Spatiotemporal Modeling and Simulation

Ivo F. Sbalzarini[1]

July 13, 2021

[1]MOSAIC Group, Chair of Scientific Computing for Systems Biology, Faculty of Computer Science, TU Dresden; and: Center for Systems Biology Dresden, Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany, (`http://mosaic.mpi-cbg.de`). E-mail: ivos@mpi-cbg.de

Contents

Chapter 1

Introduction

In this chapter:

- What is spatiotemporal modeling and simulation?
- Why and where does modeling and simulation make sense in biology?
- What makes biological systems so special?
- Which classes of modeling techniques exist?

Learning goals:

- Know when to use modeling and simulation in biology
- Know about the limitations and issues
- Know the specific characteristics of biological systems
- Know the differences between the four realms of models

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 the 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* which would not be possible in reality. This greatly expands the range of possible perturbations and observations. Computational experiments (i.e. modeling and simulation) are indicated whenever:

1. the complexity of the system prohibits manual analysis
2. the time or length scales of interest cannot be reached in experiments. Examples include molecular dynamics studies in structural biology where one is interested in Ångströms and picoseconds, or studies in ecology or evolutionary biology where the time scales can be millions of years.
3. certain variables are not observable or not controllable experimentally. In computational 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.
4. ethical considerations prohibit experiments. Computational models do not involve living beings, thus enabling experiments that would be unethical in reality.

Although in this lecture 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 hetero-dimerization 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 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*.
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 example, of little value to know the total electric current density in the whole brain. We need to know *where* the source is. These examples extend across all scales of biological systems. From the above-mentioned predator-prey interactions in ecosystems over morphogenesis and intracellular processes to single molecules – think for example

of conformational changes in proteins. Examples on the intracellular level include virus entry and transport, intracellular signaling, the diffusion of proteins in the various cellular compartments, 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. Another example is found in the role of trans-membrane signaling during morphogenesis. Differences in protein diffusion constants are not large enough to produce *Turing patterns* (discussed in Chap. 7), and the slow transport across inter-compartment membranes is essential. Examples of spatiotemporal processes on the multi-cellular level include tumor growth, and cell-cell signaling, including phenomena such as bacterial quorum sensing, the microscopic mechanism underlying the macroscopic phenomenon of bioluminescence in certain squid. Given the wide-spread 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 include E-Cell, MCell, and the Virtual Cell.

1.1 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, 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:

- Many degrees of freedom (infinitely many in the continuum limit)
- Regulated
- Delineated by complex shapes
- Nonlinear
- Coupled across scales and subsystems
- Plastic over time (time-varying dynamics)
- Non-equilibrium

Due to these properties, biological systems challenge existing methods in modeling and simulation and require state-of-the-art techniques.

1.1.1 Degrees of freedom

The *large number of 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. 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 degrees of freedom. 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 degrees of freedom (see Chap. 5). Only these are then tracked in the simulation, approximating the behavior of the full, infinite system. Discretizations must also be *consistent*, i.e., the discretized system has to converge to the full system if the number of degrees of freedom goes to infinity.
The number of degrees of freedom of a model is also called the *dimensionality* of the model. This is not to be confused with the dimensionality of the space in which the real system lives, which is typically 2, 3, or 4. The dimensionality of the model is the dimension of the state space of the model and is given by the number of degrees of freedom the model has.
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 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, complex (i.e., many degrees of freedom) systems such as patches of lipid bilayers with embedded proteins, or actin filaments. Coarse graining thus allows extending the capabilities of molecular simulations to time and length scales of biological interest.

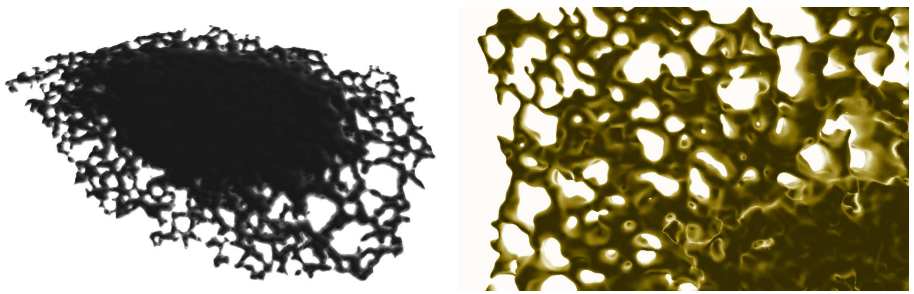
1.1.2 Regulation

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. In addition to such indirect regulation mediated by bi-stability 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. Taking all these effects into account clearly presents a grand challenge to simulation models because many of the hypothetical regulatory mechanisms are still unknown or poorly characterized.

1.1.3 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 pre-fractal structures of taxonomic and phylogenetic trees, regions of stable population growth in ecosystems, pneumonal and arterial trees, the shapes of neurons, the cytoplasmic space, clusters of intracellular vesicles, electric currents through ion channels in cell membranes, protein chain conformations, and protein structures. 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. 1.1) may appear anomalous, even if the underlying molecular diffusion is normal.



(a) (b)

Figure 1.1: (a) Shaded view of a 3D computer reconstruction of the geometry of an Endoplasmic Reticulum (ER) of a live cell (Sbalzarini et al., Biophys. J., 89, 2005). (b) Close-up of a reconstructed ER, illustrating the geometric complexity of this intracellular structure.

1.1.4 Nonlinearity

Common biological phenomena such as interference, cooperation, or 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 by the fact that common computational methods such as normal mode analysis, Fourier transforms, or the superposition principle (see Sec. 11.3), break down in nonlinear systems, because a nonlinear system is not equal to the sum of its parts.

1.1.5 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, including bioluminescence due to bacterial quorum sensing 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, which imposes 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 simulating a living cell, this would mean to resolve 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) simulations currently done consider about 10^{10} atoms over 1 nanosecond. In order to model a complete cell we would thus need a simulation about 100 000 times larger, running over a million-fold 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 decades to come. But 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 and formulations at the appropriate level of detail.

1.1.6 Temporal plasticity

While the analysis of 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. The dynamics of many processes in cells, for example, 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.

1.1.7 Non-equilibrium

According to the second law of thermodynamics, entropy can only increase. Life evades this decay by feeding on negative entropy (Schrödinger, 1948). 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 does, hence, 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 non-equilibrium processes such as vesicular recycling. Due to our incomplete knowledge of the theoretical foundations of non-equilibrium processes, they are much harder to understand. Transient computer simulations are often the sole method available for their study.

1.2 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 – physical, discrete – continuous, and deterministic – stochastic. The three axes are independent and all combinations are possible. Depending on the chosen system description, different modeling techniques are available. Figure 1.2 gives an overview of the most frequently used ones as well as examples of dynamic systems that could be described with them.

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

Figure 1.2: 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.

1.2.1 Phenomenological vs. physical models

Phenomenological models reproduce or approximate the overall behavior of the 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 reproduce the mechanistic functioning of the system bottom-up. 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.

1.2.2 Discrete vs. continuous models

The duality discrete – continuous relates to the spatial resolution of the model. In a *discrete model*, each constituent of the system is explicitly accounted for as an individual entity. Examples include MD simulations, where the position and velocity of each atom is 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, or charge density (see Sec. 2.2).
Continuous deterministic models are characterized by smoothly varying 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 (PDE), 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 (see Chap. 6), advection (see Chap. 8), flow (see Chap. 9), and waves (see Chap. 10).
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, individuals in an ecosystem, or atoms in a molecule. 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.

1.2.3 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, or 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 (SDE). These are PDEs with stochastic terms that can be used to model probabilistic processes such as the spread of epidemics, neuronal signal transduction, 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.

Chapter 2

Modeling Dynamics

In this chapter:

- Definitions of terms
- Continuity and the relation between microscopic and macroscopic models
- How to determine independent dynamic variables for any system and ensure the model represents the correct system
- How to model multiple time scales
- An 8-step recipe to write dynamic models

Learning goals:

- Know the continuity assumption and its limits of validity
- Be able to check the consistency of an equation using dimensional analysis
- Be able to find independent dimensionless groupings for a given system
- Use dimensional analysis to check the validity of a model or experiment
- Be able to distinguish and appropriately model slow, relevant, and fast time scales
- Be able to follow the 8 steps of modeling to find the governing eqution of a dynamical system
- Draw, read, and interpret causality diagrams

Let us start by looking at how an unknown biological system is approached and how its dynamic behavior can be modeled in equations that can then be simulated in the computer. After defining some basic terms, we will see how one determines the important variables in a system (also a key factor in experimental design) and determines dependencies between variables. Finally, we will see an easy to follow recipe that leads to a mathematical model in 8 steps. For now, we focus on modeling the dynamics in time only. Spatial resolution will be added in Chap. 4.

2.1 Definitions

We start by introducing some basic definitions that will be used throughout the course:

System "A system is a potential source of data."
A system has:

- a boundary. We can clearly state what is inside (belongs to the system) and what is outside.
- inputs. The environment influencing the system
- outputs. The system influencing the environment

Experiment "The process of extracting data from a system."
An experiment does:

- observe the trajectory of the system outputs (observation experiment)
- apply a defined change to the system inputs and record the reaction in the outputs (perturbation experiment)

The problem in most real-world systems is that not all inputs are controllable or not all outputs observable.

Model "A model for a system, and a specific experiment, is anything to which the experiment can be applied in order to answer questions about the system."

- each model is a system (hierarchy of models)
- models need not be mathematical
- models are only defined on the tuple (system, experiment). No model is valid for all experiments and all models are valid for the null experiment.
- It is nonsense to state that a model is "valid" or "invalid" for a given system.

Simulation "A simulation is an experiment performed on a model."

- need not be computational
- beware not be use the simulation to make an experiment for which the model is not valid, i.e. outside its "experimental frame"!

- Model description and experiment description should be modularly separated.

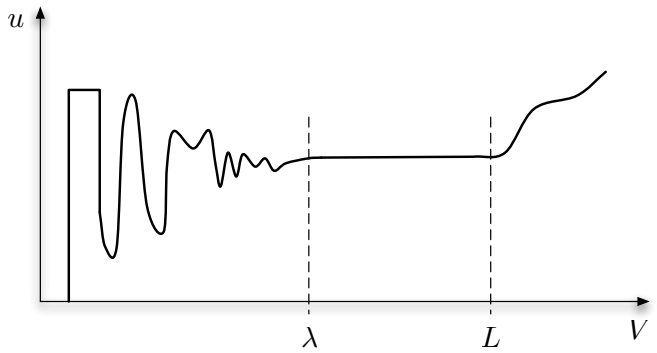
Modeling "The process of organizing knowledge about a given system."

2.2 The Continuity Assumption

The basic framework for modeling biological systems is given by the fact that ultimately all matter is composed of discrete particles. These can be atoms, molecules, or other "particles" such as individuals in an ecosystem. Particles are modeled by their basic properties:

- the position of the particle in space: $\underline{x}_p(t)$
- the physical properties (attributes) of the particle: $\underline{\omega}(P,t)$. These can be properties such as mass, charge, energy, volume, etc.

Often, there is an abundance of particles filling the space and we can abstract from individual particles to continuous "fields". A field can be modeled as a continuous and differentiable function in space and it is the result of an averaging over the physical properties of the underlying particles. This averages out the random (often thermal) fluctuations in the case of large numbers of particles. Fields thus define quantities such as the density (concentration) of particles as a function of space. The concentration of molecules can 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. These definitions are, however, only valid in special cases and it is important to keep in mind that they are a model approximation to the inherently discrete nature of things. As we increase the size V of the averaging volume that we use for "measuring" the concentration, we observe that the averaged field quantity u varies something like this:



We can define two characteristic length scales of the system:
λ: the continuum limit (related to the mean free path of the particles)
L: the length scale of field variations.

Below λ, the averaging volume is too small compared to the particle density and entry/exit of individual particles causes the average to strongly fluctuate. 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. Above L, macroscopic gradients become apparent (or we are interested in resolving/measuring them) if the field is not constant in space, again causing the average to change. The value of L is related to the so-called Kuramoto length, or the notion of well-mixedness of the system. The definition of a continuous concentration value only makes sense for length scales above λ. Spatial variations can be neglected only below L. The region between λ and L hence is the domain of validity of non-spatial dynamic models, such as pathway models and chemical reaction networks. Above L, we need to take space into account. For any system, we can define the dimensionless ratio $\mathrm{Kn} = \lambda/L$, called the *Knudsen number*. Only systems for which $\mathrm{Kn} \ll 1$ allow capturing the field variations by continuous models. For systems with $\mathrm{Kn} > 1$, spatial variations become important or apparent *before* the continuity limit is reached, hence every particle counts and must be modeled as a separate, discrete entity.

Furthermore, we can distinguish two types of properties: If the value between λ and L depends on V, the property is called *extensive*, else *intensive*. Examples of extensive properties include mass, charge, and heat. Extensive and intensive properties always come in pairs and the corresponding intensive properties in the above example are concentration, charge density, and temperature.

Temperature is an intensive property. If one liter of water at 20℃ is divided into two half-liter glasses, the water in each of the two glasses will still have a temperature of 20℃, even though the volume is halved. The temperature of the water is independent of the volume of water. Neither of the two half liters of water however has the same mass as the original liter. Mass hence is an extensive property.

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 as described in Sec. 2.3.

Example 2.2.1 (Some examples of common field variables)**.**

1. density/concentration

$$\begin{aligned} \rho &= \frac{n_p m_p}{V} = \lim_{V\to 0} \frac{\Delta m}{\Delta V} \\ [\rho] &= \frac{M}{L^3} \quad \Rightarrow \quad \frac{\mathrm{kg}}{\mathrm{m}^3} \end{aligned}$$

2. pressure (on surfaces)

$$p = \lim_{A\to 0} \frac{\Delta \underline{F} \cdot \underline{n}}{A}$$
$$[p] = \frac{M}{LT^2} \quad \Rightarrow \quad \mathrm{Pa} = \frac{\mathrm{N}}{\mathrm{m}^2}$$

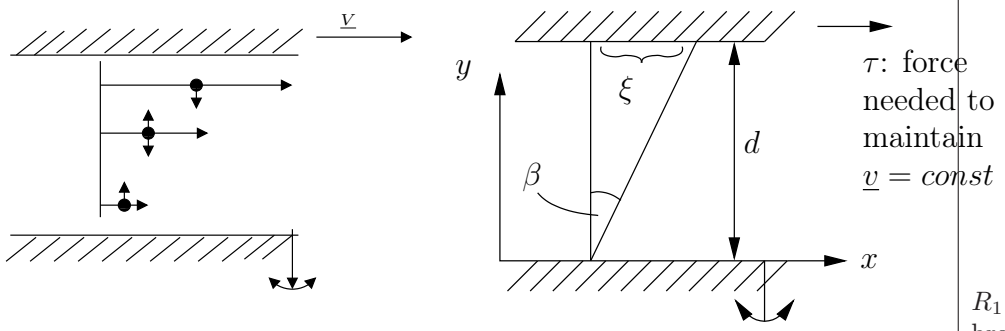
3. temperature T

$$[T] = \Theta \quad \Rightarrow \quad \mathrm{K}$$

4. velocity $\underline{v}$

$$[v] = \frac{L}{T} \quad \Rightarrow \quad \frac{\mathrm{m}}{\mathrm{s}}$$

5. viscosity



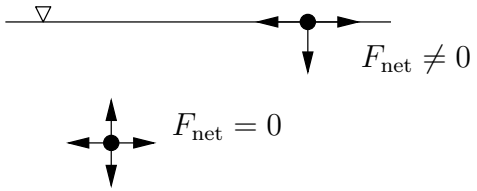
$\xi \approx vt$
$\beta \approx \frac{\xi}{d}$

Momentum exchange due to inner friction (particles moving up lack momentum and need to be accelerated):

solids: $\tau = G \cdot \beta$; G: sheer modulus
fluids: $\tau = \mu \frac{d\beta}{dt}$
$\cong \mu \frac{d}{dt}\left(\frac{vt}{d}\right) = \mu \frac{v}{d} = \mu \frac{dv}{dy}$
μ : dynamic viscosity
$[\mu] = \frac{M}{LT} \quad \Rightarrow \quad \frac{\mathrm{Ns}}{\mathrm{m}^2}$
$\mu\,(\text{water}) \approx 10^{-3} \frac{\mathrm{kg}}{\mathrm{ms}} > \mu\,(\text{air}) \approx 1.8 \cdot 10^{-5} \frac{\mathrm{kg}}{\mathrm{ms}}$

typically: $\mu \downarrow$ if $T \uparrow$

6. surface tension



$$\sigma = \frac{''E''}{L^2} = \frac{''F''}{L} \Rightarrow \frac{\mathrm{N}}{\mathrm{m}}$$

$$\text{work} = F \cdot \Delta x \quad \text{(energy)}$$
$$\text{area} = \Delta x \Delta y$$
$$\sigma = \frac{F \Delta x}{\Delta x \Delta y} = \frac{F}{\Delta y}$$

for curved liquid surfaces:

$$\Delta p = \sigma \left(\frac{1}{R_1} + \frac{1}{R_2}\right) \qquad \text{(Laplace)}$$

R_1, R_2: curvature radii in the two principal directions $\Rightarrow$ important for cell membranes.

Question 1. What is Δp for a spherical droplet of radius R?

Question 2. Estimate the maximum pressure difference across a cell membrane for $\sigma \approx 0.03 \frac{\mathrm{N}}{\mathrm{m}}$, knowing that membranes can not assume curvature radii of less than 20 nm.

2.3 Macroscopic vs. Microscopic View

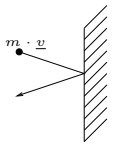
The two views correspond to different levels of model resolution:
macroscopic $\rightarrow$ continuum approximation using field functions
microscopic $\rightarrow$ account for and track the individual particles (for example in molecular dynamics simulations).

The macroscopic view is the result of some averaging procedure and requires separation of scales in order to be valid (Kn $\ll 1$).

Example 2.3.1.

1. Temperature $\frac{3}{2}k_BT = \left\langle \underbrace{\frac{1}{2}m_p\underline{v}_p \cdot \underline{v}_p}_{\text{kinetic energy}} \right\rangle_P$
2. Density $\rho = \langle n_p m_p \rangle_V$
3. Pressure

$$\frac{\text{normal force}}{\text{unit area}} = \langle \underline{F} \cdot \underline{n} \rangle_A$$



Notice that all quantities inside the averaging brackets are extensive and all averages are intensive. The quantities carried by the particles are thus in general extensive, whereas field quantities are intensive.

2.4 Dimensionality Analysis

Dimensionality analysis is a simple yet powerful technique to analyze a system without prior knowledge. It can be used to:

- estimate orders of magnitude of quantities
- find out about correlations between variables
- check similitude between model and system
- check consistency of equations.

The governing equations do not need to be known, but we need to know physics. By selecting the dimensions of the problem we postulate certain physical processes. Kolomogrov (1914) derived his famous turbulence results using dimensionality analysis.

Basic idea:

dimensions	$\neq$	units
$(M, L, T, \ldots)$		(m, kg, J, ...)
		(in, lb, btu, ...)
	independent	

Theorem 1. In classical physics, all dimensions can be expressed as power series of six independent dimensions. While there are many possible choices of six, a frequent choice is: mass (M), length (L), time (T), temperature (Θ), charge (C), and luminosity (J).

Note that in relativistic or quantum physics, there is only a single independent dimension, which is usually taken to be Energy.
The notation $[\cdot]$ means "dimensions of", thus $[\mu] = \frac{M}{LT}$.
All dimensions are power series of the independent dimensions (basis of the dimension space). Examples:

Force=ML/T^2 (Newton)
Energy=Force$\cdot L = ML^2/T^2$

Theorem 2. All equations derived from physical principles are dimensionally homogeneous

$\Rightarrow$ become familiar with an unknown governing equation

Question 3 (dimensional analysis of the Navier-Stokes equation for incompressible flows)**.** Determine the dimension of each term in the Navier-Stokes equation:

$$\rho\frac{\partial v}{\partial t} + \rho v\frac{\partial v}{\partial x} = -\frac{\partial p}{\partial x} + \mu\frac{\partial^2 v}{\partial x^2}\,.$$

The symbols are: x for position, t for time, v for the flow velocity, μ is the dynamic viscosity of the fluid, and p the pressure.

Definition 2.4.1 (dimensionless grouping)**.** A dimensionless grouping is an algebraic combination of variables that has no dimension (Example: Knudsen number).

Theorem 3. Let n be the number of *independent* dimensions of a model and p the number of variables (parameters). We need $p - n$ dimensionless groupings to completely describe the dynamics of the model (Buckingham Π Theorem, 1915).

Example 2.4.1. Let's say we have a model of a moving object. The $p = 4$ model variables are the mass of the object, the force acting on it, the time during which the force is active, and the velocity of the object. This model thus involves the $n = 3$ independent dimensions M, L, and T. There is nothing that would depend on charge, temperature, or resistance. We can completely describe the dynamics of the model using $p - n = 1$ dimensionless grouping. Since this is the *only* independent variable in this model, it must be constant. (In this case the dimensionless grouping would immediately allow us to derive Newton's second law of motion!)

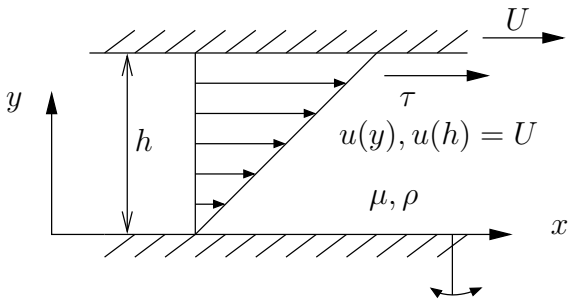
But how to find the dimensionless groupings that form the minimum set of variables needed to describe a model?

2.4.1 E. S. Taylor's Method (1974)

Taylor's method is a recipe to find dimensionless groupings:

1. determine the number of groupings needed using the Π theorem.
2. write a matrix with columns representing (independent) dimensions and rows variables.[1]
3. choose the simplest-looking column and eliminate all but one entries by Gauss elimination with the simplest row that has non-zero entries in this column.
4. If only one variable is left for a certain dimension, delete it from the matrix, along with the corresponding dimension.
5. Go to (3) until all entries are zero.

Example 2.4.2 (Couette flow wall force)**.**
Models the shear stress on the plasma membrane of an erythrocyte moving along the wall of a blood vessel.



1. independent dimensions: M, L, T ($n = 3$)
   variables: h, U, μ, ρ, τ
   ⇒ We need $5 - 3 = 2$ dimensionless groupings.

2.

	M	L	T	
h	0	1	0	
U	0	1	−1	
μ	1	−1	−1	
ρ	1	−3	0	
τ	1	−1	−2	(force/area)

[1] product Ansatz: $[R] = \prod_{i=1}^{N-1} [v_i]^{\alpha_i}$ (Note: α_i is one entry in the matrix)
⇒ result is a power product of other variables. (proof by Buckingham)

3. simplest column: $1^{\text{st}}(M)$ ⇒ eliminate using row ρ (simplest)

		M ⇓	L	T
	h	0	1	0
	U	0	1	−1
	μ/ρ	0	2	−1
⇒	ρ	1	−3	0
	τ/ρ	0	2	−2

4. ρ is the only variable having a non-zero entry left in the column M ⇒ delete row and column.

	L	T
h	1	0
U	1	−1
μ/ρ	2	−1
τ/ρ	2	−2

5. Go to (3). The next simplest column is the 3^{rd}(T) ⇒ eliminate with U (simplest row that has a non-zero entry in T)

		L	T ⇓
	h	1	0
⇒	U	1	−1
	$\mu/(\rho U)$	1	0
	$\tau/(\rho U^2)$	0	0

Delete column T and row U as there is only one non-zero entry left. The only column left is L ⇒ eliminate with h (simplest row).

		L ⇓
	h	1
⇒	$\mu/(\rho U h)$	0
	$\tau/(\rho U^2)$	0

Delete the column L and the row h. What remains are the two dimensionless groupings for this problem:

$$\Pi_1 = \frac{\mu}{\rho U h}; \qquad \Pi_2 = \frac{\tau}{\rho U^2}\,.$$

Since these are the only two parameters that are independent (all others are functionally dependent on each other and the form of the functional depen-

dence follows from the dimensionless groupings!), they must be related as:

$$\frac{\mu}{\rho U h} = f\left(\frac{\tau}{\rho U^2}\right) .$$

The form of the functional dependence can, e.g., be determined experimentally, since we know now how to control the variables.

When the governing equations are written in nondimensional form, the dimensionless groupings of the problem appear as coefficients.

2.4.2 Common dimensionless groupings

There are many pre-defined dimensionless groupings with intuitive meaning. These can be used to guide the elimination process. The most important examples include:

- Reynolds number $\mathrm{Re} = \frac{\rho U L}{\mu}$ "viscous stress"
- Mach number $\mathrm{Ma} = \frac{U}{a}$ "speed"
- Stronhal number $\mathrm{St} = \frac{\omega L}{U}$ "frequency"
- Weber number $\mathrm{We} = \frac{\rho U^2 L}{\sigma}$ "surface tension"
- Drag coefficient $C_D = \frac{\rho U^2}{\tau}$

It is not surprising that the two dimensionless groupings we found in the previous example were the Reynolds number and the Drag coefficient (well, their inverse), as the example dealt with viscous forces in a fluid flow.

2.5 Dynamic Similitude

Often, the model differs from the real system in certain parameters. When and how can we make sure that the dynamic behavior of the model is comparable to the one of the real system? We know that the dynamics are only goverend by the dimensionless groupings. If they have the same values for model (or simulation) and the system (or experiment), then the two dynamics are the same. This is true even if individual variables have different values. This is a very important principle to establish model–system correspondence.

Dynamic similitude is reached when all dimensionless groupings of model and reality match in value (use the n degrees of freedom to compensate model scaling).

