fisher waves. *Phys Rev Lett* **75**: 565–8.
127. Gierer A, Meinhardt H. (1972) A theory of biological pattern formation. *Kybernetik* **12**: 30–9.
128. Turk G. (1991) Generating textures on arbitrary surfaces using reaction-diffusion. *Comput Graph* **25**: 289–98.
129. Madzvamuse A, Wathen AJ, Maini PK. (2003) A moving grid finite element method applied to a model biological pattern generator. *J Comput Phys* **190**: 478–500.
130. Grimm HP, Verkhovsky AB, Mogilner A, Meister JJ. (2003) Analysis of actin dynamics at the leading edge of crawling cells: implications for the shape of keratocyte lamellipodia. *Eur Biophys J* **32**: 563–77.
131. Plimpton SJ, Slepoy A. (2005) Microbial cell modeling via reacting diffusive particles. *J Phys Conf Ser* **16**: 305–9.
132. Miura T, Maini PK. (2004) Speed of pattern appearance in reaction-diffusion models: implications in the pattern formation of limb bud mesenchyme cells. *Bull Math Biol* **66**: 627–49.
133. Bhalla US. (2004) Models of cell signaling pathways. *Curr Opin Genet Dev* **14**: 375–81.

134. Bray D, Bourret RB, Simon MI. (1993) Computer simulation of the phosphorylation cascade controlling bacterial chemotaxis. *Mol Biol Cell* **4**: 469–82.
135. Jönsson H, Heisler M, Reddy GV, *et al.* (2005) Modeling the organization of the WUSCHEL expression domain in the shoot apical meristem. *Bioinformatics* **21**: i232–40.
136. Benguria RD, Depassier MC. (1996) Speed of fronts of the reaction-diffusion equation. *Phys Rev Lett* **77**: 1171–3.
137. Berestycki H, Hamel F, Nadirashvili N. (2005) The speed of propagation for KPP type problems. I: periodic framework. *J Eur Math Soc* **7**: 173–213.
138. The 2020 Science Group. (2005) *Towards 2020 Science*. Cambridge, UK: Microsoft Research Cambridge.