2.6 Slow and Fast Time Scales

So far we have not included dynamics in our modeling framework. The relations between dimensionless groupings found by dimensional analysis are algebraic equations in which time at most appears as a static quantity. We will now explicitly include rates of change of quantities and temporal differences, leading to dynamic models formulated as ordinary differential equations (ODE). ODEs model the dynamics in time, i.e. resolve transients, but not in space. These models are valid on length scales between λ and L. Later on, we will then extend our framework to include also spatial resolution, which then leads to partial differential equations that include time and space. We start our discussion of dynamics by looking at different time scales.
Models should only involve time scales of similar order. Otherwise, efficient numerical simulations are not possible because the step size limit for explicit numerical integration schemes is limited by the fastest dynamics.
We distinguish three types of dynamics:

1. relevant ⇒ modeled as differential equations
2. slow ⇒ modeled as constants
3. fast ⇒ modeled as algebraic terms

"Relevance" is defined by the variables of interest, e.g. the quantity based on which the model will be compared to experiments.
No systematic rules exist for classifying the dynamics of a system. Distinguishing slow, fast, and relevant time scales is mainly a matter of experience and constitutes a good part of the "art of modeling".

Example 2.6.1 (FRAP experiment)**.**
Consider again the FRAP experiment example from the first lecture. The following time scales can be distinguished:

1. relevant: diffusion dynamics of the protein. This is the variable we are interested in.
2. slow: variations in the room temperature; changes in cell shape. Can be modeled as constant.
3. fast: camera shutter dynamics; laser switch-on dynamics. Can be modeled as algebraic functions, such as step functions.

2.6.1 Numerical stability of an explicit time integrator

Consider the model problem $\dot{x} = -ax$ with solution $x(t) = x_0 e^{-at}$. The exponent a defines the time scale of the decay.

An integrator is called stable (asymptotically stable) if all trajectories of a stable system remain bounded for $t \to \infty$. The above model problem is stable for all $a > 0$. In these situations, the numerical integration scheme should have bounded trajectories.
Let's consider the forward Euler scheme as an example. The value at the next time point $t + h$ is computed as:

$$\begin{aligned} x_{t+1} &= x_t - h \cdot a x_t = [1 - ha] x_t \\ &= [1 - ha]^{t+1} \cdot x_0 \,, \end{aligned}$$

where x_0 is the initial condition at $t = 0$. Numerical stability requires that $|x_{t+1}| < C$, which can only hold if $|1 - ha| < 1$. The two cases of the absolute value lead to:

$$\left.\begin{array}{l} (1-ha) > 0 \Rightarrow ha > 0 \\ (1-ha) < 0 \Rightarrow ha < 2 \end{array}\right\} \Rightarrow 0 < h < \frac{2}{a} \,.$$

The upper bound for the admissible time steps is inversely proportional to the exponent a. Fast decay processes require small time steps.
This concept generalizes beyond the Euler scheme. For any system $\dot{x} = Ax$ we can write a linear explicit scheme as $x_{t+1} = Fx_t$ with $A, F \in \mathbb{R}^{n \times n}$.
Numerical stability requires that all Eigenvalues λ of F with $Re(\lambda) < 0$ have $|\lambda| < 1$. The fastest time scale (largest $|\lambda|$) thus dictates the time step limit. Moreover, the ratio between the largest Eigenvalue of F and the smallest Eigenvalue of F should also not be too large. Systems with a large ratio are called "stiff" and are problematic for numerical integration.

2.7 Reservoirs and Flows

A system can only exhibit dynamic behavior if it contains storage/reservoirs (integrators). This is a necessary condition for the existence of dynamics. The levels of these reservoirs are called **state variables**. Examples include:
amount of money (discrete), mass, energy, information, ...
Only extensive quantities can be levels. Each level has an associated intensive quantity, e.g.:

Level	
kinetic Energy	→ Speed
Heat	→ Temperature

If the levels of all reservoirs in a system are known at time t, the future behavior of the system can be predicted. The levels thus describe the state of the system. If a system has m reservoir levels, its state is a point in an m-dimensional vector space. This space is called the *state space* of the system.
Reservoirs are connected by **flows** that can transport the contents from one reservoir to another one, thereby changing the levels of the two reservoirs they connect.

2.8 Modeling Steps

Based on the concept of reservoirs and flows, we present a simple recipe for modeling the dynamics of a system and deriving the corresponding governing ordinary differential equations (ODE). There are no spatial effects for the moment. We will include them later. Following are the steps of the modeling procedure:

1. Define the system boundaries (see definition in Section 2.1) and the inputs and outputs of the system. Be careful not to confuse inputs and outputs with inflows and outflows!

2. Identify the reservoirs of relevant time scales (see Section 2.6) and the corresponding level variables (see Section 2.7).

3. Formulate the (algebraic) equations that describe the flows between the reservoirs. In general, these equations have the form:

$$\text{flow} = f(\text{activating level} - \text{inhibiting level})\,.$$

   They can be found from physics, experiments, or dimensional analysis.

4. Use the conservation laws from physics to formulate the balance equations that govern the dynamics of the reservoir levels:

$$\frac{d}{dt}(\text{level}) = \sum \text{inflows} - \sum \text{outflows}\,.$$

5. Simplify the equations, resolve algebraic parts, non-dimensionalize, normalize (avoid numerical extinction and problems with finite machine precision!), ...

6. Solve the equations (analytically or numerically) to obtain level(t).

7. Identify unknown parameters values from experiments, literature, ...
   $\Rightarrow$ Parameter identification

8. Validate the model on data that have **not** been used for parameter identification.

Example 2.8.1 (FLIP-Experiment in the ER lumen)**.**
FLIP = Fluorescence Loss In Photobleaching. Repeatedly bleach the same part of the ER and monitor the dynamics of fluorescence loss in other parts of the ER.

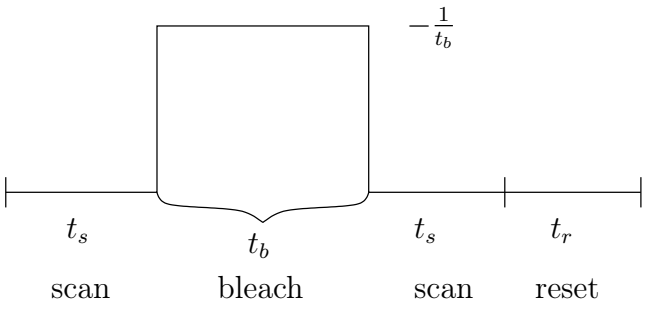
1. system boundary = ER membrane
   u input bleaching rate
   y output fluorescence intensity

2. relevant reservoir: fluorescence content in the ER $\Rightarrow$ level: fluorescent mass $m(t)$

   - too fast:
     - individual bleaching pulses
     - laser on/off dynamics
     - camera shutter dynamics
     - diffusion within a compartment
     - ...
   - too slow:
     - changes in environmental temperature
     - synthesis of new fluorescent proteins
     - cell cycle
     - ...

   These assumptions define (limit) the experimental frame of the model.

3. inflow: none (assumed that synthesis of new proteins is slow)
   outflow: bleaching $\dot{m}_{\text{out}} \Rightarrow \nu \frac{m}{V} \beta(t - k\Delta t)$
   m: activating level, there is no inhibiting level
   ν: bleached volume
   V: total ER volume
   $\beta(t) = \frac{1}{t_b}(\mathcal{H}(t - t_s) - \mathcal{H}(t - t_s - t_b))$ is the algebraic function that models the bleaching pulses (laser and camera dynamics assumed to be fast). This function periodically repeats itself every $\Delta t = 2t_s + t_b + t_r$ in order to model the experimental bleaching cycle:



   We further assume $t_b \ll t_r$ (not accurately reflected in the figure) in order to avoid one bleach to drain the entire mass, hence further limiting the experimental frame of the model.

4. balance equation for conservation of mass:

$$\frac{dm}{dt} = -\frac{\nu}{V}\beta(t - k\Delta t)m \qquad (*)$$

5. simplify → nothing here

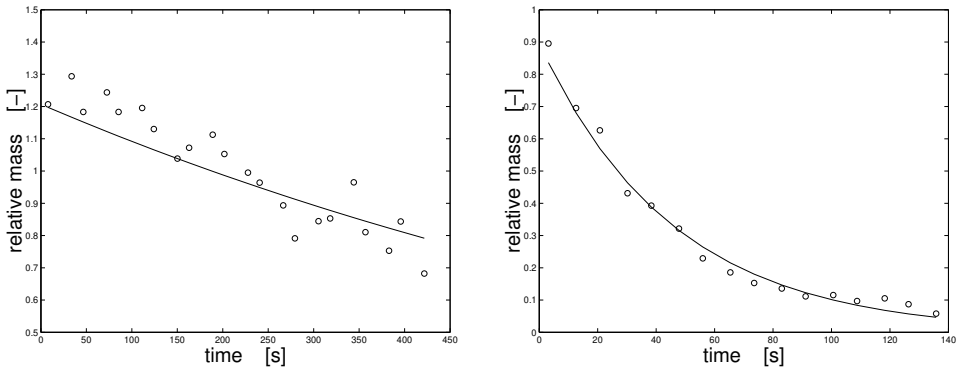
6. Model is a first order homogeneous ODE, so it can be solved analytically:

$$m(t) = m_0 e^{-\alpha\beta t} \qquad (**)$$
$$\alpha = \frac{\nu}{V}$$

7. Parameter identification: determine α by fitting (**) to FLIP measurements.

8. Validate: check that (**) explains the observations from independent experiments.

This model can already be used to evaluate experiments. Figure 2.1 shows the model fits to two FLIP experiments in the lumen of the perinuclear ER in yeast cells. The constant α, the ratio between the bleached volume and the (unknown) total volume of the ER, is determined by fitting the model to FLIP data of the membrane protein *Sec61*. Validation is then done on the soluble pure *GFP*. Comparing the exponential time constants of the fluorescence mass decay for both proteins, we can state that the soluble *GFP* drains about 20 times faster than the membrane-bound *Sec61*. This difference is explainable by the difference in diffusion constants between the two proteins. The model thus supports the hypothesis that protein transport in the ER happens through diffusion.



Sec61, $\tau_S = 1026$ *GFP*, $\tau_G = 46$

Figure 2.1: Exponential time constants of measured FLIP data when the mother cell is bleached. The fitted solution of the simple one-reservoir model is shown as a solid line and the experimental FLIP data as circles. The ratio of time constants between the ER-membrane protein and the soluble protein is $\tau_S/\tau_G = 22.3$

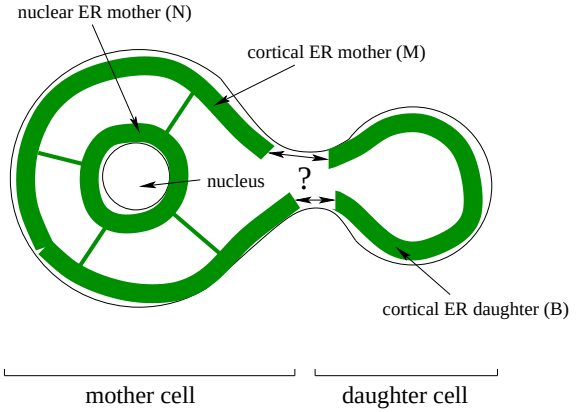


Figure 2.2: Schematic of the ER in budding yeast before nuclear division. The ER forms three compartments: one concentrated in the perinuclear area of the mother cell (N), one in the cortical region of the mother cell (M) and one in the cortical region of the bud (B). All compartments are known to be connected but the exact mechanism by which the cortical ERs are connected is unknown.

We can now extend this model to do real biology with it. Before nuclear division, the ER in budding yeast cells is organized in three compartments: the perinuclear ER in the mother cell, the cortical ER in the mother cell, and the cortical ER in the bud (see Fig. 2.2). All three compartments are connected, but transport between them is much slower than transport inside a compartment. We can thus safely identify each compartment with a different reservoir in our model.

Between the two ER compartments inside the mother cell, thin tubular connections can be observed in fluorescence microscopy. Between the cortical ERs of the mother cell and the bud, however, no connections are visible. Nonetheless there must be *some* connection since bleaching the mother cell's cortical ER also drains the bud's ER. So the biological question is: *How are the cortical ERs of the mother cell and the bud connected?* Going one step further, we also ask: *Is this connection different for membrane proteins and for soluble proteins?*

We address this question using our modeling technique. Modeling is required in the present setting because the connection we want to investigate is *not observable* in experiments. This is one of the possible indications for modeling in biology (cf. Chap. 1). Using different FLIP combinations, we can indirectly probe the connections between the ER compartments, but correcting for dimensionality and compartment sizes requires a model. We thus extend our simple one-reservoir model from above to three compartments as follows:

1. system boundary = ER membrane
   u input bleaching rate and location of bleaching (mother/bud)
   y output fluorescence intensities in all 3 compartments

2. relevant reservoirs: fluorescence content in the three ER compartments N, M, B (see Fig. 2.2) ⇒ levels: fluorescent mass contents $n(t)$, $m(t)$, $b(t)$. Slow and fast processes are the same as in the one-reservoir model above.

3. Figure 2.3 shows a representation of the three compartments and the flows between them. Each compartment is attributed an (unkown) volume V and the connections between the compartments are assumed to be diffusive. They are characterized by their length L, cross-section area A, and the diffusion constant D (different for soluble and membrane proteins).

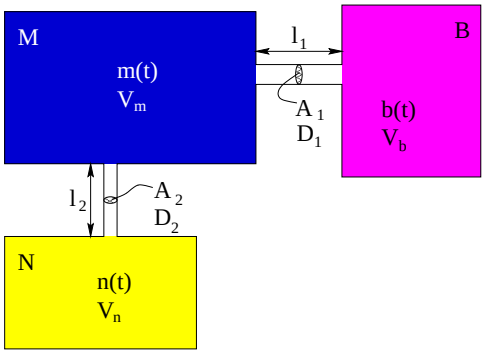


Figure 2.3: Abstraction of the three-reservoir model. The cortical ER in the mother cell (M) and the cortical ER in the bud (B) are connected by connection 1, the perinuclear ER in the mother cell (N) and the cortical ER in the mother cell are connected by connection 2. Each connection is characterized by its length L, cross-section area A, and diffusion constant D.

We thus have the following flows for the three reservoirs:

- N: diffusive inflow from M, diffusive outflow to M
- M: diffusive inflow from N, diffusive outflow to N, diffusive inflow from B, diffusive outflow to B, bleaching outflow *if* mother cell is bleached
- B: diffusive inflow from M, diffusive outflow to M, bleaching outflow *if* bud is bleached

Bleaching is modeled as $\frac{\nu}{V}\beta(t-k\Delta t)$ as above. The diffusive flows are modeled using Fick's law. Fick's law states that the flux (flow per unit area) due to diffusion is given by $j = -D\nabla c$, where D is the diffusion constant and ∇c the concentration gradient. We approximate the concentration gradient between two compartments x and y by their concentration difference, divided by the length of the connection, thus:

$$\nabla c \approx \frac{1}{L}\left(\frac{x(t)}{V_x} - \frac{y(t)}{V_y}\right) .$$

The concentration in a compartment is given by the total mass in that compartment divided by the compartment volume. We now define the *transport rate* of an inter-compartment connection as:

$$\lambda = \frac{DA}{L} .$$

The transport rate λ has the dimensions L^3/T and thus corresponds to the total volume of material that can be transported through the connection per unit time. With this, the diffusive flow between compartments x and y becomes:

$$\dot{x}(t) = \lambda\left(\frac{y(t)}{V_y} - \frac{x(t)}{V_x}\right) .$$

4. Now we can formulate the balance equations, taking all the mass flows between the reservoirs into account. Using the expressions from above for the flows, we find the equations as:

$$\begin{cases} \frac{db(t)}{dt} = \lambda_1\left(\frac{m(t)}{V_m} - \frac{b(t)}{V_b}\right) \\ \qquad\qquad -(1-\chi_m)\nu_b\frac{b(t)}{V_b}\beta(t-k\Delta t) \\ \frac{dm(t)}{dt} = \lambda_1\left(\frac{b(t)}{V_b} - \frac{m(t)}{V_m}\right) + \lambda_2\left(\frac{n(t)}{V_n} - \frac{m(t)}{V_m}\right) \\ \qquad\qquad -\chi_m\nu_m\frac{m(t)}{V_m}\beta(t-k\Delta t) \\ \frac{dn(t)}{dt} = \lambda_2\left(\frac{m(t)}{V_m} - \frac{n(t)}{V_n}\right) \end{cases} \tag{2.1}$$

with the transport rates for the two connections:

$$\lambda_1 = \frac{D_1A_1}{\ell_1}, \quad \lambda_2 = \frac{D_2A_2}{\ell_2}$$

and the indicator variable

$$\chi_m = \begin{cases} 1 & \text{if mother cortex is bleached} \\ 0 & \text{if bud is bleached} \end{cases} .$$

The colors in these equations correspond to the reservoir colors in Fig. 2.3. The structure of the flows as "activating level minus inhibiting level" is nicely

visible. Also notice that compared to the one-compartment model, we have actually included some spatial resolution. We know now in *which part* of the ER a certain fluorescent mass is. Increasing the number of reservoirs thus increases the spatial resolution. This is exactly the road we will follow in Chap. 4 to extend to spatiotemporal models.

5. The above balance equations can not be simplified any further.

6. Unlike the one-compartment model, the present three-compartment model can not so easily be solved analytically any more (in principle it could, but it would be more intricate). We thus choose to solve the model numerically. In this example, I have implemented the model in Matlab/Simulink, where reservoirs directly map to *integrators* and flows are represented by arrows. The corresponding Simulink diagram is shown in Fig. 2.4. This provides a graphical way of representing the model in the computer and simulating it by hitting the "play" button without having to program a single line of code. Once a reservoir-and-flow model is formulated, translating it to Simulink is very easy. Scopes attached to the integrators plot their respective levels over time, providing the numerical solution of the model.

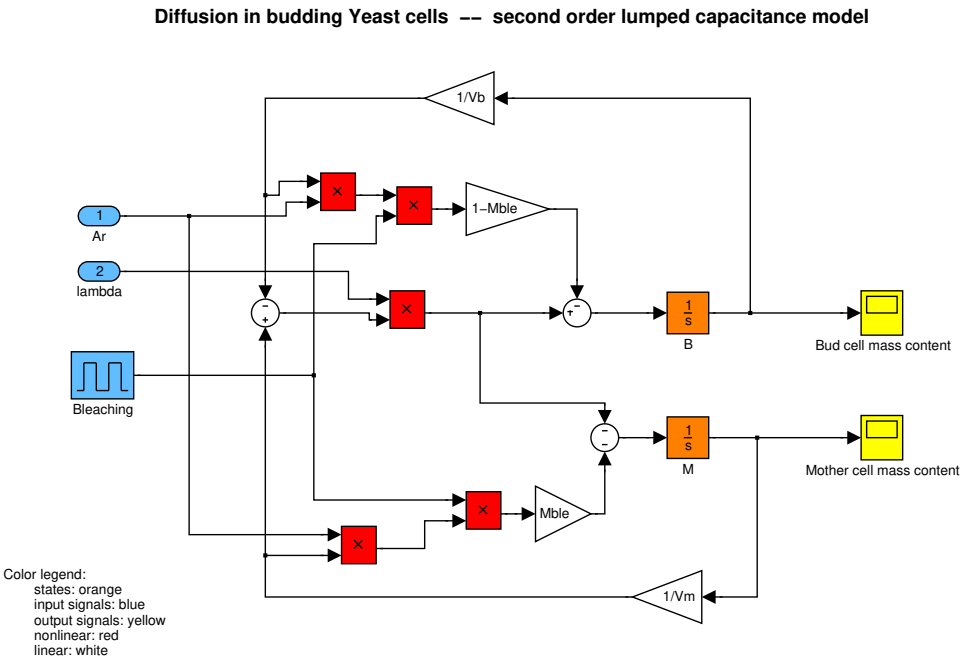


Figure 2.4: Matlab/Simulink implementation to numerically simulate the three-compartment ER FLIP model.

7. The four unknown parameters λ_1, λ_2, ν_b/V_b, and ν_m/V_m are determined by fitting the model to FLIP experiments using non-linear least-squares fitting. Four experiments are performed:

   - Membrane protein *Sec61*, cortical ER in the mother cell bleached
   - Membrane protein *Sec61*, cortical ER in the bud bleached
   - Soluble protein *GFP*, cortical ER in the mother cell bleached
   - Soluble protein *GFP*, cortical ER in the bud bleached

   It can easily be seen from the model Eq. 2.1 that this is the minimal set of experiments that allows to identify all model parameters. In each FLIP experiment, we monitor all three fluorescence levels $b(t)$, $m(t)$, and $n(t)$. The results of fitting the model to the experimental data is shown in Table 2.1. It can be seen that the model represents well the bleaching dynamics in all cases and in all compartments. Even the saw-tooth bleach-recovery cycles are captured.

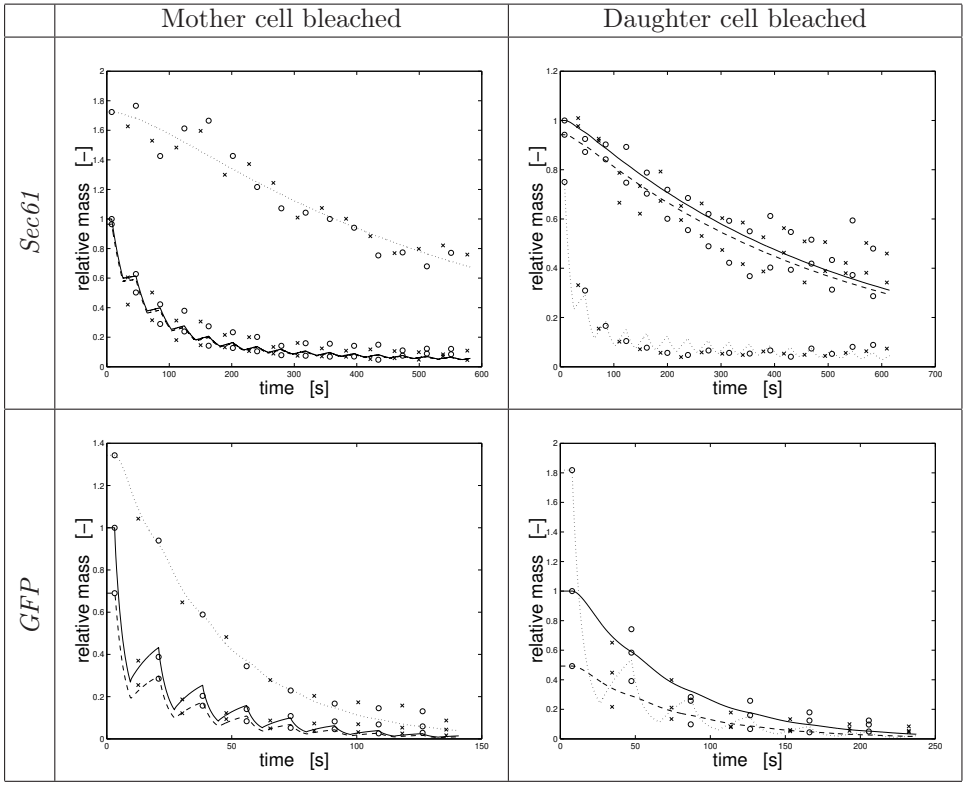


Table 2.1: Three-reservoir FLIP model parameter identification on experimental results. The solid lines corresponds to the model predictions of the total fluorescent mass in the cortical ER of the mother cell, the dotted lines to the model predictions for the cortical ER of the daughter cell, and the dashed lines to the predictions for the perinuclear ER. Circles mark experimental post-bleach values and pluses experimental pre-bleach values.

8. The model was validated on other, independent FLIP experiments in the same cells (total 7 experiments). In addition, the validity of the three-compartment approach was validated using a two compartment model in which the two ERs in the mother cell are lunmped together into one compartment. We will not show the results here as they simply confirm the validity of the present three-reservoir model.

After the model is developed and validated, we can use it to answer the original biological question. The transport rates for the different proteins and connections are summarized in Table 2.2. Comparing different connections for the same protein species, the diffusion constant D cancels out and the differences in λ are only caused by differences in A/L, i.e. the ratio between total connection cross-section and effective connection length. From the fitting results, we find:

$$\text{ssGFP: } \frac{\lambda_2}{\lambda_1} = \frac{13.7}{0.05} = \mathbf{274}\,, \qquad \text{Sec61-GFP: } \frac{\lambda_2}{\lambda_1} = \frac{15.0}{0.0034} = \mathbf{4411}\,.$$

If we assume that $L_2 \approx 5 \ldots 10 \cdot L_1$ (the radius of the mother cell is typically about $2.5\,\mu$m, such that the radial distance between the perinuclear and the cortical ER is on the order of $1 \ldots 2\,\mu$m and the separation gap between the two cortical ER parts is about $0.1 \ldots 0.4\,\mu$m), this means that the total cross-section of the connection within the mother cell is at least 10^3 to 10^4 times larger than the one between mother cell and bud. This explains why the former can be seen in fluorescence light microscopy, but not the latter. In summary, the model provides the following answer to the first biological question: The ER compartments within the mother cell behave much like a single compartment, whereas the bud is very weakly connected. This weakening of the connection is more pronounced for membrane proteins than for soluble proteins.

Table 2.2: Average model parameters from 7 FLIP experiments.

Case	λ_1	λ_2	RMS error
Sec61-GFP, mother bleached	0.00225	15.43	0.07709
Sec61-GFP, bud bleached	0.00435	14.71	0.10111
ssGFP, mother bleached	0.06668	14.07	0.06326
ssGFP, bud bleached	0.03042	13.12	0.08573

There seems to be a difference in behavior between membrane proteins and soluble proteins. This difference cannot be explained by the different diffusion constant as they cancel out in the λ ratio. In order to understand this, let's compare the transport rates of the two species for the same connection. We find:

$$\text{bud neck: } \frac{\lambda_{1,\mathrm{GFP}}}{\lambda_{1,\mathrm{Sec61}}} = 14.81\,, \qquad \text{mother: } \frac{\lambda_{2,\mathrm{GFP}}}{\lambda_{2,\mathrm{Sec61}}} = 0.934\,.$$

This means that the soluble *GFP* is exchanged about 16 times faster than the membrane-associated *Sec61* in the bud neck than it is within the mother cell. The model's answer to the second question thus is: The diffusion of *Sec61* is slowed down about 16-fold in the bud neck compared to *GFP*.

This behavior could be explained by assuming a diffusion barrier in the ER membrane across the bud neck. The model thus suggests a new hypothesis and new experiments. Indeed, the existence and identity of such a diffusion barrier (similar to the septin ring in the plasma membrane) have been confirmed experimentally (Lüdeke et al., J. Cell. Biol., 169, 2005). This was the first time that compartmentalization of an intracellular membrane has been demonstrated.

2.9 Causality Diagrams

Causality diagrams are a way of visualizing causal relationships in a system or a model. In steps (2) to (4) of the obove modeling procedure, it is often beneficial to draw a diagram with explicit cause-effect relations. This helps keeping overview and making sure all variables are accounted for in the balance equations.

2.9.1 Conventions

The following drawing conventions are used in causality diagrams:

reservoirs: shaded boxes

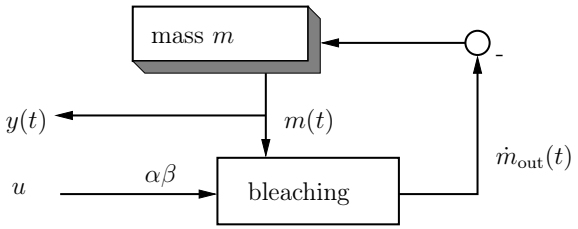
flows: plain boxes

causality: arrow with mediating variable indicated

sums: nodes

The entry point of the diagram must not depend on any property within the diagram. It must be an independent input.

Example 2.9.1 (Causality diagram of the one-reservoir FLIP model)**.**



Question 4. Derive the model equations and the causality diagram for an ecosystem with two species: A & B. Each (A,B) is characterized by:

- a probability of dying (δ) $\rightarrow$ assumed to be age-independent (not realistic)
- a probability of generating offspring (β)

A is a predator of B. At each encounter of A & B, there is a probability ϵ for B to be eaten. The birth rate of each A (β_A) depends on how many B it has eaten.

(a) Follow the modeling steps described above to write down the governing equations (ODE) for the excepted population dynamics. What are the reservoirs and flows? (K4)

(b) Draw the causality diagram for this system. (K3)

Chapter 3

Recapitulation of Vector Calculus

In this chapter:

- Scalar and vector fields
- Differential operators on fields
- Flux and work of vector fields
- Integral theorems
- Conservative fields and differential equations

Learning goals:

- Be familiar with the differential operators and the Nabla notation
- Be able to prove compute rules of differential operators in Cartesian coordinates
- Know the theorems of Gauss and Stokes by heart and be able to explain them intuitively
- Be able to define the terms "potential" and "state variable"
- Know when to use the Laplace and Poisson equations to describe a field

We review the most important concepts of vector analysis that are required in order to follow the rest of these lecture notes. The notation and concepts introduced here provide the mathematical foundation for the rest of the course and should be familiar to the reader. Knowledge of them is both necessary and sufficient as no other mathematical concepts than the ones recalled here will be used. Readers familiar with vector analysis can safely skip this chapter.

3.1 Fields

Fields are used to describe the spatial distribution or variation of quantities of interest. We distinguish:

Scalar fields : A scalar field is a scalar-valued function $f : \mathbb{R}^n \mapsto \mathbb{R} \quad (\underline{x}) \to f(\underline{x})$. Scalar fields describe the spatial distribution of scalar quantitites such as temperature, pressure, density, concentration, ...

In a scalar field, we can define **iso-lines** as curves $C(s) : s \to \underline{x}(s)$ such that $f(\underline{x})$ is constant along $C \ \forall \underline{x}$.

Vector fields : A vector field is a vector-valued function $\underline{v} : \mathbb{R}^n \mapsto \mathbb{R}^m \quad (\underline{x}) \to \underline{v}(\underline{x})$. Vector fields describe the spatial distribution of vector (directed) quantitites such as velocity, force, electric field, ...

In a vector field, we can define **field lines** as curves $K(s) : s \to \underline{x}(s)$ such that $\underline{v}(\underline{x})$ is tangential to $K \ \forall \underline{x}$. These curves are the solution of the ODE: $\frac{dx}{v_x} = \frac{dy}{v_y} = \frac{dz}{v_z}$.

Example 3.1.1.

- The field lines of the velocity field of a rigid body rotation are concentric circles in the planes normal to the axis of rotation with centers on the axis of rotation.
- The field lines of the velocity field of fluid flow are the trajectories of the molecules.

With respect to their temporal dynamics, we distinguish:

Definition 3.1.1 (stationary field)**.** In a stationary field, the scalar or vector value at each location does not change over time.

Definition 3.1.2 (unsteady field)**.** In an unsteady field, the scalar or vector value depends on time in at least certain locations.

3.1.1 Differentation of vectors

Vector fields can be differentiated analogously to scalar fields. Following are the rules of differentiation on vectors:

Let $\underline{a}(t)$, $\underline{b}(t)$, and $\underline{c}(t)$ be vector fields in the scalar variable $t \in \mathbb{R}$ and $\varphi(t)$ a scalar field. Then:

$$\begin{aligned}
\frac{d}{dt}(\underline{a} \pm \underline{b} \pm \underline{c}) &= \frac{d\underline{a}}{dt} \pm \frac{d\underline{b}}{dt} \pm \frac{d\underline{c}}{dt} \\
\frac{d}{dt}(\varphi \underline{a}) &= \frac{d\varphi}{dt}\underline{a} + \varphi\frac{d\underline{a}}{dt} \\
\frac{d}{dt}(\underline{a} \cdot \underline{b}) &= \frac{d\underline{a}}{dt} \cdot \underline{b} + \underline{a} \cdot \frac{d\underline{b}}{dt} \\
\frac{d}{dt}(\underline{a} \times \underline{b}) &= \frac{d\underline{a}}{dt} \times \underline{b} + \underline{a} \times \frac{d\underline{b}}{dt} && \text{(not commutative!)} \\
\frac{d}{dt}(\underline{a}(\varphi(t))) &= \frac{d\underline{a}}{d\varphi}\frac{d\varphi}{dt} && \text{(chain rule)}
\end{aligned}$$

3.2 Differential Operators

Scalar and vector fields can be manipulated using differential operators that involve some combination of derivatives of the field. The result of a differential operator is again a scalar or vector field, but not necessarily of the same type as the argument. The most important differential operators in spatiotemporal models in biology are:

- **Gradient** $\operatorname{grad} f(\underline{x})$

  in Cartesian $\mathbb{R}^3$:

  $$\operatorname{grad} f(x,y,z) = \left(\frac{\partial f}{\partial x}(x,y,z),\ \frac{\partial f}{\partial y}(x,y,z),\ \frac{\partial f}{\partial z}(x,y,z)\right)$$

  in general:

  $$\operatorname{grad} f = \lim_{\Delta v \to 0} \frac{\oint_\Sigma f \, d\underline{S}}{\Delta v}$$

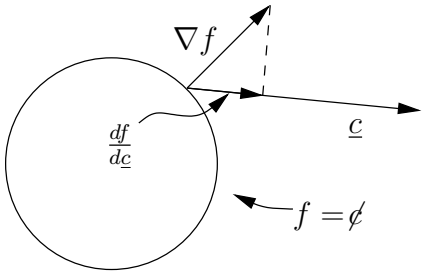
  - Σ: closed surface
  - Δv: enclosed volume

  This can be intuitively understood as follows: $d\underline{S}$ is a boundary element (with outer unit normal) of the volume Δv with boundary Σ. The outer normal on Σ is scaled with the local value of the field f at every point and the scaled normals are integrated over the whole boundary of Δv. If f has the same value everywhere, the integral evaluates to zero. If f is larger on one side of Δv than on the other, the result will be a non-zero vector pointing from the side where f is smaller to the side where it is larger. If we let the size of the volume Δv go to zero, we obtain a vector indicating the local point change in f and its direction.

  properties:

  - orthogonal to iso-surfaces $f = \not c$
  - points in direction of the steepest increase of f.
  - $\operatorname{grad} f = 0 \Leftrightarrow$ local extremum (minimum, maximum, or saddle point) of f.

- **Directional derivative** $\frac{df}{d\underline{c}}$ with $||c|| = 1$.
  The directional derivative gives the infinitesimal change of f in direction $\underline{c}$, thus: $\frac{df}{d\underline{c}} = \underline{c} \cdot \operatorname{grad} f$. It is equivalent to the projection of the gradient of f onto the unit vector $\underline{c}$.



- **Divergence** $\operatorname{div} \underline{v}$

  in Cartesian $\mathbb{R}^3$:

  $$\operatorname{div} \underline{v}(x,y,z) = \frac{\partial v_1}{\partial x}(x,y,z) + \frac{\partial v_2}{\partial y}(x,y,z) + \frac{\partial v_3}{\partial z}(x,y,z)$$

  in general:

  $$\operatorname{div} \underline{v} = \lim_{\Delta v \to 0} \frac{\oint_\Sigma \underline{v} \cdot d\underline{S}}{\Delta v}$$

  Again, this can be intuitively understood: the vector field is locally projected onto the outer unit normal of the boundary Σ of the volume Δv and all the projections are integrated over the whole closed boundary. The integral will thus evaluate to the total amount of $\underline{v}$ that is crossing the boundary. If we let the size of the volume Δv go to zero, we obtain the local amount (scalar) of $\underline{v}$ that is "emerging" out of a point in space.

  properties:

  - the divergence of a homogeneous field $\underline{v}$ (i.e. $\underline{v} \equiv \underline{a}$) is zero, thus: $\operatorname{div} \underline{v} \equiv \underline{0}$.
  - the divergence is a unit source strength; it gives the amount of the quantity that is newly generated (or removed for negative signs) per unit volume and unit time.

- **Curl** $\operatorname{curl} \underline{v}$

in Cartesian $\mathbb{R}^3$:

$$\operatorname{curl} \underline{v}(x,y,z) = \left(\frac{\partial v_3}{\partial y} - \frac{\partial v_2}{\partial z}, \frac{\partial v_1}{\partial z} - \frac{\partial v_3}{\partial x}, \frac{\partial v_2}{\partial x} - \frac{\partial v_1}{\partial y} \right)$$

in general:

$$\operatorname{curl} \underline{v} = \lim_{\Delta v \to 0} \frac{\oint_\Sigma d\underline{S} \times \underline{v}}{\Delta v}$$

The intuitive meaning of this equation is the following: at every point on the boundary of the volume Δv, we compute the cross product between the local vector field value and the outer unit normal on the volume's boundary Σ. This cross product will be maximum if $\underline{v}$ is tangential to Σ and zero if $\underline{v}$ is parallel to $d\underline{S}$. We then integrate this quantity over the whole boundary, thus measuring the net amount of $\underline{v}$ that is "running around" Σ. If we let the size of the volume go to zero, we obtain a vector whose length is the point-wise local vortex strength or rotation of the vector field $\underline{v}$ and whose direction indicates the axis of rotation.

properties:

- The curl of a vector field gives the vorticity per unit volume.

Notice that none of the above definitions depends on the shape of the test volume Δv.

Instead of using the symbolic names of the differential operators, they can also be compactly written in terms of the

- **Nabla operator ∇** (Notation, **not** a new operator!)

  - **in Cartesian $\mathbb{R}^3$:**

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$$

The basic differential operators introduced so far can thus equivalently be written as:

- $\operatorname{grad} f = \nabla f$
- $\operatorname{div} \underline{v} = \nabla \cdot \underline{v}$
- $\operatorname{curl} \underline{v} = \nabla \times \underline{v}$ (Note that while this notation is always used, it does not work to derive a formula in 2D, where the curl is actually a scalar. There, an auxiliary dimension has to be introduced.)

This, for example, allows the compact definition of the important

- **Laplace operator $\Delta = \nabla \cdot \nabla = \nabla^2$** (scalar product)

in Cartesian $\mathbb{R}^3$:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

property : The Laplace operator of a scalar field is invariant to translation and rotation of the coordinate system.

We can summarize the family of differential operators used in this lecture as:

Operator	Symbol	Nabla	Argument	Result	Interpretation
gradient	$\operatorname{grad} f$	∇f	scalar	vector	steepest ascent
divergence	$\operatorname{div} \underline{v}$	$\nabla \cdot \underline{v}$	vector	scalar	source density
curl	$\operatorname{curl} \underline{v}$	$\nabla \times \underline{v}$	vector	vector	vortex strength
Laplace	$\triangle f$	$(\nabla \cdot \nabla) f$	scalar	scalar	sources in potential field

3.2.1 Compute rules for differential operators

The following compute rules are handy when doing algebra with differential operators. All rules can easily be proven from the basic definitions of the operators as given above.

$$\begin{aligned}
(1): &\quad \operatorname{grad}(f_1 + f_2) = \operatorname{grad} f_1 + \operatorname{grad} f_2 \\
(2): &\quad \operatorname{grad}(cf) = c \operatorname{grad} f \\
(3): &\quad \operatorname{grad}(f_1 f_2) = f_1 \operatorname{grad} f_2 + f_2 \operatorname{grad} f_1 \\
(4): &\quad \operatorname{grad} F(f) = F'(f) \operatorname{grad} f \\
(5): &\quad \operatorname{div}(\underline{v}_1 + \underline{v}_2) = \operatorname{div} \underline{v}_1 + \operatorname{div} \underline{v}_2 \\
(6): &\quad \operatorname{div}(c\underline{v}) = c \operatorname{div} \underline{v} \\
(7): &\quad \operatorname{div}(f\underline{v}) = \underline{v} \cdot \operatorname{grad} f + f \operatorname{div} \underline{v} \\
(8): &\quad \operatorname{curl}(\underline{v}_1 + \underline{v}_2) = \operatorname{curl} \underline{v}_1 + \operatorname{curl} \underline{v}_2 \\
(9): &\quad \operatorname{curl}(c\underline{v}) = c \operatorname{curl} \underline{v} \\
(10): &\quad \operatorname{curl}(f\underline{v}) = f \operatorname{curl} \underline{v} - \underline{v} \times \operatorname{grad} f \\
(11): &\quad \operatorname{div} \operatorname{curl} \underline{v} = 0 \\
(12): &\quad \operatorname{curl} \operatorname{grad} f = \underline{0} \\
(13): &\quad \operatorname{div} \operatorname{grad} f = \triangle f \\
(14): &\quad \operatorname{curl} \operatorname{curl} \underline{v} = \operatorname{grad} \operatorname{div} \underline{v} - \triangle \underline{v} \\
(15): &\quad \operatorname{div}(\underline{v}_1 \times \underline{v}_2) = \underline{v}_2 \cdot \operatorname{curl} \underline{v}_1 - \underline{v}_1 \cdot \operatorname{curl} \underline{v}_2
\end{aligned}$$

3.3 Flux (Φ)

In models of real-world systems, we are often concerned with the flow of things between reservoirs or regions of space (see Section 2.7). The concept of flows is mathematically formulated by fluxes and defined for a vector field $\underline{v}$ and a surface S, where $\underline{n}$ is the outer unit normal onto S.
Question: be $\underline{v}$ the velocity field of a flow; how much fluid flows through S per unit time in direction $\underline{n}$?

$\Rightarrow$ split S into infinitesimal area elements dS. Because they are infinitesimal, the dS are flat and $\underline{v}$ is homogeneous within a single dS. The total flow through a single dS thus is:

$$d\Phi = \underline{v} \cdot \underline{n} dS \,.$$

The flux is the "sum" over all infinitesimal dS and thus given by the integral:

$$\Phi = \int_S \underline{v} \cdot \underline{n} dS \,.$$

Often, we simplify the notation by writing: $\underline{n}dS = d\underline{S}$ and $\Phi = \int_S \underline{v} \cdot d\underline{S}$.

3.4 Work (W)

Another fundamental quantity in modeling is the work (a flow of energy) done by a vector field $\underline{v}$ along a line path L with beginning A and end B.

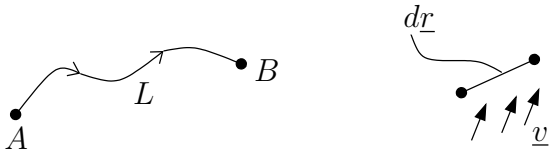
Question: be $\underline{v}$ a force field; how much work is done by $\underline{v}$ moving a point mass along L from A to B?

$\Rightarrow$ split L into infinitesimal line segments $d\underline{r}$. Because they are infinitesimal, the $d\underline{r}$ are straight and $\underline{v}$ is homogeneous within a single $d\underline{r}$. The work done by the force $\underline{v}$ per line segment thus is:

$$dW = \underline{v} \cdot d\underline{r} \,.$$

The total work is the "sum" along the complete path L, thus the integral:

$$W = \int_L \underline{v} \cdot d\underline{r} \,.$$



For parametric curves:

$$L : t \mapsto r(t) \quad \Rightarrow \quad d\underline{r} = \dot{r}(t) dt \,.$$

Example 3.4.1.

- Circulation is the work done by moving a unit mass in a velocity field.
- Voltage is the work done by moving a unit charge in an electric field.

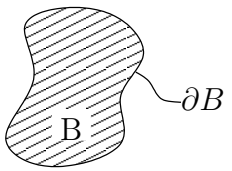
3.5 Integral Theorems

The quantities of flux and work are linked through important integral theorems that are due to Gauss and Stokes. The Gauss theorem relates surface fluxes to space quantities and the Stokes theorem relates surface quantitites to line work. They can be used to transform all quantitites in a model to space/surface quantitites in order to derive the differential equation in that space. The theorems are:

- **Gauss**
  Consider a vector field $\underline{v}$ that is defined and continuously differentiable in the closed region B with boundary ∂B and outer unit normal $\underline{n}$.

  Gauss Theorem:

$$\boxed{\oint_{\partial B} \underline{v} \cdot \underline{n} \, dS = \int_B \operatorname{div} \underline{v} \, dV}$$



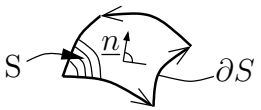
"The flux of $\underline{v}$ through ∂B from inside to outside is equal to the volume integral of $\operatorname{div} \underline{v}$ over the enclosed volume B."

Intuitive: "What is produced inside B has to flow out." This is a consequence of the conservation of mass in a flow with velocity field $\underline{v}$.

- **Stokes**
  Consider a vector field $\underline{v}$ that is defined and continuously differentiable in a region D. Consider further a bounded surface S that is entirely contained in D and has the border line ∂S.
  C is a path along ∂S such that its sense forms a right-hand screw with the normal onto S.

  Stokes Theorem:

$$\boxed{\oint_C \underline{v} \cdot d\underline{r} = \int_S \operatorname{curl} \underline{v} \cdot \underline{n}\, dS}$$



"The work of $\underline{v}$ along the boundary path C is equal to the flux of $\operatorname{curl} \underline{v}$ throught the enclosed surface S."

Intuitive: "The circular work invested is equal to the flux of rotational energy." This is a consequence of the conservation of energy.

Based on these two fundamental integral theorems, others can be derived. Green's theorems are, for example, frequently used when deriving partial differential equations that govern spatiotemporal models. They can be obtained by applying Gauss to $\underline{v} = f_1 \nabla f_2$ and read:

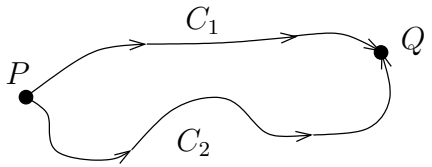
- **Green**
  1. $\int_B (f_1 \Delta f_2 + \nabla f_1 \cdot \nabla f_2) dV = \oint_{\partial B} f_1 \nabla f_2 \cdot d\underline{S}$
  2. $\int_B (f_1 \Delta f_2 - f_2 \Delta f_1) dV = \oint_{\partial B} (f_1 \nabla f_2 - f_2 \nabla f_1) \cdot d\underline{S}$

3.6 Conservative Fields

The special class of conservative fields plays an important role in practical modeling applications. It directly relates to physically conserved quantitites such as mass, energy, impulse, or momentum, and it formalizes the concept of state variables that we used for the reservoir levels in Section 2.7.

Definition 3.6.1. A vector field $\underline{v}(\underline{x})$ is called conservative if and only if the work along all possible paths from P to Q is equal, for all (P, Q).

$\Rightarrow$ The work along a closed curve is always zero.
$\Rightarrow$ The work does not depend on the specific path chosen, but only on the starting point and end point. These points (P and Q above) are hence *states* of the system, which we earlier identified with the levels of the reservoirs. For a conserved quantity, it does hence not matter *how* the level of a reservoir is reached, but only how large it is. The concept of conservative fields formalizes this. The points P and Q are then points in the state space of the model, i.e., each point corresponds to a certain set of reservoir levels.



The following facts about conservative fields are important:

- Each gradient field is conservative and vice versa. $\operatorname{grad} f$ is thus often called "potential field" and f its "potential".
- Each potential (gradient) field is vortex-free and vice versa, $\operatorname{curl} \operatorname{grad} f \equiv 0$ for all f.
- The work in a potential field only depends on the potential difference between the end point and the starting point, $f(Q - P)$. These are thus the **state variables** (reservoir levels) of the system and the expression should recall the "activating level minus inhibiting level" form of flows between reservoirs (see Section 2.8). We see now that this is true for all conserved quantities, which earlier allowed us to formulate the balance equations for the reservoirs.

3.7 Differential Equations

Using the above conservation properties of differential operators, we can define two basic partial differential equations that occur in many models. We can use these equations as building blocks for models whenever the conditions they describe appear. The equations are:

- **Laplace:** $\Delta f = 0$
  - The solution of the Laplace equation is the potential of a conservative field without sources (i.e. $\operatorname{div} \operatorname{grad} f = 0 \Leftrightarrow \Delta f = 0$).
  - The solutions are called "harmonic functions".
- **Poisson:** $\Delta f = q(\underline{x})$

- The solution of the Poisson equation is the potential of a convervative field with given source density $q(\underline{x})$. This can be used as a model for the potential whenever the source density is known.
- The solution is called the "Coulomb potential".

The actual solutions of these equations depend on the shape of the system (domain) in which they are solved and on the boundary conditions. From this, the entire field f inside the domain is determined, mathematically defining a *boundary value problem*.

Chapter 4

Modeling Spatial Effects

In this chapter:

- Eulerian and Lagrangian control volumes
- The Reynolds transport theorem for extensive quantitites
- Infinitesimal control volumes and PDEs
- An example: derivation of the diffusion equation

Learning goals:

- Know the Eulerian and Lagrangian descriptions
- Know material derivatives by heart and be able to explain them intuitively
- Be able to formally define intensive and extensive quantities
- Know the Reynolds transport theorem
- Be able to apply the Reynolds transport theorem and conservation laws to derive PDEs

We now extend our modeling framework to spatiotemporal models, still using the notion of reservoirs and flows. Recall the FLIP example of Sec. 2.8. In the one-reservoir model we only knew the total fluorescent mass in the ER, but had no information about how it is distributed. In the three-reservoir model we already knew in which part of the ER a certain amount of fluorescent mass is, but still not how it is distributed within each part (compartment). We can continue playing this game and subdividing the reservoirs further. Every time we do so, we gain more spatial resolution. Our modeling framework remains valid, but we will get a very large number of ODEs. Writing balance equations for the time evolution of the reservoir levels in all the control volumes leads to a spatiotemportal model where the spatial resolution is defined/limited by the size of the control volumes. Mathematically, we can let their size go to zero (and their number to infinity), in which case we recover continuous PDE models. Such models are, however, only valid for Kn $\to 0$. In reality, the control volumes can thus not be smaller than λ. In order to resolve all field gradients, they should, however, be smaller than L.
We rely on the mathematical tools from vector analysis and on numerical computer simulations to formulate and solve such models. Also, as we continue subdividing the reservoirs, they will not correspond to real-world compartments any more, but are simply identified with subspaces of the modeled physical space. These subspaces can be arbitrarily placed, they can be fixed in space or move, and they can be made infinitesimally small. In the case of an infinite number of infinitely small reservoirs, we will recover the continuum limit of partial differential equations (PDEs). Still, all modeling is based on formulating balance equations for conserved extensive quantities.

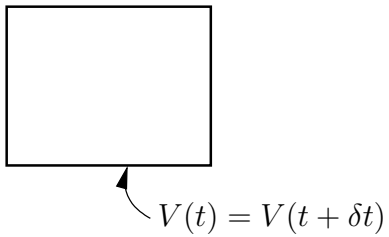
4.1 Control Volume Methods

The "subspaces" mentioned above are formalized as control volumes:

Definition 4.1.1 (Control Volume)**.** A control volume is a volume of integration, contained in a field. It can be arbitrarily placed and shaped.
⇒Because a control volume has a defining boundary, it is a *system* in and by itself!

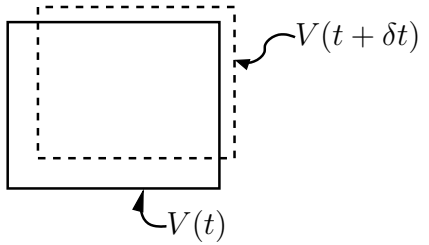
Let's assume a scalar field $f(\underline{x},t)$ (e.g. concentration) and a vector field $\underline{v}(\underline{x},t)$.(e.g. flow velocity). There are two formulations of control volumes, depending on whether the control volume is fixed in space or moving:

Euler



- fixed in space w.r.t. system boundary
- flux of $\underline{v}$ across the boundary

Lagrange



- movig and deforming with $\underline{v}$. Only "sees" the temporal change when following a set of particles
- always contains the same set of particles, no flux of $\underline{v}$ across the boundary.

A Lagrangian control volume always contains the same set of particles and tracks their motion in the velocity field $\underline{v}$. Therefore, there is no flux of $\underline{v}$ across the boundary of the control volume. There is, however, a flux of f across the boundary as the control volume is traveling through the field f with velocity $\underline{v}$.
As an example, consider the flow in a river. An Eulerian control volume is like a basket that we hold into the water and we observe from the shore how the water flows through it. Once we sit into the basket and let go of the shore, we travel downstream with the flow and hence the basket became a Lagrangian control volume.

4.1.1 Derivatives in control volumes

What is the temporal change of f (not $\underline{v}$!) that one "feels" in an Eulerian and a Lagrangian control volume? Let's say the water temperature in the river becomes warmer as we go further downstream. What's the temporal change of temperature we feel in the basket that is anchored to the shore versus the basket that travels with the flow? These will lead to the balance equations in the control volumes. The temporal derivatives of the field quantity f in the two control volume formulations are:

Euler

$$\left.\frac{\partial f}{\partial t}\right|_{\underline{x}=\text{const}}$$

($\underline{x}$ fixed !) (field derivative)
⇒change of f at a fixed point in space. In our example with the river, this derivative would be local change in water temperature at the given, fixed location, for example from the sun heating up the water.

Lagrange

$$\frac{\partial}{\partial t}(f(\underline{x}(t),t)$$

($\underline{x}$ varying!)

$$= \frac{\partial f}{\partial \underline{x}} \underbrace{\frac{\partial \underline{x}}{\partial t}}_{\text{velocity } \underline{v}} + \frac{\partial f}{\partial t}$$

$$= \boxed{\underbrace{\frac{\partial f}{\partial t}}_{\text{creation}} + \underbrace{(\nabla f)\cdot \underline{v}}_{\text{flow}} = \frac{Df}{Dt}}$$

(material derivative)
⇒change of f in a control volume that travels with velocity $\underline{v}$. In the example with the river, this derivative contains the change in temperature as the basket gets convected down into warmer waters in addition to the sun locally heating up the water. The former is captured by the second term on the left-hand side, which describes the change due to moving through the field gradient ∇f with velocity $\underline{v}$.

Even though we have derived the expression for the material derivative purely mathematically, it intuitively makes a lot of sense. It states that the rate of change felt when traveling with the control volume is composed of two terms: (1) the intrinsic change in the field quantity f at the location where the control volume currently is $(\partial f/\partial t)$; (2) the change felt from moving along the gradient of f, i.e. moving into regions of higher/lower f. The scalar product makes sense since moving perpendicular to the gradient (along iso-lines) does not make you move into regions where f is different.

Example 4.1.1 (Stepping out of the house)**.** Consider the example of you stepping out of the house. What is the temperature difference that you feel when doing so? You are a Lagrangian control volume because you are moving with yourself. The change in temperature that you experience has two components: (1) spontaneous changes of the outside temperature at any fixed location, e.g. because a cloud is occluding the sun causing the temperature to drop over time. (2) your motion in the temperature gradient between indoors and outdoors. The first component is $\partial f/\partial t$, the second one is $(\nabla f)\cdot\underline{v}$. The temperature gradient between indoors and outdorrs is ∇f, and $\underline{v}$ is the velocity with which you are moving. The scalar product tells that only the component of $\underline{v}$ *along* the gradient lets you feel a change in temperature, and the velocity is there because moving faster lets you feel a more rapid change.

For vector fields, the expressions look exactly the same with the vector gradient $\frac{\partial v_j}{\partial x_i}$ and the matrix-vector product $(\nabla\underline{v})\underline{v}$.

4.1.2 Reynolds transport theorem (Reynolds, 1895)

Now we know how to compute the time derivative of an intensive field quantity in a moving (Lagrangian) control volume, namely by the material derivative

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \overbrace{\underline{v}\cdot(\nabla f)}^{=(\underline{v}\cdot\nabla)f},$$

which relates the temporal change in the traveling control volume (Df/Dt) to the instantaneous local change of the field quantity f $(\partial f/\partial t)$.

But how do we compute the material derivative of integral (extensive) quantities $\varphi = \int_V f dV$ in a control volume V?

Example 4.1.2.
An example for such an integral quantity is mass, defined as the volume integral over the scalar density (concentration) field, thus: $m = \int_V \rho \, dV$.
Mass can of course be transported in space, so mass is in general not conserved in an Eulerian control volume. In a Lagrangian control volume, however, mass is conserved. Remember that a Lagrangian control volume always contains the same particles. Conservation of mass thus mathematically means $\frac{Dm}{Dt} = 0$, rather than $\frac{\partial m}{\partial t} = 0$ (!).

Now we can define:

Definition 4.1.2 (Conservation)**.** An **extensive quantity** that remains constant within a **Langrangian** control volume is called **conserved**.

In order to build models based on conservation laws (i.e. formulate the balance equations for reservoirs), we thus need to express $D\varphi/Dt$ in terms of derivatives of f. We can express it in terms of Lagrangian (material) derivatives of f as:

$$\frac{D\varphi}{Dt} = \frac{D}{Dt}\int_V f \, dV = \int_V \frac{Df}{Dt}\, dV$$

(Notice that the volume of integration does not change with time as we are sitting inside the Lagrangian control volume and traveling with it. There is thus no relative velocity bewteen the control volume and the volume of integration. There is thus no change of the control volume with respect to the integration volume. Do not confuse this with the change of the Lagrangian control volume with respect to the embedding space!)

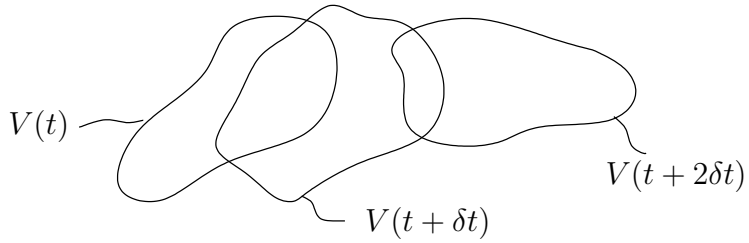
or in terms of Eulerian (field) derivatives of f as:

$$\begin{aligned}
\frac{D\varphi}{Dt} &= \frac{D}{Dt}\int_{V(t)} f dV(t) \\
&= \int_{V(t)} \frac{Df}{Dt} dV(t) + \int_{V(t)} f \underbrace{\frac{D}{Dt}[dV(t)]}_{\underbrace{\frac{\partial dV(t)}{\partial t}}_{\underbrace{=0}_{\substack{\text{volume element}\\ \text{at same location}\\ \text{stays the same.}}}} + (\underline{v}\cdot\nabla)dV(t)} \\
&= \int_{V(t)} \left[\frac{Df}{Dt} + f(\underline{v}\cdot\nabla)\right] dV(t) \\
&= \int_{V(t)} \left[\frac{\partial f}{\partial t} + \underbrace{\underline{v}\cdot(\nabla f) + f(\nabla\cdot\underline{v})}_{=\nabla\cdot(f\underline{v})\text{: see compute rules}}\right] dV(t) \\
&= \int_{V(t)} \left[\frac{\partial f}{\partial t} + \nabla\cdot(f\underline{v})\right] dV(t)\,.
\end{aligned}$$

(Here, the volume of integration changes with time as we are watching the Lagrangian control volumes "flow by" at a fixed position.) Using Gauss' theorem, we obtain the Reynolds transport theorem:

$$\boxed{\frac{D\varphi}{Dt} = \int_{V(t)} \frac{\partial f}{\partial t}\, dV(t) + \oint_{S(t)} f\underline{v}\cdot\underline{n}\, dS(t)}\,.$$

The Reynolds transport theorem relates the Lagrangian derivative of an extensive quantity to the Eulerian description of the corresponding intensive quantity. It is therefore of great importance in modeling. Models are formulated in terms of reservoirs and their (extensive) levels. PDEs, however, are formulated for the associated intensive field quantity. Reynolds provides the link and allows translating Lagrangian, extensive conservation laws to Eulerian, intensive governing equations. It holds for any control volume of any shape.



Intuitively, Reynolds makes sense: consider the example of a bank account. The material derivative is the change in balance when "moving with the account". It is given by the integral of all field changes, such as fees or interest, plus all in and out transactions, corresponding to the flux in the second integral. It is the balance equation for a Lagrangian control volume.

Now that we have defined control volumes and know how to compute the time evolution of extensive and intensive quantities inside them, we can define the two types of quantities more formally:

Definition 4.1.3 (Extensive quantity)**.** The value of an extensive quantity depends on the size of the control volume. All integral properties are extensive (e.g. mass, energy, impulse, ...).

Definition 4.1.4 (intensive quantity)**.** The value of an intensive quantity is independent of control volume size. All field properties are intensive (e.g. density, temperature,velocity, ...).

4.2 Infinitesimal Control Volumes

If we are interested in the spatial variations of a field, we can apply a great number of very small ($\ll L$) control volumes that tile the space. Letting their size $|V| \to 0$ leads to the mathematical continuum limit of partial differential equations. Notice that this is only a mathematical limit and does not contradict the fundamental particle nature of matter. In fact we still require $\mathrm{Kn} \ll 1$ and hence only consider physical or biological length scales $> \lambda$. Within each infinitesimal control volume, the modeling technique of Chapter 2 can be used. The Reynolds transport theorem then allows translating the conservation laws to governing PDEs.
Finite-sized control volumes are used in **finite volume methods** to numerically solve the equations by means of flux balance in each grid cell. The method of control volumes thus directly relates to numerical simulations.

Example 4.2.1 (Diffusion)**.**
Let's see how the Reynolds transport theorem and infinitesimal control volumes can be used to derive in four steps the governing PDE for diffusion processes:

1. What quantities do we want to track? In diffusion, the intensive property of interest is the scalar concentration field $u(\underline{x}, t)$. The corresponding extensive (integral) property is the mass $m = \int_V u \, dV$.

2. What is conserved? Mass is a conserved quantity, thus:

$$\frac{Dm}{Dt} = 0 \,.$$

This already is the extensive Lagrangian formulation of the diffusion equation. Usually, however, we want a PDE that describes the spatiotemporal dynamics of the concentration field $u(\underline{x}, t)$, so we need to translate this extensive Lagrangian formulation into an intensive Eulerian one. We therefore apply Reynolds in a control volume V and express the conservation law for mass as:

$$\frac{Dm}{Dt} = \int_V \frac{\partial u}{\partial t} \, dV + \oint_{\partial V} u\underline{v} \cdot \underline{n} \, dS = 0 \,.$$

3. Formulate the algebraic equations for the fluxes. We need an expression for the flux density $u\underline{v}$. This is equivalent to the algebraic equations that we used for the flows between reservoirs in Section 2.8. This algebraic equation for the flux is called the *constitutive equation* and it is, in general, determined experimentally or from dimensional analysis. Since the constitutive equation is empirical, it is not universally valid. For diffusion, the constitutive equation is given by Fick's law as: $u\underline{v} = -D\nabla u$.
D is the diffusion constant of dimension $[D] = \frac{L^2}{T}$. Fick's law states that in diffusion mass flows against the concentration gradient. It has been found experimentally (empirically), but it can also be derived from statistical mechanics over the microscopic description of diffusion as molecules undergoing Brownian motion due to thermal fluctuations. Fick's law provides the equation for the fluxes across the control volume boundaries. Inserting the constitutive equation into the Reynolds theorem yields:

$$\Rightarrow \int_V \frac{\partial u}{\partial t} \, dV = -\oint_{\partial V} u\underline{v} \cdot \underline{n} \, dS = \oint_{\partial V} D\nabla u \cdot \underline{n} \, dS \overset{\text{Gauss}}{=} \int_V \nabla \cdot (D\nabla u) \, dV$$
$$\Rightarrow \int_V \left[\frac{\partial u}{\partial t} - \nabla \cdot (D\nabla u) \right] dV = 0 \,.$$

4. Take the limit to infinitesimal control volumes. This has to hold for all V, independent of their size, location, or shape. We can thus, in our minds, let the sizes of the control volumes go to zero ($|V| \to 0$) and at the same time increase the number of control volumes to infinity in order to ensure coverage of the complete

space. The only way the above equation can then still hold for all of the infinitely many (point-like) control volumes is if the integrand itself is zero, thus:

$$\boxed{\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u)}\,.$$

This is the PDE governing diffusion, the *diffusion equation*. Note how the model was formulated in terms of reservoirs of mass and using the conservation law for mass. The final equation, however, is formulated for the intensive concentration field. The Reynolds transport theorem, independently applied to each of the infinitely many infinitesimal control volumes, allowed us to make this link.

We have derived the diffusion equation in its most general, anisotropic and inhomogeneous form. If D does not depend on space and is a scalar (isotropic, homogeneous diffusion), we can exploit the linearity of the Nabla operator to further simplify the right-hand side of the diffusion equation to:

$$\nabla \cdot (D \nabla u) = D \nabla \cdot (\nabla u) = D \Delta u\,.$$

Chapter 5

Simulating Spatiotemporal Models Using Particle Methods

In this chapter:

- Function approximation using particles
- Operator approximation using particles
- Boundary conditions using the method of images
- Remeshing and moment-conserving interpolation in hybrid particle-mesh methods
- Fast neighbor search algorithms
- Short-range particle interactions using symmetry

Learning goals:

- Be able to explain the duality between intensive and extensive numerical simulation methods
- Know the particle function approximation by heart
- Be aware of approximation errors and the overlap condition
- Be able to explain and implement the method of images to handle boundary conditions in a simulation
- Be able to explain the need for remeshing and moment-conserving interpolation schemes
- Be able to implement cell lists and Verlet lists

Until now we have seen how to model dynamic systems in time and space. The method of reservoirs and flows, together with the concept of control volumes and the Reynolds transport theorem, can be used to derive the governing PDE of a system. Dimensional analysis can help check correctness of the derived PDE (dimensional homogeneity) and plan experiments to find the constitutive equations for the flows.
We now focus on how such models can be simulated in the computer. Mostly, the equations can not be solved analytically, especially not in the complex-shaped geometries of biological systems. Numerical solutions are thus an important tool.
Numerically solving a PDE relies on *discretizing* the equation in space and time. A PDE has an infinite number of dimensions, corresponding to the reservoir levels in the infinitely many infinitesimal control volumes. It can thus not be represented in a computer, which always has finite memory. Out of the infinitely many dimension, we thus select a finite number of "representative" ones. This amounts to selecting certain control volumes (or points in space) for which we will explicitly track the temporal evolution of the level variables. The assumption is that these represented control points are close enough together such that the field does not significantly vary between adjacent points and can be approximated by interpolation (i.e. the distance between control points should be $\ll L$). If this is not the case, then the discretization is *under-resolved* and the results can not be trusted.
After discretizing a PDE in space by selecting a finite number of representative *discretization points*, the number of degrees of freedom is finite and they can all be explicitly tracked over time. This, however, also requires discretizing the equation in time as we cannot compute the solution at every possible time point. Rather, we fix a temporal resolution δt and compute the solution only at times $k\delta t$ for integer k. This reduces the number of time points to a countable number, and we can use a *time integrator* ("time stepper") to iteratively advance the solution from time step k to $k+1$. After the PDE has been discretized in space and time, the infinite-dimensional problem is reduced to computing an approximation to the solution at a finite number of discretization points in space and for a finite number of time steps.
Depending on whether the chosen discretization points are positioned on a (regular or irregular) lattice or not, we distinguish grid-based and mesh-free discretization schemes. The classical grid-based schemes are finite differences (discretization along lines), finite elements (discretization over surface areas), and finite volumes (discretization over sub-volumes). Mesh-free methods include particle methods (discretization on zero-dimensional points), where particles directly correspond to Lagrangian control volumes that can also move.
We will use mesh-free particle methods to numerically simulate spatiotemporal models. In these methods, the representative control volumes are represented in the computer by particles of finite volume. These particles thus directly correspond to Lagrangian control volumes for which we will explicitly track the temporal evolution of the position and the level variables. It is intentional that I selected particle methods rather than more classical, grid-based finite difference or finite element schemes. The main reason for this choice is that particle methods naturally fit into the framework described so far. The computational particles used in these numerical methods correspond to Lagrangian control volumes that "contain" extensive quantities. Particle methods are thus closely linked to the physical or biological

processes underlying the model, which makes them intuitive and easy to understand and implement. We will not need to talk about series expansions or Sobolev spaces, making the topic much more accessible to a wide range of scientists. In addition, particle methods have a number of unique favorable properties in complex geometries and for simulating flows. Also the fact that they don't require a computational grid or mesh makes them easier to implement.

The fundamental data structure in particle methods are computational particles. These are discretization elements and do not necessarily need to correspond to real, physical particles. Particles, thus, do not directly represent molecules or atoms (except in molecular dynamics simulations, which we will not cover), but rather can be thought of as Lagragian control volumes.

Each particle is described by:

- its position $\underline{x}_p(t)$,
- its extensive quantity $\underline{\omega}_p(t)$, sometimes referred to as the particle's "strength",
- and its volume $V_p(t)$.

Particle methods provide

→ a direct link to physics and modeling since they are formulated in terms of extensive quantities (like the models)

→ relaxed stability limits because convection terms vanish in the Lagrangian description (there is no linear CFL condition in particle methods and the Navier-Stokes equations become linear in the Lagrangian form; see Sec. 8.2.1 and 9.2).

→ easy treatment of complex-shaped and deforming geometries because there is no need to generate a mesh, which is difficult in complex geometries

→ universality: particle methods can simulate a wide range of models, including discrete and continuous ones as well as deterministic and stochastic ones.

→ a way of simulating models for which the governing PDE does not exist or has not beed derived. We don't necessarily need to take the "detour" via Reynold's theorem and an intensive representation, but we can directly identify particles with reservoirs or control volumes and let them interact according to the flows between them. We hence remain in a purely extensive description.

The drawbacks of particle methods are:

→ boundary conditons are harder to impose than in grid-based methods, also because particles may not be available exactly at the boundary and extrapolation must be used.

→ the computational cost of particle methods is usually higher, due to neighbor-search algorithms and interpolations necessary. Also the irregular memory access patterns of particle methods (as opposed to grid-based methods) also leads to more cache misses and is less coalescent.

→ ensuring that the problem remains well-resolved (i.e., the particles remain "representative" discretization points) is more difficult if particles move.

In this course we focus on continuous models, always assuming that $\mathrm{Kn} \ll 1$ and that we are only interested in length scales $\gg \lambda$ (no molecular or atomistic processes). We will thus restrict our discussion to continuum particle methods, keeping in mind that particles can also be used for discrete models (trivially). A particle is hence represented by the tuple of its attributes:

$$(\underline{x}, \underline{\omega}, V)_p \,,$$

where the extensive quantity $\underline{\omega}_p$ contained in particle p is related to the underlying intensive field $\underline{u}$ through the particle volume as

$$\underline{\omega}_p = V_p \underline{u} \,.$$

This of course assumes that the value of $\underline{u}$ can be considered constant inside a particle. Hence, the particle sizes must be $\ll L$. The size of the particles thus defines the resolution limit of the numerical method and we must always use enough particles to resolve the spatial patterns that we are interested in.
The particle attributes $\underline{x}_p$ and $\underline{\omega}_p$ depend on time and evolve so as to satisfy the model PDE in a Langrangian frame of reference. Each particle property is thus governed by an ODE:

$$\begin{aligned} \frac{d\underline{x}_p}{dt} &= \sum_{q=1}^{N} \underline{K}(\underline{x}_p, \underline{x}_q, \underline{\omega}_p, \underline{\omega}_q) \\ \frac{d\underline{\omega}_p}{dt} &= \sum_{q=1}^{N} \underline{F}(\underline{x}_p, \underline{x}_q, \underline{\omega}_p, \underline{\omega}_q) \,. \end{aligned} \tag{5.1}$$

In general, there can also be an equation of evolution for the particle volumes V_p, but for now we will consider them constant for simplicity.

The right-hand sides of the above ODEs correspond to numerical quadrature (integration) of the kernels $\underline{K}$ and $\underline{F}$. This comes directly from the fact that particles carry extensive properties and we want to evaluate some integral over the control volumes they represent. This is in contrast to other numerical methods that directly discretize the derivative in the differential operator. These two way are illustrated in

Fig. 5.1 for time as the independent variable for simplicity. The same also works in space. In some numerical methods, the derivatives of $\underline{u}$ are discretized first, e.g. using finite differences, leading to algebraic equations. The resulting (linear) system of equations is then solved numerically. In the first step, one must be careful to ensure *consistency* of the method, i.e. ensuring that the discretized equations describe the same dynamics as the original equation in the limit for the discretization step going to zero. In the second step, we are worried about the numerical stability and the accuracy of the solution of the algebraic equations. In other methods, alternatively, we imagine solving the equation analytically by integration. This integral is then solved numerically by quadrature with support points t_i in time and quadrature weights w_i. Since the first step is exact and the second one is always stable (we simply sum numbers), the solution accuracy is the only thing left to worry about. While Fig. 5.1 illustrates this over time, the same also works in space, where the integral is over Green's function of the differential equation. This analytical solution always exists (if the equation has a solution at all), but may not be analytically integrable. This is why numerical methods use quadrature. The whole "art" then of course becomes to choose the quadrature weights such that the correct dynamics are represented. This will be the main topic of the rest of the lecture.
The first class of numerical methods, using way (1), is called "collocation methods" or, in case the independent variable is space, "intensive methods". The second class of methods, using way (2), is called "Galerkin methods" or, in the spatial case, "extensive methods" because they work with the integral quantity u. Particle methods can be of both type (1) and type (2), depending on the formulation. In the following, we will focus on type-2 methods where the particles cary integral quantities.

Although particle methods have been around for decades and have many favorable properties, they are not widely known and used. The reason is that evaluating the right-hand sides of above ODEs amounts to solving an N-body problem, in which each of the N particles interacts with all of the $N-1$ others. This leads to a total of $O(N^2)$ interactions to be evaluated, which quickly becomes computationally infeasible. Classical grid-based methods only need N interactions. Recent advances in algorithms, however, allow reducing the computational cost of particle methods to $O(N)$ as well, making them a viable alternative to grid-based methods. We will discuss these algorithms later in this chapter.

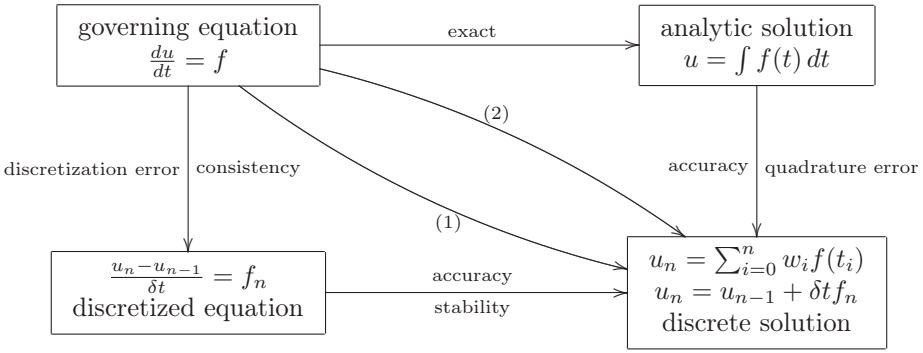


Figure 5.1: Strategies to numerically solve a differential equation in time: (1) discretization of the differential equation followed by numerical solution of the discretized equations, or (2) integral solution that is numerically approximated by quadrature.

5.1 Function Approximation by Particles

Any numerical method must do two things: approximate continuous field functions discretely, and approximate differential operators on those functions. The goal of particle function approximation is to approximate $u(\underline{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ by particles. This can be developed in three steps:

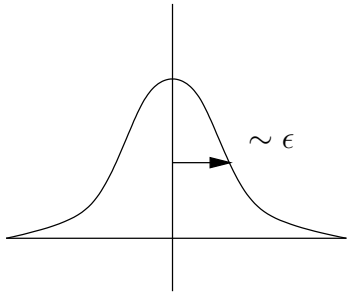
- **Integral Representation** We use the Dirac-delta identity to write the field u as an integral:

$$u(\underline{x}) = \int u(\underline{y})\delta(\underline{x} - \underline{y})\, d\underline{y}\,.$$

  This is already a particle representation if we interpret the delta functions as particles located at positions $\underline{y}$. In *point particle methods*, this is what people are doing. The problem is that the above integral is only exact for an infinite number of deltas. In practice this can of course not be done and we will

approximate the integral using a finite number of deltas. This then means, however, that the value of u can only be recovered at particle locations and is unknown in-between. Point particle methods thus amount to a mere sampling of the field u rather than to a smooth function approximation.

- **Mollification (Regularization)** In order to obtain a smooth approximation whose value is defined everywhere in space, we replace the delta functions by smooth kernels of a finite width ϵ. This amounts to regularizing δ as $\zeta_\epsilon = \epsilon^{-d}\zeta(\frac{x}{\epsilon})$ such that $\lim_{\epsilon\to 0}\zeta_\epsilon = \delta$. This obviously requires that $\int \zeta\, dx \stackrel{!}{=} 1$. ζ is called the mollification kernel of characteristic width ϵ:



The pre-factor ϵ^{-d} rescales the function such that its integral is always 1. Intuitively, ζ can be thought of as a cloud or blob of mass (or whatever extensive property the particles carry) that the particle carries around. This leads to the mollified (smooth) function approximation:

$$u_\epsilon(\underline{x}) = \int u(\underline{y})\zeta_\epsilon(\underline{x}-\underline{y})\, d\underline{y}\,. \tag{5.2}$$

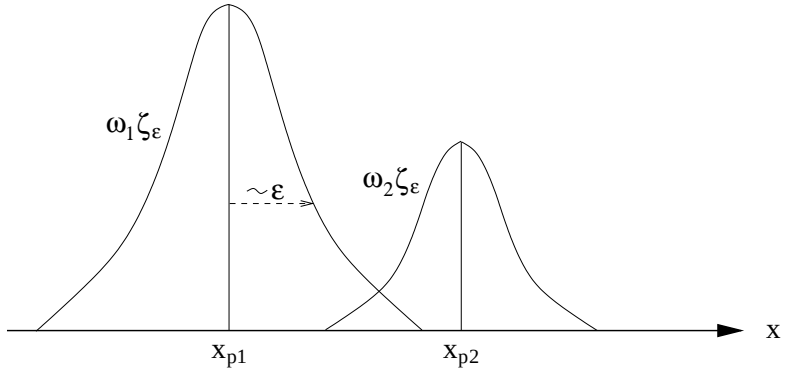
The approximation is more accurate the more moments of the delta function are conserved by the mollified kernel ζ. If ζ conserves the first $r-1$ moments of δ, the approximation is of order

$$u_\epsilon(\underline{x}) = u(\underline{x}) + O(\epsilon^r)\,.$$

This condition means that

$$\int x^s\zeta(x)\, dx \stackrel{!}{=} \int x^s\delta(x)\, dx \quad \forall s \in \{0\ldots r-1\}\,.$$

This means that the moment of order $s=0$ has to be 1, and all higher-order moments have to be zero. Non-negative kernels can thus never be of an order higher than 2, as only their first moment can vanish. In principle, any smooth and local function can be used as a kernel. The most frequent choice is a Gaussian, for which $r=2$.



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

- **Discretization** So far we have a smooth and continuous function approximation on infinitely many particles. We now discretize the approximation over a finite set of particles. This amounts to selecting "representative" particles which we want to explicitly represent in the computer. The problem dimension is thus reduced from infinity to a finite number. Discretizing the integral in Eq. 5.2 is straightforward. We use N-point quadrature with the particle locations as quadrature points:

$$u^h_\epsilon(\underline{x}) = \sum_{p=1}^{N} \omega^h_p \zeta_\epsilon(\underline{x}-\underline{x}^h_p)\,,$$

where $\underline{x}^h_p$ and ω^h_p are the numerical solutions of the particle positions and strengths, determined by discretizing Eqs. 5.1 in time. The quadrature weights ω^h_p are the particle *strengths*. They are extensive quantities because $\omega^h_p = u(\underline{y})d\underline{y}$. The discretized value depends on the specific quadrature rule used. For midpoint quadrature (the rectangular rule), we have:

$$\omega^h_p = u(\underline{x}^h_p)V_p\,.$$

The function approximation error now has two components: the mollification error and the discretization (quadrature) error:

$$u^h_\epsilon(\underline{x}) = u(\underline{x}) + O(\epsilon^r) + O\left(\frac{h}{\epsilon}\right)^s\,.$$

s is the number of continuous derivatives of ζ (for a Gaussian $s\to\infty$), and

h is the distance between particles.

From this expression, we see that in order for the error to be bounded, we have to require that

$$\frac{h}{\epsilon} \stackrel{!}{<} 1\,.$$

This condition means that the kernel widths of the particles must be greater than the distance between particles. The condition is thus frequently called *overlap condition* because it states that "particles must overlap." This makes sense because otherwise the value of the function u at off-particle locations could not be computed any more and we would be back to a point particle method.

5.2 Operator Approximation

Now that we know how to smoothly approximate field functions over a finite set of discrete particles that carry extensive properties, the question is how we can evaluate differential operators on the particles. Depending on how the operators are approximated, we distinguish between *pure particle methods* and *hybrid particle-mesh methods*. In pure particle methods, all operators are directly evaluated on the particles, whereas hybrid particle-mesh methods use an intermediate Cartesian mesh to evaluate some of the operators.

5.2.1 Pure particle methods

In order to evaluate differential operators on particles, we have to convert them to equivalent integral operators. This conversion is of course not exact and we end up with an approximate operator of a certain order of accuracy. The integral operator is then again discretized as a sum over the particles, leading to the sums on the right-hand side of Eqs. 5.1. The resulting operator for a differential operator of order β is of the form

$$\frac{1}{\epsilon^{|\beta|}} \sum_q V_q (u(\underline{x}_q) \pm u(\underline{x}_p)) \eta_\epsilon^\beta (\underline{x}_q - \underline{x}_p)\,, \tag{5.3}$$

where the operator kernel $\eta_\epsilon^\beta(\underline{x}) = \epsilon^{-d}\eta^\beta(\underline{x}/\epsilon)$ is suitably chosen (we will see later what this means). The evaluation of such an operator amounts to particle-particle interactions as governed by the integral kernel $\eta(\underline{x}_p, \underline{x}_q)$ of the operator. Notice that this kernel is not (necessarily) the same as the mollification kernel ζ used in the function approximation! In general, ζ and η can fulfill different moment conditions. We will see later how the kernels η look for different operators. For now, let's just distinguish between:

short-range operators where η has local, but not necessarily compact, support. In this case, only the neighbors within the kernel support of each particle contribute to the sum and we have to compute $O(N)$ interactions. This is done by limiting particle-particle interaction to particle pairs that are closer together than a cutoff radius r_c. An example would be the kernel for diffusion. Diffusion is a local process because its effect is not immediately apparent at remote locations.

long-range operators where all particles contribute. This defines an N-body problem because the kernel function η is not local. We potentially have to evaluate $O(N^2)$ interactions, rendering such methods infeasible. Fortunately, fast multipole algorithms are available to reduce the computational cost to $O(N)$ also in these cases. We will not discuss these algorithms here, but rather focus on the simpler hybrid particle-mesh approach. Examples of long-range processes are found in hydrodynamics or electrostatics, where a change in some part of the space has an immediate (modeling the speed of light as fast dynamics) effect throughout the entire domain.

5.2.2 Hybrid particle-mesh methods

Hybrid particle-mesh methods are available to efficiently compute the long-range parts of an integral (sum) approximation of a differential operator. This is done by introducing a regular Cartesian mesh that is superimposed over the particles. It is sufficient to consider regular Cartesian meshes because small-scale phenomena are still retained by direct particle-particle interactions and there are no boundaries to be considered. In a hybrid particle-mesh method, only the short-range parts of the operator are evaluated on the particles, whereas the long-range parts are accounted for by solving the corresponding PDE on the mesh. This requires:

- interpolating $\underline{\omega}_p$ from the particles to the mesh,
- solving the PDE[1] on the mesh using finite differences in a multi-grid method, or FFTs, and
- interpolating the solution back to the (not necessarily same) particles.

The equation that is solved on the mesh is called the *field equation*. Thanks to the regularity of the Cartesian mesh, it can be solved efficiently. The computational cost of multi-grid methods is $O(M)$ and that of FFT solvers is $O(M \log M)$, where M is the total number of grid points.

Example 5.2.1 (Fluid Flow)**.** If we were to simulate fluid flow using a hybrid particle-mesh scheme, we would solve for the velocity field on the mesh since

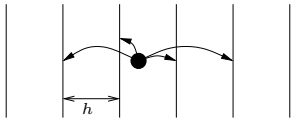
[1] Since most fields in real applications are gradient fields, this usually is the Poisson equation.

this is a long-range interaction. The convection, however, would be done on the particles (locally) by simply moving them. This preserves the favorable stability (no linear CFL condition, see Sec. 8.2.1) of particle methods, but allows taking advantage of efficient field solvers on the mesh.

The interpolation from particles to mesh can be done such that the conservation laws underlying the model are respected. This requires special interpolation schemes that conserve the moments of the represented extensive quantity up to a certain order. The moments directly represent the physically conserved quantities such as mass (zeroth-order moment), center of gravity (fist-order moment), and moment of inertia (second-order moment). Exactly conserving them when interpolating between particles and mesh constitutes a clear advantage as the interpolation will not introduce inconsistencies in the method.

Particles carry extensive quantities, but the mesh nodes store the values of the corresponding intensive field. Interpolation converts between the two representations. While moment-conserving interpolation schemes conserve the moments in particle-to-mesh interpolation, they generally do not do so in mesh-to-particle interpolation, unless the particles are themselves arranged on a regular lattice. Nonetheless, the interpolation error in mesh-to-particle interpolations decreases as a power of h, the mesh spacing. This power is called the *order of convergence* of the interpolation scheme. It is important not to confuse the order of convergence of the mesh-to-particle interpolation with the order of the highest conserved moment in the particle-to-mesh interpolation!

During particle-to-mesh interpolation, the strength of each particle is redistributed onto the surrounding mesh nodes. In the simplest case, the entire strength is assigned onto the nearest mesh node. This obviously conserves the zeroth-order moment (total mass), but no higher moments. Higher-order schemes can be derived by solving a linear system of equations for the coefficients of the interpolation polynomial, or by Fourier space methods. The most frequently used interpolation scheme is the M_4' function:



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

$s = \frac{|x|}{h}$ $\quad$ $|x|$: distance of particle from mesh node

Its order of convergence is 3 and it conserves moments up to and including the second moment (in particle-to-mesh interpolation). The M_4' scheme can be used to interpolate from particles to mesh nodes and also vice versa. For each particle, the two neighboring mesh nodes in each direction are considered. For each of those mesh nodes, we compute the distance $|x|$ between the particle and the mesh node and normalize it with the mesh spacing h. Using this value for s we then evaluate the M_4' function to compute the interpolation weight $W \le 1$ for this specific particle-node pair. The portion $W\omega_p$ of the particle's strength ω_p is then added to the mesh node. Doing this for all particles yields the complete field interpolated onto the mesh.

In higher dimensions, the interpolation kernels are Cartesian products of the 1D kernels. We can thus simply use the above scheme in each direction independently and multiply the weights to get the final weight. In 3D, for example, the weight for each particle-node pair would be computed as: $W(x,y,z) = W_x(x)W_y(y)W_z(z)$.

5.3 Remeshing

We have seen that the overlap condition requires the particles to always be closer together than ϵ. If the particles move, it may, however, happen that they evacuate from certain regions of space, where the overlap condition might then be violated. In order to prevent this, the particles are periodically redistributed. This *remeshing* step consists of:

- interpolating the particle strengths to a regular mesh of resolution $h < \epsilon$,
- deleting the old set of particles, and
- creating new particles at the locations of the mesh nodes, carrying the node weights as their new strengths.

Since we are using the above-described moment-conserving interpolation schemes, remeshing does not harm the conservative properties of the method. Moreover, if the order of convergence of the interpolation scheme is at least one higher than the order of convergence of the operator approximation, the convergence properties of the simulation remain unaffected as well.

5.4 Boundary conditions and the method of images

The operator approximations described above only apply in infinite domains. For simulations in constrained geometries, they need to be modified in order to take into account the prescribed boundary conditions. There are two basic types of boundary conditions:

- **Dirichlet:** In a homogeneous[2] Dirichlet boundary condition, the value of the *function* is zero at the boundary, thus $u(\text{boundary}, t) = 0$.

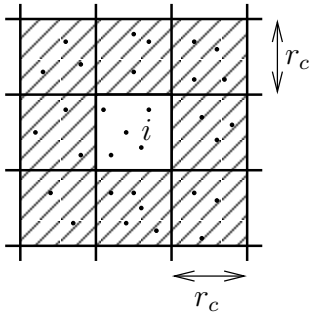
- **Neumann:** In a homogeneous Neumann boundary condition, the value of the *normal derivative* is zero at the boundary, thus $\underline{n} \cdot \nabla u(\text{boundary}, t) = 0$, where $\underline{n}$ is the normal onto the boundary.

For short-range operators and *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. Each of these mirror particles is the mirror image (mirrored at the boundary along the normal $\underline{n}$) of a particle inside the domain. In order to satisfy a homogeneous Dirichlet boundary condition, the strengths of all mirror particles are set to the negative strength of the corresponding real particle. This ensures that when evaluating the interaction kernel, the two strengths cancel at the boundary, leading to $u(\text{boundary}, t) = 0$. For homogeneous Neumann boundary conditions, the strength of the mirror particles are set equal (without sign inversion) to the strength of the corresponding real particle. This ensures that the normal derivative (the gradient in the direction normal to the boundary) vanishes at the boundary. Due to the use of mirror particles, this method is called the *method of images*. The method of images is a general concept that is valid beyond particle methods. It can also be used to analytically solve differential equations in bounded domains by superposition of mirrored solutions. This, however, only works if the equations are linear and the superposition principle is valid. An intuitive interpretation of the method of images is given in Sec. 10.5 for waves.

5.5 Fast Neighbor Lists

Short-range interactions are directly evaluated on the particles and each particle only needs to interact with its neighbors. This renders the method $O(N)$, provided the neighbors are known or can be found in at most $O(N)$ time (recall that there is no connectivity information on particles!). Fast neighbor list algorithms are available to find all neighbors within a cutoff radius r_c of each particle in $O(N)$ time. There are two basic methods: cell lists and Verlet lists.

5.5.1 Cell lists



In cell lists, the particles are sorted into cells of edge length r_c. Each cell stores a list (or linked list) of the particles that are inside it. Construction of these lists for N particles is $O(N)$:

For each particle $p = 1, \ldots, N$ at position $\underline{x}_p = (x_p, y_p, z_p)$:

1. Compute the index of the cell it is in as $(i, j, k) = \left(\lceil \frac{x_p}{r_c} \rceil, \lceil \frac{y_p}{r_c} \rceil, \lceil \frac{z_p}{r_c} \rceil\right)$. This assumes that the cell numering starts from 1. If cells are numbered from 0, the floor should be used.
2. Add the index p to the list of cell (i, j, k).

In 2D, the z component of the position and the index k are absent.

In order to compute all interactions of particle i with its neighbors, we have to consider:

- all particles in the same cell as i, and
- all particles in all adjacent cells (8 is 2D, 26 in 3D).

The evaluation of the interactions of all particles i with all their respective neighbors closer than r_c is thus also $O(N)$.

5.5.2 Verlet lists (Verlet, 1967)

For spherically symmetric interactions in 3D, cell lists contain up to 6 times more particles than actually needed (ratio between the volume of the cube and the inscribed sphere of radius r_c). Verlet lists reduce this overhead by storing an explicit list of all interaction partners on each particle. This can speed up the evaluation of particle-particle interactions by a factor of up to 6. In order not to have to rebuild all lists whenever a particle has moved, we add a safety margin (called "skin") to the cutoff radius. The lists then only need to be updated once any particle has moved farther than the skin thickness. This of course causes additional overhead,

[2] A boundary condition is called homogeneous if its right-hand side, i.e. the value imposed at the boundary, is zero. Imposing non-zero values leads to inhomogeneous boundary conditions that are more difficult to treat.

and makes the Verlet list at most $81/(4\pi(1+\text{skin})^3)$-times smaller than cell lists. Verlet lists are efficiently constructed in $O(N)$ time using intermediate cell lists.

In order to compute particle-particle interactions, we simply loop over all Verlet list entries for each particle. This is also $O(N)$.

5.6 Symmetry

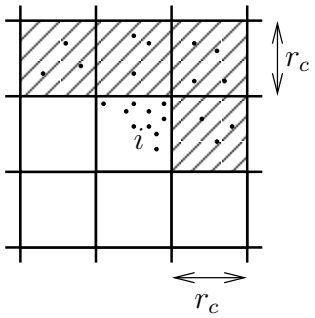
The computational cost of evaluating short-range particle-particle interactions can be reduced further by a factor of at most 2 when evaluating the particle-particle interactions in a symmetric way. Most particle interaction kernels (resulting from operator approximation) are symmetric, i.e., they only depend on the **distance** between two particles and not on their absolute positions. This is a direct consequence of the fact that our models are based on conservation laws. If particle i gives a certain amount of its strength to particle j, particle j has to receive exactly that amount. No strength (e.g., mass) is lost during an interaction. Instead of looping over all particles when computing the interactions, it would thus be sufficient to loop over all unique interaction *pairs* and directly attribute the computed change to *both* interaction partners. In the cell list and Verlet list algorithms as outlined above, each interaction is computed twice and the result only attributed to particle i. We can exploit symmetry by:

- computing η_{ij}, the influence of particle j on particle i, only once for each unique pair (i,j),
- $\omega_i \leftarrow \omega_i + \eta_{ij}$
- $\omega_j \leftarrow \omega_j - \eta_{ij}$ (or + for some quantities)

5.6.1 Symmetric cell lists

In order to ensure that each unique interaction is considered *exactly* once, cell-cell interactions in cell lists have to be done symmetrically. In order to compute all interactions of particle i *that have not yet been accounted for elsewhere*, we thus consider:

- in the same cell where i is: only the particles $(i+1)...N_{\text{cell}}$
- *all* particles in *half* of the neighboring cells as shown below.



The reader is encouraged to verify that this is sufficient and that indeed every interaction is considered in this scheme. In 3D, the scheme remains unchanged in the middle z-plane and we also consider all the interactions with all the cells on the z-plane below (but exclude those above). Figure 5.2a–b summarize the cell-cell interactions in asymmetric and symmetric cell list algorithms.

In Fig. 5.2c, diagonal interactions are introduced in order to further reduce the memory overhead for the boundary layers by 33% in 2D and 40% in 3D. 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: 0–0, 0–1, 0–3, 0–4, and 1–3 in 2D, 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 3D.

5.6.2 Symmetric Verlet list

Symmetric Verlet lists only contain the unique interactions in the first place, as they are built using intermediate symmetric cell lists (either with or without the diagonal interactions). They thus only contain half of the entries and there is no change needed in the way particle-particle interactions are evaluated. The loop still runs over the entire Verlet list.

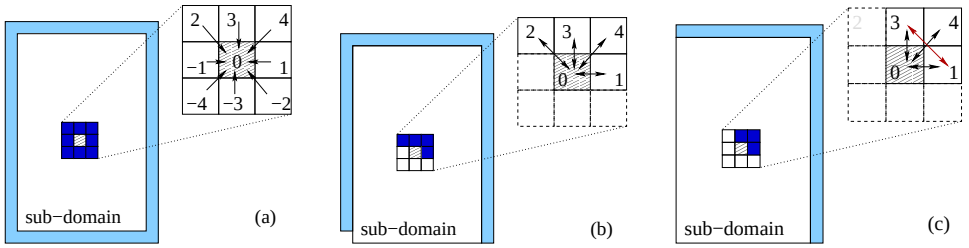


Figure 5.2: Cell-cell interactions in cell list algorithms. (a) For asymmetric interactions, all adjacent cells have to be considered and the interactions are one-sided. (b) In traditional symmetric cell list algorithms, interactions are required on all but one boundary. (c) Introducing diagonal interactions (1–3), the cell layers for the boundary conditions (light blue; cf. Sec. 5.4) 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.

Chapter 6

Diffusion

In this chapter:

- Terminology and the governing equation for diffusion
- A stochastic particle method for simulating diffusion: Random Walk
- A deterministic particle method for simulating diffusion: PSE
- Comparison of PSE and Random Walk

Learning goals:

- Be able to implement and use PSE and Random Walk
- Know about the convergence properties of the two methods
- Be able to correctly name different types of diffusion

Now that we know the modeling framework and the numerical simulation tool, we look at various transport processes that are important in biological systems. For each one, we derive the governing equation using the method of control volumes and then we see how these equations can be simulated using particle methods. Since the function approximation is the same for all cases, we mainly focus on the operator approximation and see what the operator kernels η look like for the different cases. We start with the important and fundamental process of diffusion, modeling passive (not energy-dependent) transport in biology.

6.1 Governing Equation

We have already derived the governing equation for diffusion in its most general form in Section 4.2:

$$\frac{\partial u(\underline{x},t)}{\partial t} = \nabla \cdot \left(\underline{\underline{D}}(\underline{x},t)\nabla u(\underline{x},t)\right) .$$

Depending on the form of the diffusion tensor $\underline{\underline{D}}$ we distinguish different cases:

- **isotropic diffusion:** The diffusion tensor does not depend on the spatial direction and hence reduces to a scalar diffusion constant ν times the identity matrix: $\underline{\underline{D}}(\underline{x},t) = \nu(\underline{x},t)\underline{\underline{\mathbb{1}}}$. In ansitropic diffusion, $\underline{\underline{D}}(\underline{x},t)$ is a full matrix.
- **homogeneous diffusion:** The diffusion tensor does not depend on space and has the same value at every point in space: $\underline{\underline{D}}$ is not a funtion of $\underline{x}$. In inhomogeneous diffusion, the tensor amounts to a different matrix at different locations.
- **normal diffusion:** The diffusion tensor $\underline{\underline{D}}$ is constant over time and thus not a function of t. If $\underline{\underline{D}}$ depends on t, diffusion is called anomalous.

For isotropic, homogeneous diffusion, the diffusion equation simplifies to: $\frac{\partial u}{\partial t} = \nu\Delta u$ with the scalar diffusion constant ν.

This equation describes the spatiotemporal dynamics of the intensive concentration field of a diffusing quantity. It is thus only valid on length scales $\gg \lambda$ and constitutes a macroscopic description. On the microscale, diffusion amounts to Brownian motion of the molecules or real-world particles. Each microscopic particle performs a random walk by selecting the direction of its next step uniformly at random and the length of the next step from a Gaussian distribution. Again, the microscopic and macroscopic descriptions are related through an averaging operation (see Section 2.3). If we average the number of Brownian particles in a control volume for infinitly many, independent random walk trajectories, we recover the diffusion equation for their density. This provides a second way of deriving the diffusion equation, and even the constitutive equation for the flows (i.e., Fick's law) can be derived this way, hence confirming Fick's experimental observations.

6.1.1 Anomalous diffusion

Anomalous diffusion processes play important roles in biology. It is thus worthwhile considering them in a bit more detail. Diffusion processes become anomalous if, e.g.:

- the diffusing molecules engage in chemical reactions that limit their mobility,
- diffusion takes place in a small volume of confinement, such as in a cell organelle, and the diffusing molecules constantly "feel" the presence of the boundaries,

- diffusion is combined with a superimposed deterministic drift or directed motion, e.g. from motor proteins.

While there can also be other reasons for anomalous diffusion, these are probably the most frequent ones in biology. In all these cases, the diffusion tensor depends on time.

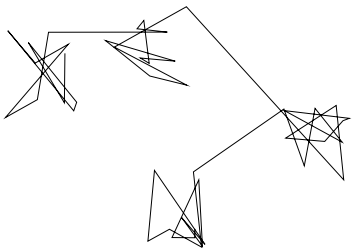
This is probably best understood on the microscopic scale, resolving length scales $< \lambda$. Let $\underline{x}(t)$ be the position along the trajectory of a diffusing particle or molecule. In normal diffusion, the second moment $\langle x(t)^2 \rangle$ is related to the diffusion constant and to time as $\langle x(t)^2 \rangle \propto Dt$. The angle brackets denote averages over a large ensemble of independent trajectories of the microscopic diffusing molecules. Intuitively, this result means that the area that can be covered by a Brownian walker grows linearly with time.

In anomalous diffusion, the second moment is no longer linearly proportional to time, but we rather have the more general form $\langle x(t)^2 \rangle \propto Dt^\alpha$ with $\alpha \neq 1$. This can be decomposed as $\langle x(t)^2 \rangle \propto Dt^{\alpha-1}t$, such that we find the time-dependent diffusion constant $\Gamma(t) = Dt^{\alpha-1}$ as the proportionality constant in a linear-time law.

In real experiments, the values of α and D can, e.g., be determined by single-particle tracking (tracking the motion of a single, fluorescently labeled molecule and explicitly computing the above averages). Depending on the value of the time-scaling exponent α, we distinguish:

$\alpha < 1 \Rightarrow$ subdiffusion. Common reasons for the reduced molecular mobility in subdiffusion are molecular crowding, complex confinement, compartmentalization, or binding reactions.

$\alpha > 1 \Rightarrow$ superdiffusion, mostly due to an overlayed active transport process. Endocytic cargo vesicles in cells frequently behave superdiffusively as their Brownian motion can be combined with intermediate stretches of active transport along microtubules.



Superdiffusion

6.2 Simulations using Random Walk (RW)

The microscopic description of diffusion as Brownian motion can directly be used to construct a first particle method to simulate the dynamics of diffusion processes on the macroscopic scale: the method of Random Walk (RW). In RW, we explicitly simulate the Brownian motion of computational particles. But these particles now live on the macroscopic scale and are thus not necessarily identical to the true physical particles. Each particle carries a certain mass, which never changes (conservation of mass in a Lagrangian control volume). With respect to the function approximation, RW is a point-particle method. Also, the operator discretization only involves moving the particles, making RW a pure particle method (Chorin, 1973).

Since the particles carry mass, RW is an extensive method and, according to Fig. 5.1, we start from the analytical solution of the diffusion equation:

$$u(\underline{x},t) = \int_\Omega G(\underline{x},\underline{y},t)u_0(\underline{y})\,d\underline{y} \quad ;\underline{x},\,\underline{y} \in \mathbb{R}^d\,.$$

In this analytical solution, the kernel G is called *Green's function*. It always exists, even though it is unknown in most cases. For isotropic, homogeneous, normal diffusion, however, the analytical form for G is know to be:

$$G(\underline{x},\underline{y},t) = \frac{1}{(4\pi\nu t)^{d/2}} \cdot \exp\left(-\frac{||\underline{x}-\underline{y}||_2^2}{4\nu t}\right)\,.$$

This is Green's function on the macroscopic scale. It can easily be derived from the microscopic scale. Consider the process of Brownian motion of an individual, microscopic walker. If that particle is at location $\underline{x}_0$ at time t, what is the probability of finding it at location $\underline{x}$ at a later time $t+\delta t$? Since Brownian motion involves random steps sampled from a Gaussian distribution, this probability is given by the Gaussian

$$P(\underline{x}|\underline{x}_0,\delta t) = \frac{1}{(4\pi\nu\delta t)^{d/2}} \cdot \exp\left(-\frac{||\underline{x}-\underline{x}_0||_2^2}{4\nu\delta t}\right)\,.$$

This is the *transition density* of the discrete stochastic process on the microscale. If we let the number of independently moving Brownian particles go to infinity, the transition density becomes identical to Green's function on the continuous macroscale. This follows from the central limit theorem of probability.

The numerical point-particle method of RW uses the above equivalence between the microscale and the macroscale to simulate the continuous diffusion equation with a randomized algorithm:

Particles

location $\underline{x}_p(t)$

strength $\omega_p = V_p u_0(\underline{x}_p^0) = \text{const} \Rightarrow$mass

Dynamics

$$\begin{cases} \dfrac{d\underline{x}_p}{dt} = \underline{\mathcal{N}}(0, 2\nu) \\ \dfrac{d\omega_p}{dt} = 0 \end{cases}$$

According to the above argument, this probabilistic dynamics converges to the analytical integral solution of the diffusion equation for the number of particles $N \to \infty$. RW thus amounts to a Monte Carlo integration of Green's function solution.

Note that there are two different ways of sampling the next step in a random walk: (a) sample the x, y, (and z) displacements separately, or (b) sample a random direction and then a step length along this direction. In the first scheme, Δx, Δy, (and Δz) are independently sampled from a 1D Gaussian with variance $2\nu\delta t$. In the second scheme, the direction is uniformly sampled on the unit half-circle (by uniformly sampling the polar angle between 0 and π) or the unit hemi-sphere (by sampling polar and azimuthal angles as shown below). The step length is then sampled from a 1D Gaussian with variance $2d\nu\delta t$. Since the step length can be positive or negative, the direction is only sampled on the half-circle (-sphere). Notice the difference between the two schemes: in the second one, the space dimension d occurs as a factor in the variance while it does not in the first one. How come? It is easy to see from the Pythagorean theorem for right-angled triangles. Taking steps of length (standard deviation) $\sqrt{2\nu}$ along every coordinate axis leads us a distance of $\sqrt{2d\nu}$ away from the origin. The algorithm below uses the second sampling scheme since uniformly distributed random numbers are cheaper to generate on a computer than Gaussian random numbers.

Algorithm Random Walk in space $\mathbb{R}^3$ ("continous random walk")

1. initialize

$$\begin{aligned} \underline{x}_p^0 &\leftarrow \underline{x}_0(p) \\ \omega_p^0 &\leftarrow V_p u_0(\underline{x}_p^0) \qquad \forall p = 1 \dots N \end{aligned}$$

2. loop $\forall p$:

   - choose a random direction from uniformly distributed points on the unit semi-sphere :

$$\varphi = \pi\mathcal{U}(0,1); \quad \vartheta = \operatorname{asin}(2 \cdot \mathcal{U}(0,1) - 1) + \frac{\pi}{2}$$

   - choose the step lenght as $s \sim \mathcal{N}(0, 2d\nu\delta t)$ for simulation time step size δt
   - move the particles: $\underline{x}_p^{n+1} \leftarrow \underline{x}_p^n + s \begin{pmatrix} \sin\vartheta\cos\varphi \\ \sin\vartheta\sin\varphi \\ \cos\vartheta \end{pmatrix}$

3. Advance time by δt

4. Go to (2) until the final time is reached.

The RW method advances point particles that conserve their mass. In order to reconstruct the approximate intensive concentration field, the particles need to be binned. One subdivides the space into small volumes. The concentration in each volume is then given by the total mass of all particles inside it, divided by the size of the volume. These Eulerian binning control volumes have to be of a minimal size in order to be in the continuum region and contain enough particles for a stable approximation of the average mass (concentration). This recovers a piecewise constant approximation of the concentration field.
In 1D, the interval of solution could, for example, be subdivided into M disjoint intervals of size $\delta x = X/M$ and the RW particles are binned in these intervals as follows: each interval $j = 1, \dots, M$ is assigned the sum of the strengths of all the particles having positions between $(j-1)\delta x$ and $j\delta x$, thus

$$u^{\mathrm{RW}}((j-1/2)\delta x, n\delta t) = \frac{1}{\delta x} \sum_p \left\{ \omega_p : (j-1)\delta x < x_p^n \leqslant j\delta x \right\} \tag{6.1}$$

for $j = 1, \dots, M$.

RW is the simplest method to simulate diffusion processes and as such it is widely used and known. It is, however, important to be aware of its advantages and limitations.

Advantages:

- RW is very easy to implement.
- RW readily extends to anomalous and anisotropic diffusion by changing the step probability density from a normal distribution to a multivariate normal distribution or to a distribution that depends on the location $\underline{x}$.
- RW extends to diffusion on curved surfaces by projecting all step diplacements onto the surface.
- RW extends to combined convection-diffusion problems by superimposing the convective particle displacements and the RW displacements.

Disadvantages:

- Due to its Monte Carlo character, RW converges slowly. The simulation error decreases with increasing particle number N as $O(N^{-1/2})$. We thus need to simulate a large number of trajectories in order to get reasonably close to Green's function solution. Notice that this slow convergence is imposed by the variance of the central limit theorem. This theorem is the very foundation of RW and there is no way to improve the situation.

- The solution accuracy deteriorates with increasing ν as the sample variance grows.
- The solution accuracy also deteriorates for small $\nu \ll \delta x^2/\delta t$ as the particle motion becomes masked by the binning noise. This is a direct consequence of RW being a point-particle method. In order to recover the concentration field at the end of a simulation, we have to average over Eulerian control volumes. For very small ν, however, particles would never move to another control volume and the final solution would appear identical to the initial condition.
- If the particles move in a bounded domain, we must check for collisions with the boundary at every time step and for each particle. This is computationally expensive.

6.3 Simulations using Particle Strength Exchange (PSE)

PSE is a deterministic pure particle method to simulate diffusion in the continuum (macroscopic) description. The method was introduced by Degond and Mas-Gallic in 1989 and it is based on a deterministic integral approximation of the diffusion operator. Moreover, PSE uses a smooth particle function approximation, which allows recovering the field values everywhere in space without the averaging (binning) needed in RW. Since PSE is a pure particle method, we look for an integral operator approximation with a certain kernel η.

Isotropic Homogeneous Diffusion

We start with the simplest case of isotropic, homogeneous diffusion, where we want to approximate the Laplacian on scattered particle locations such that mass is conserved. For simplicity, we consider the 1D case. The derivation in n dimensions is analogous.
In 1D, the diffusion equation is

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}\,.$$

We start by expansion of the concentration field $u(y,t)$ into a Taylor series around point x:

$$u(y) = u(x) + (y-x)\frac{\partial u}{\partial x} + \frac{1}{2}(y-x)^2\frac{\partial^2 u}{\partial x^2} + \frac{1}{6}(y-x)^3\frac{\partial^3 u}{\partial x^3} + \dots\,.$$

We then substract $u(x)$ on both sides, multiply with the kernel η_ϵ and integrate over the entire domain of solution Ω in order to arrive at an integral operator approximation:

$$\begin{aligned}\int_\Omega (u(y)-u(x))\eta_\epsilon(y-x)dy = &\int_\Omega (y-x)\frac{\partial u}{\partial x}\eta_\epsilon(y-x)dy \\ &+ \frac{1}{2}\int_\Omega (y-x)^2\frac{\partial^2 u}{\partial x^2}\eta_\epsilon(y-x)dy \\ &+ \frac{1}{6}\int_\Omega (y-x)^3\frac{\partial^3 u}{\partial x^3}\eta_\epsilon(y-x)dy + \dots\end{aligned}$$

The term we want is the $\frac{\partial^2 u}{\partial x^2}$ on the right-hand side. We thus design the kernel $\eta_\epsilon = \epsilon^{-1}\eta(x/\epsilon)$ such that this term is the only one remaining on the right-hand side, up to a certain order r. This requires that (we do the change of variables $z = (y-x)/\epsilon$ in order to go from η_ϵ to the scale-invariant η):

- η be even $\Leftrightarrow$ all integrals over odd powers vanish
- $\int z^2\eta(z)dz \stackrel{!}{=} 2 \Leftrightarrow$ second term becomes $\frac{\partial^2 u}{\partial x^2}\cdot\frac{1}{2}\cdot 2\cdot\epsilon^2$
- $\int z^s\eta(z)dz \stackrel{!}{=} 0 \quad \forall\, 2 < s \leq r+1 \Leftrightarrow$ higher-order terms vanish up to order $r+1$

Using such an η, the only terms remaining are:

$$\Rightarrow \int_\Omega (u(y)-u(x))\eta_\epsilon(y-x)dy = \frac{\partial^2 u}{\partial x^2}\epsilon^2 + O(\epsilon^{r+2})\,.$$

The factor ϵ^2 comes from the fact that we are computing a second derivative (compare Eq. 5.3). We now solve this equation for the desired term, which is the righ-hand side of the diffusion equation:

$$\Rightarrow \frac{\partial^2 u}{\partial x^2} = \frac{1}{\epsilon^2}\int_\Omega (u(y)-u(x))\eta_\epsilon(y-x)dy + O(\epsilon^r)\,.$$

This is the integral operator approximation to the 1D diffusion operator. Any kernel function η that fulfills the above three conditions (called "moment conditions") can be used.
The next step is to **discretize** this integral operator as a quadrature (sum) over the set of N particles, thus:

$$\frac{\partial^2 u^h}{\partial x^2}(x_p^h) = \frac{1}{\epsilon^2}\sum_{q=1}^{N} V_q(u_q^h - u_p^h)\eta_\epsilon(x_q^h - x_p^h)\,.$$

If all particles have the same volume, the difference in the first parenthesis can simply be computed over the strengths ω. This is the discrete, extensive form of the diffusion operator. In n dimensions, the operator looks exactly the same. Even

the pre-factor ϵ^{-2} remains the same because it comes from the order of the approximated differential operator and not from the dimension. The only thing that changes is that a different η has to be used, namely one that satisfies the above moment conditions in nD, for the respective n.

If we assume that all particle volumes are the same, i.e., $V_p = V_q = V$, the final PSE method can be formulated in terms of particle strength:

Particles:

location $\underline{x}_p$
strength $\omega_p(t) = Vu(\underline{x}_p, t)$

Dynamics:

$$\begin{cases} \dfrac{d\underline{x}_p}{dt} = \underline{0} \\ \dfrac{d\omega_p}{dt} = \dfrac{V\nu}{\epsilon^2} \displaystyle\sum_{q=1}^{N} (\omega_q - \omega_p)\eta_\epsilon(\underline{x}_q - \underline{x}_p)\,. \end{cases}$$

If particles have different volumes, then they need to be explicitly carried as an additional particle property and strengths accordingly normalized. This, however, is rarely the case in applications where one usually places particles regularly/uniformly for a PSE diffusion simulation.
PSE is dual to RW. While in PSE the particles do not move, but exchange strength (hence the name), they move in RW, but keep their mass.
Duality to RW

RW		**PSE**
$x_p(t)$	$\rightleftharpoons$	$x_p = \not{c}$
$\omega_p = \not{c}$	$\rightleftharpoons$	$\omega_p(t)$
microscopic: Brown		macroscopic: Fick

Both methods have an intuitive interpretation. While it was the microscopic Brownian motion for RW, it is Fick's law of diffusion for PSE. Fick's law states that in diffusion, mass is flowing against the concentration gradient. This is exactly what the PSE operator does: the first parenthesis computes the concentration gradient between a pair of interacting particles whereas the kernel η converts this gradient into a flux of mass depending on the distance between the two particles.

The advantages and limitations of PSE are:

Advantages

- PSE can be arbitrarily accurate depending on how many moments of η vanish.
- Since diffusion is a short-range process, η has local support and the sum can be evaluated in $O(N)$ time using cell lists or Verlet lists.
- PSE can easily be extended to convection-diffusion problems by moving the particles according to the convective velocity field.
- In the absence of convection, all geometry and boundary processing only needs to be done once because the particles don't move.

Disadvantages

- One needs to derive ("engineer") good kernels η.
- The implementation is complicated by the need for cell lists or Verlet lists.

Example 6.3.1 (2^{nd}-order accurate kernels)**.**

$$\eta_\epsilon(x) = \frac{1}{2\epsilon\sqrt{\pi}} e^{-\frac{x^2}{4\epsilon^2}} \qquad x \in \mathbb{R}$$

$$\eta(\underline{x}) = \frac{15}{\pi^2} \frac{1}{|\underline{x}|^{10} + 1} \qquad \underline{x} \in \mathbb{R}^3$$

The first kernel is a Gaussian, which naturally follows from the transition density (Green's function) of diffusion being a Gaussian. The second example shows that there can also be other kernels that fulfill the moment conditions. Polynomial kernels such as the one here are computationally more efficient than Gaussians because we don't need to evaluate an exponential function.

As a side note, it is instructive to see that PSE is not the only possibility of deriving an integral operator approximation for the Laplacian. Another possibility would be to simply take the smooth particle function approximation

$$u_\epsilon^h(x) = \sum_{p=1}^{N} \omega_p^h \zeta_\epsilon(x - x_p^h)$$

and compute the righ-hand side of the (1D) diffusion equation directly as

$$\frac{\partial^2 u_\epsilon^h(x)}{\partial x^2} = \frac{\partial^2}{\partial x^2} \sum_{p=1}^{N} \omega_p^h \zeta_\epsilon(x - x_p^h) = \sum_{p=1}^{N} \omega_p^h \frac{\partial^2 \zeta_\epsilon}{\partial x^2}(x - x_p^h)\,.$$

This operator approximation is older than PSE and is known as *Fishelov's scheme*. While it might seem more straightforward than the PSE method, the resulting operator approximation is not conservative. The additional freedom in PSE to choose a different kernel η for the operator approximation than the kernel ζ used for the function approximation allows conserving mass exactly. In light of the fact that our models are based on conservation laws, this is a clear advantage.

In order to understand this, consider two particles p and q. In PSE, the amount of mass transferred from p to q is given by

$$\Delta\omega_{p\to q} = (\omega_p - \omega_q)\eta_\epsilon(\underline{x}_p - \underline{x}_q)$$

and the amount of mass transferred from q to p is

$$\Delta\omega_{q\to p} = (\omega_q - \omega_p)\eta_\epsilon(\underline{x}_q - \underline{x}_p)\,.$$

Since the kernel η is symmetric, it is $\Delta\omega_{p\to q} = -\Delta\omega_{q\to p}$. Hence, the mass given by particle p to particle q exactly equals the amount of mass received by particle q from particle p. Mass is thus conserved as nothing is lost or created. The interactions are symmetric. As shown in Sec. 5.6, this can also be exploited to reduce the computational cost of the simulation algorithm by a factor of 2.
In Fishelov's scheme, we have

$$\Delta\omega_{p\to q} = \omega_q \frac{\partial^2\zeta_\epsilon}{\partial x^2}(\underline{x}_p - \underline{x}_q)$$

and

$$\Delta\omega_{q\to p} = \omega_p \frac{\partial^2\zeta_\epsilon}{\partial x^2}(\underline{x}_q - \underline{x}_p)$$

and hence $\Delta\omega_{p\to q} \neq -\Delta\omega_{q\to p}$. Not all the mass given away by particle p is thus received by particle q and vice versa. The scheme does not conserve mass because the interactions are not symmetric. This also prevents the use of fast symmetric cell or Verlet lists.

Anisotropic Inhomogeneous PSE

So far we have focused on isotropic and homogeneous diffusion for simplicity. The same derivations, however, can also be made for anisotropic and inhomogeneous diffusion, where $\underline{\underline{D}}$ is a full matrix. We then need to find an integral approximation to the operator $\nabla\cdot(\underline{\underline{D}}\nabla)$ rather than the Laplacian Δ. In n dimensions, this leads to the integral operator approximation:

$$\nabla\cdot(\underline{\underline{D}}\nabla u) \approx Q_\epsilon u(\underline{x},t) = \epsilon^{-2}\int_\Omega (u(\underline{y}) - u(\underline{x}))\sigma_\epsilon(\underline{x},\underline{y},t)d\underline{y}$$

(we skip the details of the derivation because there is nothing conceptually new) and the PSE scheme remains

$$\begin{cases} \dfrac{d\omega_p}{dt} = \dfrac{V_p}{\epsilon^2}\displaystyle\sum_{q=1}^{N}(\omega_q - \omega_p)\sigma_\epsilon(\underline{x}_p,\underline{x}_q,t) & \text{for } V_p = V_q \\ \dfrac{d\underline{x}_p}{dt} = \underline{0}\,. \end{cases}$$

The operator kernel σ is now a bit more complicated and has the form

$$\sigma_\epsilon(\underline{x}_p,\underline{x}_q,t) = \underbrace{\epsilon^{-2}\overline{\eta}_\epsilon(\underline{x}_p - \underline{x}_q)}_{\text{isotropic part}} \quad \underbrace{\sum_{ij=1}^{d}\underline{\underline{M}}_{ij}(\underline{x}_p,\underline{x}_q,t)(\underline{x}_p - \underline{x}_q)_i(\underline{x}_p - \underline{x}_q)_j}_{\text{anisotropic}}$$

While the isotropic part of the operator (NOT of $\underline{\underline{D}}$!) looks analogous to isotropic PSE, there is a second part, which depends on the space directions i and j. It contains the mapping tensor $\underline{\underline{M}}$, which maps distance to strength in a direction-dependent way (before this was just the scalar η). In order for the method to conserve mass, $\underline{\underline{M}}$ must be symmetric, such that $\underline{\underline{M}}(\underline{x}_p,\underline{x}_q) = \underline{\underline{M}}(\underline{x}_q,\underline{x}_p)$ for any pair of interacting particles p and q. The simplest way to ensure this is to set:

$$\underline{\underline{M}}(\underline{x}_p,\underline{x}_q,t) = \frac{1}{2}(\underline{\underline{m}}(\underline{x}_p,t) + \underline{\underline{m}}(\underline{x}_q,t))\,,$$

where $\underline{\underline{m}}$ is related to the diffusion tensor as:

$$\underline{\underline{m}}(\underline{x},t) = \underline{\underline{D}}(\underline{x},t) - \frac{1}{d+2}\mathrm{Tr}(\underline{\underline{D}}(\underline{x},t))\cdot\underline{\underline{\mathbb{1}}}\,.$$

Subtracting the trace of the diffusion tensor from itself leaves the anisotropic part. This is correct because the isotropic part has already been accounted for in the pre-factor to the sum in σ.

Example 6.3.2 (radially symmetric isotropic kernel $\overline{\eta}(r)$)**.**

$$\overline{\eta}_\epsilon(\underline{x}_p - \underline{x}_q) = \frac{4}{\epsilon^3\pi\sqrt{\pi}}e^{-\frac{||\underline{x}_p - \underline{x}_q||_2^2}{\epsilon^2}} \qquad \text{in } \mathbb{R}^3\,.$$

This kernel is second-order accurate as it fulfills the moment conditions for $r = 2$.

6.4 Comparison of PSE and RW

We compare the accuracy of the RW and PSE methods using a benchmark case of isotropic homogeneous diffusion on the one-dimensional ($d = 1$) ray $\Omega = [0,\infty)$, subject to the following initial and boundary conditions:

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

The exact analytic solution of this problem is

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

Both RW and PSE simulations of this benchmark case are performed with varying numbers of particles in order to study spatial convergence. The boundary condition at $x = 0$ is satisfied using the method of images (see Sec. 5.4).
For the PSE we use the 2^{nd} order accurate Gaussian kernel

$$\eta_\epsilon(x) = \frac{1}{2\epsilon\sqrt{\pi}} e^{-x^2/4\epsilon^2}\,, \tag{6.2}$$

which fulfills the moment conditions in one dimension to order $r = 2$. The concentration values at particle locations x_p and simulation time points $t_n = n\delta t$ are recovered as

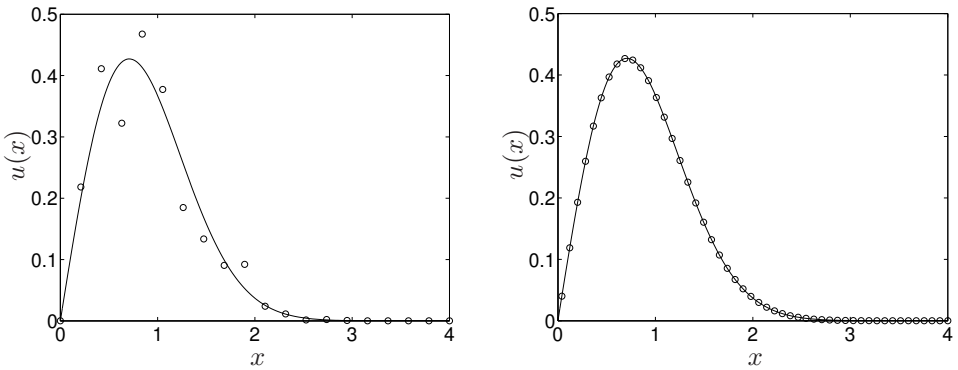
$$u^{\text{PSE}}(x_p, t^n) = \omega_p^n \cdot N/X\,.$$

For RW, the binning described in Eq. 6.1 is used to recover the intensive concentration field.

Figure 6.1 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 $\nu = 10^{-4}$. The accuracy of the simulations for different numbers of particles is assessed by computing the final L_2 *error*

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

for each N. The resulting convergence curves are shown in Fig. 6.2.



(a) RW (b) PSE

Figure 6.1: Comparison of RW (a) and PSE (b) solutions of the benchmark case. The solutions at time $T = 10$ are shown (circles) along with the exact analytic solution (solid line). For both methods $N = 50$ particles, a time step of $\delta t = 0.1$, and $\nu = 10^{-4}$ are used. The RW solution is binned in $M = 20$ intervals of $\delta x = 0.2$. For the PSE a core size of $\epsilon = h$ is used.

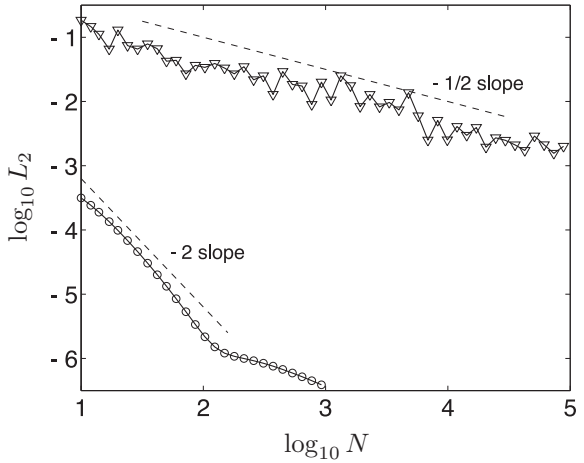


Figure 6.2: Convergence curves for RW and PSE. The L_2 error versus the number of particles for the RW (triangles) and the PSE (circles) solutions of the benchmark case at time $T = 10$ are shown. For both methods a time step of $\delta t = 0.1$ and $\nu = 10^{-4}$ are used. The RW solution is binned in $M = 20$ intervals of $\delta x = 0.2$ and for the PSE a core size of $\epsilon = h$ is used. The machine epsilon is $\mathcal{O}(10^{-6})$.

For RW we observe the characteristic slow convergence of $O(1/\sqrt{N})$. For PSE, a convergence of $O(1/N^2)$ is observed, in agreement with the employed 2^{nd} order kernel function. Below an error of 10^{-6}, machine precision is reached (the simulations were done in single precision, which yields 23 bits of significant precision). It can be seen that the error of a PSE simulation 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.

Chapter 7

Reaction-Diffusion

In this chapter:

- The governing equation for reaction-diffusion
- Stochastic and deterministic simulation of chemical reactions
- Moving fronts and Turing patterns in reaction-diffusion systems

Learning goals:

- Be able to simulate reaction-diffusion systems using PSE and Random Walk
- Know about the existence of Turing patterns and their importance in biology
- Know the deterministic and stochastic (SSA) way of simulating chemical reactions
- Be able to derive dimensionless groupings for reaction-diffusion systems

Reaction-diffusion systems are particularly important in biology as most molecular processes amount to biochemical reactions in some spatial compartment and the reactants and products of the biochemical reactions are mostly transported by diffusion. This means that the concentration fields of different chemical species are coupled via species conversion by the reactions.

7.1 Governing Equation

The governing equation for N different species with respective concentration fields u_i and diffusion tensors $\underline{\underline{D}}_i$ is given by a diffusion equation with a source term:

$$\frac{\partial u_i}{\partial t} = \nabla \cdot (\underline{\underline{D}}_i \nabla u_i) + f_i(\underline{u}) \quad \forall i = 1 \dots N \,.$$

The source terms f_i describe the rate of creation/consumption of species i by reactions. Note that since each concentration field u_i is governed by such an equation (with possibly different reaction terms f_i), this describes a system of coupled PDEs. This equation is called the *Fisher-KPP equation*, named after R.A. Fisher and A.N. Kolmogorov, I.G. Petrovsky, and N.S. Piskunov, who independently formulated the same theory in 1937.

7.2 Simulation

Since the spatial part of the Fisher-KPP equation is a diffusion equation, we can use the methods of the previous chapter to simulate it using particles.

Particles: locations $\underline{x}_p(t)$
strengths $\underline{\omega}_p(t)$; $\omega_{p,i} = V_p u_i$

Since we are now solving a system of coupled PDEs, the particle strengths are vectors with one entry for each of the N chemical species. We observe the following properties:

- Diffusion of all components u_i is independent;
- Reactions are source terms in the corresponding diffusion equation.

This means that the methods of PSE and RW work unchanged for each u_i. In PSE we only need one set of particles since each particle supports the concentration values of all species in a vector $\underline{\omega}_p$. If we used different sets of particles for the different species, we would need to evaluate the smooth particle function approximation whenever we need to recover the entire concentration vector $\underline{u}$ at any point in space in order to evaluate the reaction term $f_i(\underline{u})$ there. In RW, we need to use different sets of particles for the different species if they have different diffusion constants, so different particles can move differently. The binning takes care of reconstructing concentration values at coinciding locations so we can execute the reactions locally in each bin. This means that binning is required after each time step when using RW to simulate diffusion in a reaction-diffusion model.

We also observe that the reaction terms $f_i(\underline{u}(\underline{x}_p))$ only depend on $\underline{u}$ at the location of the corresponding particle. Reactions are thus purely local operations if all concentrations are stored in the same strength vector. This is a direct consequence

of the fact that particles correspond to infinitesimal control volumes and as such constitute homogeneous reaction spaces with no spatial gradients within a single particle. (Recall that the sizes of the particles are $\ll L$.)

7.2.1 Evaluating the reaction terms

The reaction (source) terms f_i can be evaluated using either:

- continuous models (based on chemical mass-action kinetics) → ODE
- or stochastic models (based on molecular collisions) → SSA (Stochastic Simulation Algorithm, D. T. Gillespie, 1976).

While the continuous model is generally simpler to compute, it is only valid if the number of molecules of each species is large (comparable to or larger than Avogadro's number) within each particle. Otherwise there might not be enough molecules of each species inside the control volume that corresponds to the particle and the discrete nature of individual molecules becomes important.

We now describe how to evaluate the reaction terms using either of the two methods:

using ODEs:
Inside a particle there is no diffusion since particles constitute homogeneous reaction spaces. On each particle, we thus have the temporal reaction dynamics

$$\left.\frac{du_i}{dt}\right|_p = f_i(\underline{u}(\underline{x}_p, t))\,,$$

where f_i is a known explicit algebraic function. We can thus evaluate f_i on each particle using the local value of $\underline{u}$ (or $\underline{\omega}$) and add this change to the $\delta\underline{\omega}$ from PSE. In order to do so, the intensive change in concentration given by f_i must first be translated to a change in the extensive strength (mass) of the particle. This is done by multiplying with the volume of the particle. In PSE, we can then use the same time integrator for both the diffusive part of the change of strength and the reactive part, which means that we have to pay attention to the time step stability condition!

using SSA:
SSA operates on molecule numbers (note that this is an extensive quantity). In order to convert the mass carried by particles to molecule number, we use the molar mass of the chemical species:

1. compute the "molecular weight" M as the number of molecules per unit mass ($[M] = \frac{\#}{M}$).
2. convert mass (strength) to number of molecules using this factor
3. apply SSA to compute the local change in molecule number
4. convert back to obtain the corresponding change of mass (strength).

The change of mass is then added to the $\delta\underline{\omega}$ from PSE. Alternatively, the particles could also directly store the molecular population in a vector of integers, since this is an extensive quantity, too. But then the conversion needs to appropriately happen for the PSE step.

7.2.2 SSA in a nutshell

Gillespie's SSA amounts to an exact sampling of trajectories from the chemical master equation and is, as such, rigorous. It is based on two quantities:

h: the number of possible collision pairs between the reactant molecules. For a binary reaction of species a and b, this is: $h = X_a X_b$, where X_i denotes the number of molecules of species i that are inside the current particle. This number is called the "reaction degeneracy".

c: the probability that a reaction occurs *given* a collision has occured. This is called the *specific probability rate* of the reaction and it is connected to the reaction's kinetic rate constant k as: $k = MVc$, where V is the particle volume. The specific probability rate is a constant property of the reaction that can be measured or looked up in tables.

The product $a = hc$ is called the *reaction propensity*. It is defined for each reaction μ and changes over time as the molecule numbers X change. The propensity of a reaction is proportional to the probability of that reaction happening. In addition, the expected waiting time until that reaction happens next is given by an exponential distribution with exponential time constant $1/a$ (the higher the propensity of a reaction, the lower the expected time until it happens again). This is a fundamental result from statistical physics. SSA uses these facts in order to advance the reaction system from reaction event to reaction event. This is done in two steps: first, the index μ of the next reaction is determined, i.e., which reaction out of all possible ones will occur next inside a given particle. Then, the time τ until this next reaction is sampled. Since SSA is a stochastic method, both steps involve random numbers. Gillespie's original Direct Method (DM) consists of the following steps:

1. at $t \leftarrow 0$, initialize $\underline{X}$, a_μ, and the total propensity $a = \sum_\mu a_\mu$
2. Sample μ: generate a random number r_1 from a uniform distribution $\mathcal{U}(0,1)$ and determine μ as the smallest integer satisfying $r_1 < \sum_{\mu'=1}^{\mu} a_{\mu'}/a$
3. Sample τ: generate a random number r_2 from $\mathcal{U}(0,1)$ and compute τ as the real number satisfying $r_2 = 1 - \exp(-a\tau)$, thus $\tau = -a^{-1}\ln(r_2)$
4. Update: $\underline{X} \leftarrow \underline{X} + \boldsymbol{\nu}_\mu$, where $\boldsymbol{\nu}_\mu$ is the stoichiometry of reaction μ, i.e., the change in molecule numbers caused by this reaction; recompute all a_μ and a

5. $t \leftarrow t + \tau$, go to step 2

The algorithm contains several sums over all reaction indices μ and every time a reaction has fired, all propensities are recomputed. It is thus easy to see that the computational cost of the algorithm is linearly proportional to the total number of reactions in the system one simulates. Step 2 amounts to a search for the reaction to happen next. Since the probability of each reaction happening is proportional to its propensity, one can imagine a collection of boxes, each of which having floor area a_μ and representing one reaction. These boxes are arranged in an array of total area $a = \sum_\mu a_\mu$, the total propensity of the whole reaction network. Sampling the next reaction can then be visualized as throwing a ball into the array of boxes. The probability for the ball to land in a given box is proportional to how much of the total area this box occupies, hence a_μ/a. Step 2 of the algorithm now states that one should visit boxes until one has found the one containing the ball (symbolized by the random number r_1). The simplest algorithm to do so is linear search, where the boxes are visited one after the other until the one containing the ball has been found.
Alternatively, one could also first compute the expected waiting times for all reactions, $\tau_\mu = -a_\mu^{-1} \ln(r_2)$ using independent random numbers for each reaction. Then, one could sort these times and pick the reaction with the shortest waiting time to happen next. While this algorithm is formally equivalent with the one above (it samples from the same chemical master equation), it is in general less efficient. The reason is that is uses as many random numbers as there are reactions in the system, whereas the algorithm above only uses 2 random numbers.

7.2.3 Overall algorithm

The overall algorithm for simulating reaction-diffusion systems using particles and PSE as a diffusion solver thus becomes:
$\forall$ time steps $n = 1 \dots T$, compute:

1. $\delta\underline{\omega}_1^n$ due to reactions (using ODE or SSA) on the current strengths $\underline{\omega}^n$. Do not forget to properly convert strength to molecule numbers or concentrations, and back.

2. compute $\delta\underline{\omega}_2^n$ due to diffusion using PSE on the current strengths $\underline{\omega}^n$.

3. compute the total change of strength $\delta\underline{\omega}^n = \delta\underline{\omega}_1^n + \delta\underline{\omega}_2^n$.

4. time integration: $\underline{\omega}^{n+1} \leftarrow \underline{\omega}^n + F(\delta\underline{\omega}^n)$. Use time step δt if using ODEs for the reactions, else use SSA time step τ.

When using SSA, the simulation advances in an event-driven manner from reaction to reaction, where a reaction can only happen in one particle at a time. There is, thus, no fixed time step δt as when using ODEs. Instead, the time step τ is different in each iteration. When advancing the particle strengths and positions forward in time, the same time step must be used. SSA only executes a single reaction (the one with the minimum τ) in one of the particles at each iteration. This is then followed by a diffusion step across all particles using the time step τ of the executed reaction. Then, SSA selects another reaction in possibly another particle. In order to select which particle hosts the reaction to be executed next, we use SSA over the *total propensities* of the particles, i.e., the sum of all reaction propensities within each particle. We thus first use an SSA sampling step to find the particle in which the next reaction will occur, then use SSA inside that particle to find out which reaction occurs, then execute the reaction in that particle, and then advance diffusion on all particles by the same τ of that executed reaction. In cases where diffusion is slow compared to the reactions, it is also possible to execute many SSA steps, over multiple particles, before then doing a single diffusion step over the sum of the τ's.
While technically all 4 combinations of ODE–SSA/PSE–RW are possible, not all of them are equally valid in all situations, and the above algorithm would need to be extended by binning (remeshing) in each time step when using RW for diffusion). Which combination one should use depends on the length scales in the model. The combination ODE–PSE amounts to a pure continuum deterministic model. It is thus valid for scales $\gg \lambda$ and for abundant molecule numbers (when the notion of concentration makes sense). The combination SSA–RW is a discrete stochastic model and should be used when molecules occur with low copy number or a concentration field cannot be defined (discrete model). The combination ODE–RW defines a continuum stochastic model, where the diffusion of the continuous concentration fields is simulated using RW as a Monte Carlo integrator for the governing diffusion equation. It is valid for length scales $\gg \lambda$ and abundant molecules. Due to its stochastic character, it is, however, usually less accurate than the combination ODE–PSE (see also Section 6.4), but easier to generalize to complex geometries and diffusion types. Finally, the combination SSA–PSE relies on the existence of smooth concentration fields where the diffusion equation is valid (hence length scales $\gg \lambda$), but allows for small numbers of molecules within each particle. This is a computationally expensive combination. Since particles can mostly not be empty (otherwise PSE is not a good choice), reactions happen frequently and hence the time steps are small. At the expense of an additional error, diffusion steps can also be done only every n reactions with $n > 1$. This is particularly a good approximation if the time scale of the reactions is much faster than the time scale of the diffusion in the system.

7.3 Physical Behavior

Much of the biophysical significance of reaction-diffusion systems comes from the fact that they can exhibit two types of non-trivial behavior:

1. traveling concentration fronts (similar to waves)

2. inhomogeneous stationary concentration distributions ("Turing patterns").

In particular Turing patterns have attracted great attention both from experimentalists and theoreticians. They are counter-intuitive. Diffusion tends to average out concentration differences such that one would expect complete homogeneous mixing for infinitely long times. In the presence of reactions, however, the stationary solution at infinite time may exhibit stable spatial patterns that are created by a non-equilibrium steady state of the diffusion and the disturbances from the reactions (concentration constantly being produced at certain locations and then diffusing from there). In biology, Turing patterns and traveling waves are used to model phenomena such as:

- morphogenesis
- pattern formation (e.g., animal fur coats or butterfly wing patterns)
- cell motility
- signaling

For most of these pattern-forming processes, a reaction-diffusion system can be found that exhibits these patterns as its steady-state solution. In most cases, however, these models are phenomenological. This means that they generate correct-looking patterns, but the chemical species and reactions in the model do not necessarily correspond to real, experimentally identifyable molecules and reactions. Nonetheless, such models are valuable to study the general principles of pattern formation or to generate biologically-looking patterns that are then used in other models.

7.3.1 An example with moving fronts

Consider the simple binary reaction $a + b \xrightarrow{k} 2a$. Identify the concentration of a with the variable $u = [a]$. Notice that we only need one variable since $[b] = 1 - u$ due to conservation of mass if we initially normalize the total amount of mass in the system to 1. From mass-action kinetics we can derive the reaction term as: $f = ku(1-u) = \underbrace{ku}_{\text{production}} - \underbrace{ku^2}_{\text{consumption}}$. The species a is thus both produced and consumed, albeit at different rates. Let us also assume that a and b diffuse isotropically and homogeneously with identical diffusion constant D. The overall governing equation then becomes:

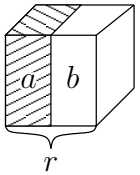
$$\frac{\partial u}{\partial t} = D\Delta u + ku - ku^2 \,.$$

Use SSA
If we simulate the reactions using SSA, we define:

$X_{a,b}$: the number of molecules of a and b, respectively, contained in any given particle

$M_{a,b}$: "molecular weights", i.e. the number of molecules per unit mass for each of the species. For simplicity we assume that $M_a = M_b = M$.

Setup: We simulate the system in a cube of edge length r, initially filled with a in one half and b in the other half.



As the reaction-diffusion process evolves, b will gradually be "eaten up" by a. At steady state, the whole cube will be filled with a and all of b will have been consumed. The plane that separates a and b will move into the domain that is initially filled with b and thus form a moving reaction front:

$$u(\underline{x}, t) = U(\underline{x} \cdot \underline{n} - st)$$

with speed of propagation s and direction of propagation $\underline{n}$ normal to the front. Reactions only occur at the front, where a and b meet, and the front thickness increases over time due to diffusion. Without diffusion, no reactions would ever happen, since a and b don't mix. In order to better understand this dynamics we perform a dimensional analysis.

Dimensional analysis

Independent dimensions: 3 (L, M, T)
Variables: 7 (D, s, M, k, r, V, u)
⇒we need $7 - 3 = 4$ dimensionless groupings.

		L	M	T ⇓	
	D	2	0	−1	diffusion constant
⇒	s	1	0	−1	front propagation speed
	M	0	−1	0	"molecular weight"
	k	3	−1	−1	reaction rate constant
	r	1	0	0	cube edge length
	V	3	0	0	particle volume
	u	−3	1	0	concentration of a

The dimensions of the reaction rate constant can be seen from the expression for the reaction term: $f = ku(1-u)$. Both u and $(1-u)$ have the dimensions of concentration and the overall expression has the dimensions of concentration/time (corresponding to the left-hand side of the governing equation), thus:

$$k(\text{conc})^2 = \frac{\text{conc}}{T} \qquad \Rightarrow \qquad k = \frac{1}{\text{conc} \cdot T} = \frac{L^3}{MT} \,.$$

We follow Taylor's method to find the dimensionless groupings:

		L	M
			$\Downarrow$
	D/s	1	0
$\Rightarrow$	M	0	-1
	k/s	2	-1
	r	1	0
	V	3	0
	u	-3	1

	L
	$\Downarrow$
D/s	1
$k/(sM)$	2
$\Rightarrow$ r	1
V	3
uM	-3

	L
$D/(sr)$	0
$k/(sMr^2)$	0
V/r^3	0
uMr^3	0

The four dimensionless groupings completely describing the dynamics of the problem thus are:

$\Pi_1 = \frac{D}{sr}$	dimensionless diffusion constant	
$\Pi_2 = \frac{k}{sMr^2}$	"	reaction rate
$\Pi_3 = \frac{V}{r^3}$	"	particle volume
$\Pi_4 = uMr^3$	"	concentration

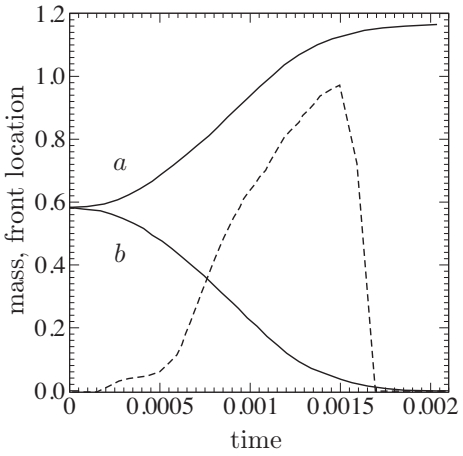
The simulation results are shown in Fig. 7.1. They were done using PSE and SSA with $M = 10$ and two different diffusion constants of $D = 0.1$ and $D = 1.0$. Figure 7.1(a) shows the time evolution of the total mass of a and b in the cube and, as a dashed line, the location of the reaction front, defined as the plane where $[a] = [b] = 0.5$. Figure 7.1(b) shows the dimensionless front propagation speed (slope of the dashed line in (a) before saturation) as a function of the dimensionless specific probability rate. As expected, the two curves for the two different diffusion constants collapse into one when using dimensionless groupings. Using this dimensionless representation, we could now design experiments to measure the front propagation speed and identify a law of how it depends on the reaction rate. The dependence on the diffusion constant is clear from the dimensional analysis.

Figure 7.1(a) also illustrates the three regimes that a reaction-diffusion system can be in. For small times, the slope of the dashed curve (front position) is smaller than for later times. This early stage is called the "diffusion-limited regime". Here, diffusion is slower than the reactions since it has not yet fully developed. The reactants are thus not delivered fast enough for the reaction to work at its full kinetic speed. Once diffusion has sufficiently mixed the two sides of the cube (i.e., created a sufficiently diffuse interface), the reaction can go at full speed and the front moves faster. This is the "reaction-limited" regime where diffusion is no longer the rate-limiting process. Finally, b gets depleated so much that the reaction again slows down as there is not enough "fuel" for it any more. This is the "resource-limited regime" where the front speed rapidly breaks down.

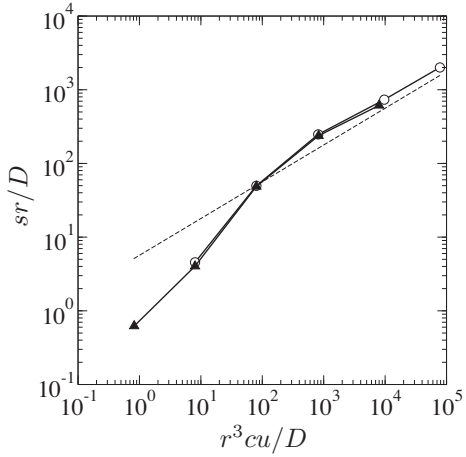
Despite the simplicity of this example problem, it has attracted a great deal of attention in both theory and experiment. The questions of how the front propagation speed depends on the diffusion constant and the reaction rate, and under which conditions such traveling fronts exist has been addressed by numerous scientists. The theoretical prediction for the front speed in the present example problem is:

$$s^* = 2\sqrt{f'(0)}\,. \tag{7.1}$$

The dashed line in Fig. 7.1(b) shows this theoretical prediction. Compared to our simulations we see that the prediction is good only for larger reaction rates. This is consistent with the theory assuming a reaction-dominated system, where reactions are fast compared to diffusion. Using numerical simulations, however, we can also explore the other regimes.



(a)



(b)

Figure 7.1: (a) Evolution of the total mass of a and b (solid lines) for $M = 10$ molecules per unit mass. The location of the reaction front is shown by the dashed curve. The reaction front moves into the region of b until all of b is consumed; then, it collapses. (b) Dependence of the front speed s on the specific probability rate c. In dimensionless numbers, the two curves for $D = 0.1$ (open circles) and $D = 1.0$ (filled triangles) collapse into one. The theoretical scaling according to Eq. (7.1) is indicated by the dashed line.

7.3.2 Examples of Turing patterns

Besides moving fronts, reaction-diffusion systems can also exhibit inhomogeneous stationary concentration distributions. These patterns were first described by A. M. Turing in his seminal work "The chemical basis of morphogenesis", which was published in 1952. Since then, the body of literature on Turing patterns has become vast and we will only give a few examples here.

Turing patterns can exist under the following conditions:

- the diffusion constants of the different species are very different from each other (usually several orders of magnitude)
- there is a local self-enhancement (auto-catalysis) in the reaction system
- there is a long-range inhibition (supressor) in the reaction system.

There are many well-known examples of pattern-forming systems. In his classical 1952 paper, Turing proposed the following reaction-diffusion system, known as Turing's spot-forming system:

$$\begin{aligned} \frac{\partial a}{\partial t} &= s(16 - ab) + D_a \Delta a \\ \frac{\partial b}{\partial t} &= s(ab - b - \beta) + D_b \Delta b \,. \end{aligned}$$

It consists of two species, a and b, that diffuse isotropically and homogeneously with diffusion constants D_a and D_b, and react. Through the reaction, a enhances b and b inhibits a. In addition, there is a local self-inhibition of b, a is produced at a constant rate of $16s$, and b is consumed at a constant rate of βs. This reaction-diffusion system creates a stationary spot pattern at steady state.

Another spot-forming system was later described by Hans Meinhardt in 1982. It is given by the equations

$$\begin{aligned} \frac{\partial a}{\partial t} &= s\left(a p_1 + \frac{0.01 a_i a^2}{b} + p_3\right) + D_a \Delta a \\ \frac{\partial b}{\partial t} &= s\left(b p_2 + 0.01 a_i a^2\right) + D_b \Delta b \end{aligned}$$

for the two species a and b. The pattern created by this system when starting from a uniformly random distribution of a and b on the unit sphere is shown in Fig. 7.2.

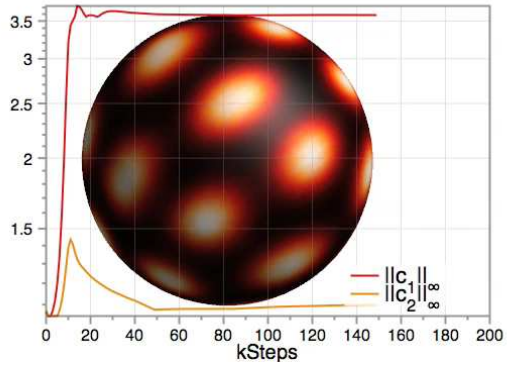


Figure 7.2: Steady-state pattern created by the Meinhardt spot-forming system on the unit sphere. The simulation started from a uniformly random distribution of a and b and was done by Michael Bergdorf (CSE Lab, ETHZ) using particle methods. The lines in the plot show the time evolution of the maximum concentration of a (c_1) and b (c_2) during the formation of the pattern.

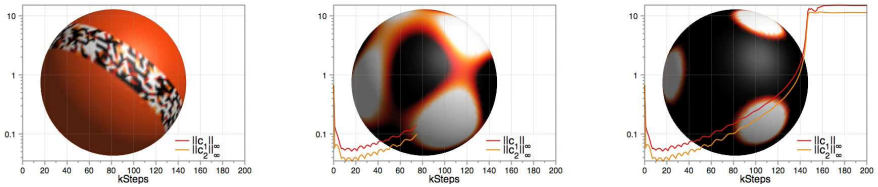


Figure 7.3: Bifurcation instability in the Meinhardt spot-forming system when starting from the initial condition shown on the left. The system first generates large, connected spots, but becomes unstable after some time due to numerical noise. After some oscillations, it then falls into a second pattern of more compact, smaller spots. The second steady state is stable and the system will stay there. The transient oscillations are clearly visible in the time evolution of the concentrations shown by the lines in the plot.

The patterns that are generated by a reaction-diffusion system depend on the initial condition one starts from. This is illustrated in Fig. 7.3, where the initial random perturbations are limited to a narrow band around the equator of the sphere. The rest of the sphere is homogeneously filled. The pattern that is generated differs from the one shown in Fig. 7.2, even though it still consists of spots. More interestingly, also the transient behavior changes. While the system in Fig. 7.2 directly converges

to the final pattern, the equator-local initial perturbations lead to a bifurcation instability. The system first forms the spot pattern shown in the middle panel of Fig. 7.3. After some time, this pattern becomes unstable and begins to oscillate. This causes the system to converge to a different steady state as shown in the right panel. This steady state is stable and the pattern will remain for all times.
In the same 1982 paper, Meinhardt also described the classic stripe-forming system

$$\begin{aligned}
\frac{\partial g_1}{\partial t} &= \frac{c s_2 g_1^2}{r} - \alpha g_1 + D g \nabla^2 g_1 + \rho_0 \\
\frac{\partial g_2}{\partial t} &= \frac{c s_1 g_2^2}{r} - \alpha g_2 + D g \nabla^2 g_2 + \rho_0 \\
\frac{\partial r}{\partial t} &= c s_2 g_1^2 + c s_1 g_2^2 - \beta r \\
\frac{\partial s_1}{\partial t} &= \gamma (g_1 - s_1) + D s \nabla^2 s_1 + \rho_1 \\
\frac{\partial s_2}{\partial t} &= \gamma (g_2 - s_2) + D s \nabla^2 s_2 + \rho_1 \, .
\end{aligned}$$

Figure 7.4 shows the steady-state pattern created by this system when starting from a uniformly random distribution of a and b on the unit sphere.

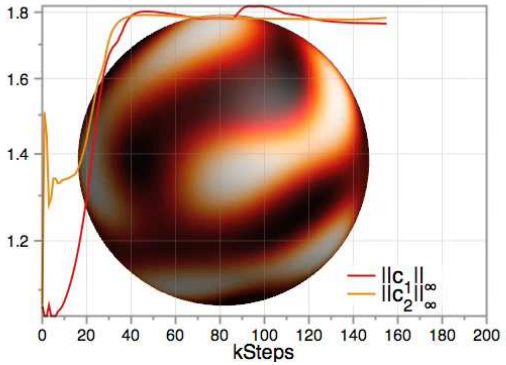


Figure 7.4: Steady-state pattern created by the Meinhardt stripe-forming system on the unit sphere. The simulation started from a uniformly random distribution of a and b and was done by Michael Bergdorf (CSE Lab, ETHZ) using particle methods. The lines in the plot show the time evolution of the maximum concentration of a (c_1) and b (c_2) during the formation of the pattern.

Pattern-forming reaction-diffusion systems are frequently used to model morphogenesis. This can be done by letting the surface deform according to the concentration of one of the species. This species models a growth factor or growth hormon that leads to local proliferation in the modeled tissue. Figure 7.5 shows an example using the classical Turing spot-forming system on the unit sphere where the sphere deforms according to the local concentration of a. Where a is more abundant (highlighted by the red color in Fig. 7.5), the surface moves radially outward. The deformation velocity of the surface is thus given by $Ca\underline{n}$, where C is a constant and $\underline{n}$ the outer unit normal onto the surface. The deformation of the surface influences the behavior of the reaction-diffusion system on it by changing the local surface area and curvature. This coupled interplay between growth and reaction-diffusion ultimately leads to the creation of the shape shown in the right panel of Fig. 7.5.

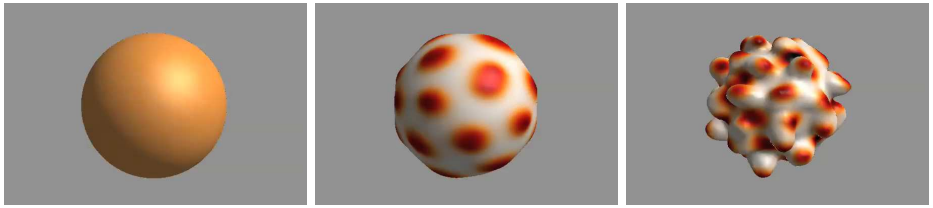


Figure 7.5: Turing's spot-forming system on the unit sphere with surface deformation proportional to the local concentration. The simulation was done by Michael Bergdorf (CSE Lab, ETHZ) using particle methods and a level-set surface representation. The left panel shows the initial condition, the middle panel an intermediate time point, and the right panel a later time point.

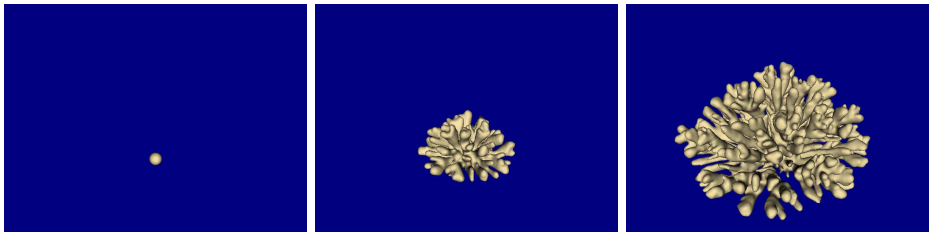


Figure 7.6: Simulation of the morphogenesis of a coral by Kaandorp et al., Proc. R. Soc. B, 2005. The left panel shows the initial shape, a sphere. As the reaction-diffusion system evolves and drives the deformation of the shape, it gradually develops into a coral-like structure. The middle panel shows the simulation result at an intermediate time point, the right panel at a later time point.

One can now look for reaction-diffusion systems that, when simulated on a sphere or another geometric shape, lead to the growth of shapes as they are found in nature. One example is shown in Fig. 7.6. Kaandorp et al. found a reaction-diffusion

system which leads to the growth of coral-like shapes. The system phenomenologically reproduces typical coral shapes. However, it is a different story to biologically interpret the meaning of the different chemical species and reactions, and to identify them with specific growth factors and biochemical pathways. Simulation, however, provide guidance about where one should look for these reactions and what kind of mechanisms and networks *could* be responsible for the control of morphogenesis. This is still an active area of research in many fields of biology.
But Turing patterns are not only used in biology. Another important area of application is computer graphics. There, Turing patterns are used to create textures on 3D objects in virtual worlds. Figure 7.7 shows two examples: the fur coat of a giraffe and a zebra. Both are steady-state solution of specifically engineered reaction-diffusion systems. The applications of this technique range from computer-animated movies to video and computer games.

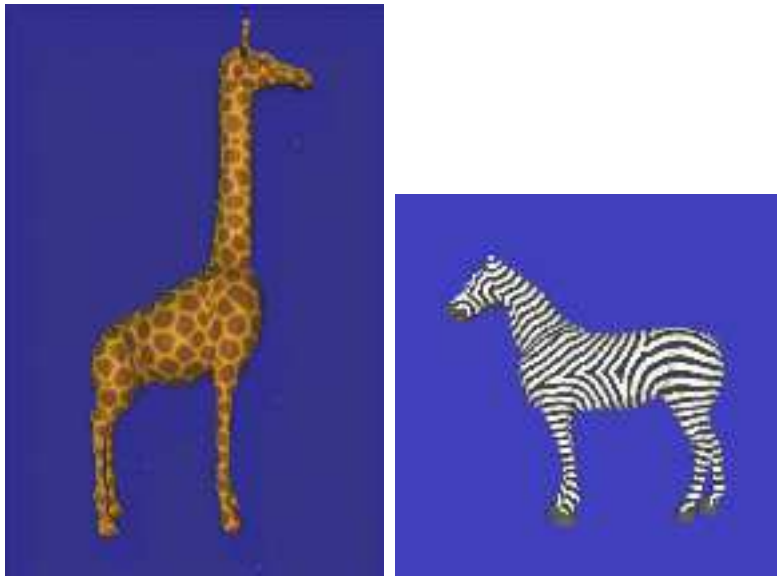


Figure 7.7: Application of Turing patterns in computer graphics: the fur coats of a giraffe and a zebra as computed from the steady-state solution of pattern-froming reaction-diffusion system. These two examples are taken from the work of Greg Turk (Georgia Tech)

.

Chapter 8

Advection-Diffusion

In this chapter:

- The governing equation for advection-diffusion
- Advection-diffusion simulations using particles
- Stability of the discretization
- Incompressibility conditions
- Remeshing and a hierarchy of moment-conserving interpolation schemes

Learning goals:

- Be able to simulate advection-diffusion systems using PSE and Random Walk
- Be able to define compressible and incompressible advection mathematically
- Know the advection stability condition and the Lagrangian CFL condition
- Know what incompressibility means and how it simplifies the governing equation
- Know why and when remeshing is needed and be able to implement it
- Be able to explain the particle cloud and assignment function interpretations of interpolation
- Know the first three members of the hierarchy of moment-conserving interpolation schemes

Moving to larger length scales, flows become an important transport phenomenon in biology. Examples include the flow of blood in the blood vessels or the flow of air in the lungs. But also external flows such as the flow of water around swimming fish or corrals pose interesting modeling problems. In flows, we distinguish between *advection* and *convection*. In advection, the velocity field of the flow, i.e., the velocity of the fluid elements at each position in space and time, is explicitly given or known. In convection, this velocity field is itself an emergent property of the flow. We first consider the simpler case of advection and introduce the important concept of *incompressibility*. In the next chapter, we generalize to convection.

8.1 Governing Equation

In a combined advection-diffusion problem, the transport of a quantity by advection with a given velocity field $\underline{v}$ is overlaid with diffusion. The governing equation can be derived using the method of control volumes and conservation of mass. It is:

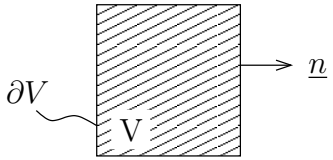
$$\frac{\partial u}{\partial t} + \overbrace{\nabla \cdot (\underline{v} u)}^{\text{advection}}_{=\underline{v}\cdot(\nabla u)+u(\nabla\cdot\underline{v})} = \overbrace{\nabla \cdot (\underline{\underline{D}} \nabla u)}^{\text{diffusion}} .$$

Depending on the form of the first term, we distinguish two cases:

- *incompressible advection* with a divergence-free velocity field, i.e., $\nabla \cdot \underline{v} = 0$. In this case, the governing equation simplifies to
$$\frac{\partial u}{\partial t} + \underline{v} \cdot (\nabla u) = \nabla \cdot (\underline{\underline{D}} \nabla u)$$
- *compressible advection*, where $\nabla \cdot \underline{v} \neq 0$ and the governing equation remains in its full form.

Derivation

In order to derive this governing equation, consider a finite Eulerian control volume V with boundary ∂V and outer unit normal $\underline{n}$.



This control volume forms a reservoir for the extensive quantity ("mass"): $\int_V u \, dV$. Formulating the balance equation, we relate the temporal change of mass inside the control volume to the fluxes across its boundary. Since the normal $\underline{n}$ onto the boundary is positive in the outward direction, influxes have a negative sign. There are two fluxes: the one due to diffusion across the boundary and the one due to advection, thus:

$$\frac{\partial}{\partial t} \int_V u \, dV = -(\text{diffusive flux}) - (\text{advective flux}) .$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\quad (1) \qquad\qquad (2)$$

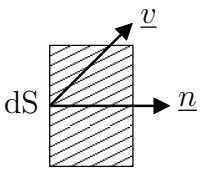
For the diffusive flux density (flux per unit area), we use Fick's law $\underline{j} = -\underline{\underline{D}}\nabla u$, such that the total diffusive flux (in direction of the normal) across the whole boundary of the control volume is given by

$$-\oint_{\partial V} (\underline{\underline{D}}\nabla u) \cdot \underline{n}\, dS\,.$$

The scalar product with the normal ensures that we only account for the normal component of the flux that is actually crossing the boundary of the control volume (see Section 3.3 for the mathematical form).

In order to derive an algebraic expression for the advective flux (2), we consider an infinitesimal surface element dS on the boundary of the control volume. Because the surface element is infinitesimal, it is flat and the advection velocity $\underline{v}$ is constant and homogeneous across it. The total flux (in direction of the normal) through this boundary element is given by the normal component of the flow velocity multiplied with the size of the surface element. This can be interpreted as the volume that is pushed across the boundary by the flow per unit time (shaded area in the figure below). The fluid volume is traveling with a normal velocity of $\underline{v} \cdot \underline{n}$ through the area dS. The total volume transported per time thus is:

$$\underline{v} \cdot \underline{n}\, dS\,.$$



The quantity u that is transported by the flow is also homogeneously distributed along the infinitesimal surface element. The total mass moving across the boundary is given by the transported volume (above) times the concentration u, thus:

$$u\underline{v} \cdot \underline{n}\, dS\,.$$

Again, we integrate over the whole boundary of the control volume and obtain the total advective flux in direction of the normal:

$$\oint_{\partial V} u\underline{v} \cdot \underline{n}\, dS\,.$$

Substituting the two algebraic expressions for the diffusive and advective fluxes, the balance equation becomes

$$\int_V \frac{\partial u}{\partial t}\, dV = \oint_{\partial V} (\underline{\underline{D}}\nabla u) \cdot \underline{n}\, dS - \oint_{\partial V} u\underline{v} \cdot \underline{n}\, dS\,.$$

This equation contains both a volume integral on the left-hand side and surface integrals on the righ-hand side. We therefore apply Gauss' theorem to convert all integrals to volume integrals and take all terms together under the same integral:

$$\int_V \frac{\partial u}{\partial t} dV = \int_V \nabla \cdot (\underline{\underline{D}}\nabla u) dV - \int_V \nabla \cdot (\underline{v}u) dV$$
$$\int_V \left[\frac{\partial u}{\partial t} - \nabla \cdot (\underline{\underline{D}}\nabla u) + \nabla \cdot (\underline{v}u) \right] dV = 0\,.$$

This has to hold for all possible control volumes V and the only way this can be satisfied is for the integrand to be zero. This recovers the governing equation for advection-diffusion processes as stated above.

8.2 Simulation

We discretize the advection-diffusion equation using particles that correspond to Lagrangian control volumes, carrying mass as their extensive strength. As the particles move with the flow (i.e., with the advective flow velocity), the only change of mass they experience is the one due to diffusion. For divergence-free velocity fields, the Lagrangian form of the governing equation thus becomes:

$$\frac{Du}{Dt} = \nabla \cdot (\underline{\underline{D}}\nabla u)$$
$$\frac{dx}{dt} = \underline{v}\,.$$

(Compare the material derivative for u to the left-hand side of the governing equation!) In general, we also have to account for the term $u(\nabla \cdot \underline{v})$. This is the divergence of the velocity field of the flow or, in other terms, the source strength of the velocity field. If the velocity field has a local source, this means that all velocity vectors are oriented to point radially away from that location. This means that any particle containing that point will be "inflated" (recall that particles are Lagrangian control volumes and as such always contain the same set of fluid elements), hence the name *compressible advection*. In the compressible case, we thus have to add an evolution equation for the particle volumes that is proportional to the local source strength (divergence) of the velocity field, thus:

$$\frac{dV}{dt} = (\nabla \cdot \underline{v})V\,.$$

These are the equations describing the evolution of the particle attributes over time, namely the:

locations $\underline{x}_p(t)$
strengths $\omega_p(t)$
volumes $V_p(t)$.

The time evolution of the particle positions $\underline{x}_p$ is only governed by the advection velocity. The strengths ω_p evolve according to diffusion only, and, in the compressible case, the volumes V_p only feel the effect to the dilation (inflation). In the Lagrangian description, the three phenomena are thus nicely decoupled and can be treated separately. This was not the case in the Eulerian formulation, but provides an important advantage: our simulations become modular! We can independently implement (and test) solvers for diffusion (e.g., using PSE), particle motion, and volume dilation and can later connect them together in any way to perform a simulation. The modules for diffusion, advection, flow, or chemical reactions can thus be freely combined, enabling modular and flexible software architectures.

Converting from concentration u to strength ω and using a discretized form ∇^h of the Nabla operator (e.g., using PSE), the final simulation scheme becomes:

$$\begin{cases} \dfrac{d\omega_p}{dt} = V_p \nabla^h \cdot (\underline{\underline{D}} \nabla^h u(\underline{x}_p, t)) \\ \dfrac{d\underline{x}_p}{dt} = \underline{v}(\underline{x}_p, t) = \underline{v}_p \\ \dfrac{dV_p}{dt} = V_p \nabla^h \cdot \underline{v}(\underline{x}_p, t) \qquad (**). \end{cases}$$

For incompressible advection, the particle volumes remain constant and the third equation disappears.

In this scheme, diffusion can be done using either PSE or RW and advection simply amounts to moving the particles with the local flow velocity. When using PSE for the diffusion part, the righ-hand side of the evolution equation for ω above is evaluated over the neighboring particles using isotropic or anisotropic PSE kernels (depending on the shape of the diffusion tensor $\underline{\underline{D}}$). If one uses RW, the particle strengths do not change and we add the RW contribution to the advective particle velocity $\underline{v}$ before moving the particles.

8.2.1 Stability

It is instructive to see under which conditions the above simulation scheme is numerically stable. This highlights one of the advantages of particle methods in advective problems. We first consider the simpler case of incompressible advection and then generalize.

Incompressible advection: In incompressible advection, the evaluation of the right-hand side of equations (**) (i.e., the spatial discretization) is always stable. There is no linear stability condition from the space discretization. For homogeneous velocity fields, particle methods even yield exact solutions. This is an important difference to mesh-based numerical methods, where the linear stability condition imposes that the flow must never travel farther than a certain number (called the CFL limit number) of mesh cells per time step. In particle methods, there is no such condition on the time step for linear stability. The time integration scheme, however, will still impose stability limits on the time step!

Compressible advection: In compressible advection, stability requires that any two particle trajectories do not cross within a time step. This imposes the time step limit $\delta t < C||\nabla \otimes \underline{v}||_\infty^{-1}$ (Bergdorf & Koumontsakos, SIAM J. MMS 5, 2006), defining a Lagrangian CFL number

$$\text{LCFL} := \delta t ||\nabla \otimes \underline{v}||_\infty$$

such that LCFL $\overset{!}{<} C$. This is the time step limit coming from the spatial discretization using Lagrangian particles and is in addition to the time step limit of the time integration scheme. The infinity norm of the outer product of ∇ and $\underline{v}$ is the largest (by absolute value) component of the velocity gradient in any spatial direction.

8.3 Incompressibility

Let's look at the physical meaning of incompressibility and how this can be used to understand the above stability results. The condition that incompressible advection has a divergence-free velocity field makes intuitive sense: recall that divergence describes a source density. If there were sources in the velocity field, then the Lagrangian control volumes (i.e., the particles) would inflate as described above. Since a Lagrangian control volume always contains the same set of fluid elements, their density would decrease because the control volume inflates, but the total mass inside it remains constant. This is in contradiction to the requirement of incompressibility. Something that is incompressible has to have a constant density. Sources (or sinks) in the velocity field can therefore not exist in an incompressible fluid. The two conditions of incompressibility, $\rho = \not{\epsilon}$ and $\nabla \cdot \underline{v} = 0$, are thus equivalent.

This also provides an intuitive explanation for the absence of a time step limit. If we model a fluid as incompressible, we neglect sound waves (density fluctuations) and we are only interested in the bulk transport properties of the flow. This is a good approximation if the maximum advection velocity is small compared to the speed of sound, thus $|\underline{v}|_{\max} \ll c_{\text{sound}}$. This is a direct consequence of selecting the relevant time scales for the model. When simulating flows with relevant velocities that are much smaller than the speed of sound, we treat sound waves as fast dynamics and do not include them explicitly (sound is assumed to instantly propagate throughout the entire domain). While this is not a definition of incompressibility, it can be used in practice to decide whether or not to model a given flow as incompressible and hence be allowed to neglect particle volume changes. This can sometimes be counter-intuitive. While air is mostly percieved as compressible in

our everyday experience (imagine, for example, a manual bicycle tire pump), we would still model it as incompressible when simulating air flow in the lungs. This is because the maximum flow velocity during breathing is much smaller than the speed of sound in air. Since we are not trying to resolve sound waves, we don't need a time step limit that guarantees that we resolve their dynamics. Incompressible advection can thus not be used to model fast (compared to the speed of sound) flows. To simulate acoustics or shock waves, we need compressible advection with variable density (particle volumes).

8.4 Remeshing

The motion of the particles with the advection velocity field can lead to irregular particle distributions. In particular, there can be regions where the particles sparsify and eventually violate the overlap condition introduced in Section 5.1. In order to maintain the integrity of the smooth function approximation, the particle distribution is therefore periodically regularized by remeshing the particles. As already introduced in Section 5.3, remeshing consists of the following steps:

- interpolation of the particle strengths onto a regular Cartesian grid of resolution $h \leq \epsilon$;
- creation of a new set of particles at grid nodes where the field value is non-zero (in practice: larger in magnitude than machine epsilon);
- resetting the particle volumes to h^d in $\mathbb{R}^d$ in the case of compressible advection (this is new compared to Section 5.2.2).

Interpolation is done using moment-conserving schemes as introduced in Section 5.2.2. We take this opportunity to revisit the topic of interpolation schemes and provide some more details below.
Remeshing also provides a consistent way of inserting or removing particles from the simulation where needed (adaptivity, another important advantage of particle methods). In areas where particles tend to cluster the simulation becomes wasteful as the function could also be represented using much less particles there. Remeshing will reduce the number of particles there to match the given resolution h. In areas where new particles are needed because the field starts to develop, they are created by remeshing.
In simulations of compressible advection, remeshing in addition resets the volumes. This also prevents particles from becoming arbitrarily large or small.

Interpolation

Good interpolation kernels have to fulfill the following properties:

1. at particle separations $\gg h$, fluctuations should be negligible ⇒errors should be localized.
2. if we interpolate a field from particles to a mesh, the result should vary smoothly as particles move across mesh cells ⇒fluctuations should be small in close vicinity.
3. moments of the interpolated function in particle-to-mesh interpolation should be conserved up to a certain order.

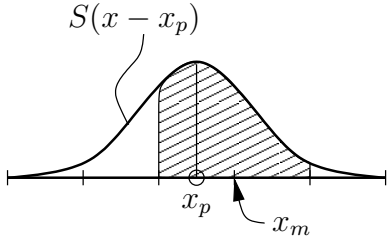
These quality requirements can directly be used to construct good interpolation kernels in a systematic way:

1. restricts the mathematical form of the interpolation kernels to functions with local support,
2. enforces smooth interpolation kernels,
3. allows determining the coefficients of the interpolation kernel.

In Section 5.2.2, we have introduced particle-to-mesh interpolation as a weighted distribution of particle strength onto the surrounding mesh nodes, where the weight is determined by the interpolation kernel as a function of the distance between the particle and the mesh node. We will now adopt a more visual view. This can be done in two ways:

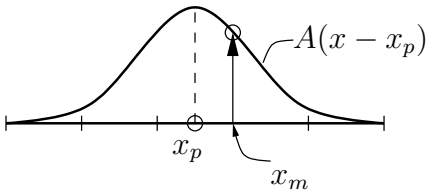
- **Particle clouds:** In the particle cloud interpretation of interpolation, we imagine that the particles carry clouds S of finite size. These are not to be confused with the mollification kernels used in smooth particle function approximations. The clouds here are just a way of visually interpreting interpolation schemes. The fraction of strength assigned from particle p at x_p to mesh node m at x_m is given by the overlap of the cloud with the mesh cell, thus:

$$W(x_p - x_m) = W_m(x_p) = \int_{x_m - \frac{1}{2}h}^{x_m + \frac{1}{2}h} S(x' - x_p) dx' \,.$$



- **Assignment functions:** Again, we imagine that the particles carry functions A, but this time the fraction of strength assigned from particle p at x_p to mesh node m at x_m is given by the function *value* at x_m, thus:

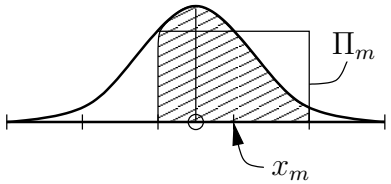
$$W(x_p - x_m) = A(x_m - x_p)\,.$$



The two interpretations are related through:

$$W = A(x_m - x_p) = \int_\Omega \Pi_m \left(\frac{x'}{h}\right) S(x' - x_p)dx'\,,$$

where Π_m is the top-hat indicator function on mesh cell m, i.e., the function that is 0 everywhere outside of the mesh cell centered at node m and 1 inside. For even cloud functions S, this amounts to a convolution: $A = \Pi(\frac{x}{h}) * S(x)$. The assignment function and the cloud function of any interpolation scheme are thus related through a convolution with the top-hat function over the mesh cell. This can be intuitively illustrated:



8.4.1 A hierarchy of interpolation schemes

Since every assignment function is also a valid cloud function when normalized by $\frac{1}{h}$ (not easy to see), the above relationship between cloud functions and assignment functions defines a recursion that can be used to systematically construct a hierarchy of interpolation schemes that fulfill the quality criteria listed above. The hierarchy starts from the simplest local, moment-conserving interpolation scheme: assign all strength of the particle to the nearest mesh point. This scheme is called NGP (nearest grid point) interpolation and obviously conserves the moment of order 0, i.e., the total mass. However, no higher-order moments are conserved, and the interpolated values change discontinuously when particles move from one mesh cell into a neighboring one. The cloud function of the NGP scheme is $S(x) = \delta(x)$ and the assignment function can be determined by convolution with the top hat, thus:

$$A(x) = \Pi\left(\frac{x}{h}\right) * \delta(x) = \Pi\left(\frac{x}{h}\right)\,.$$

The cloud and assignment functions of the NGP interpolation scheme are shown in Fig. 8.1. This scheme is of order 0 (in the sense of the order of the highest conserved moment of the extensive quantity, not the order of convergence of the intensive quantity) and yields a piecewise constant interpolated field.

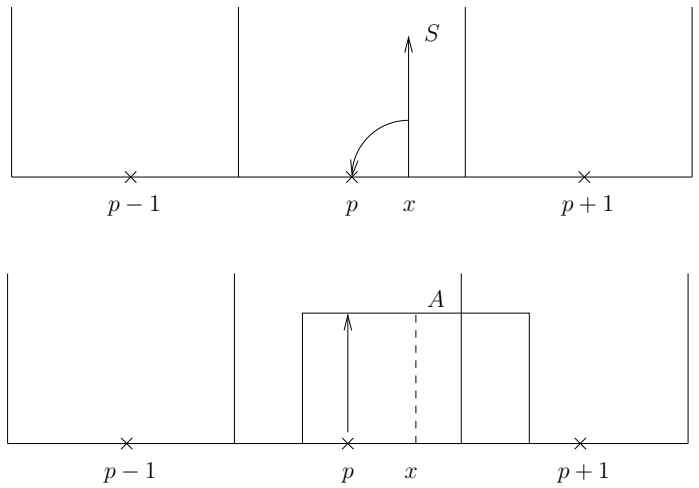


Figure 8.1: Particle cloud (top) and assignment function (bottom) of the nearest grid point (NGP) interpolation scheme.

It is easy to see that the normalized top hat function $\frac{1}{h}\Pi\left(\frac{x}{h}\right)$ can also serve as a cloud function. The normalization is makes the cloud conserve mass. This will still conserve the 0 order moment, but in addition also the moment of order 1. The resulting interpolation scheme is called CIC (cloud in cell) interpolation. Its cloud function now has a support of 2 mesh cells in each direction and the assignment function (again computed by convolution with the top hat) is triangular (see Fig. 8.2). The CIC scheme is of order 1 and yields a piecewise linear interpolated field.

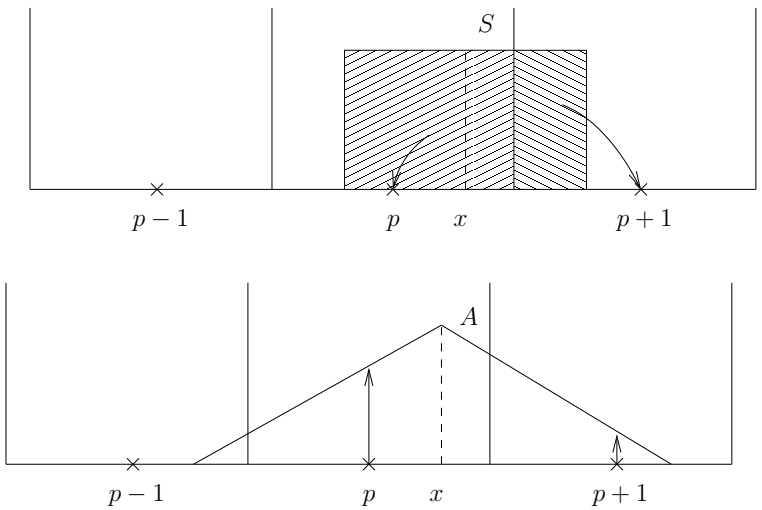


Figure 8.2: Particle cloud (top) and assignment function (bottom) of the cloud in cell (CIC) interpolation scheme.

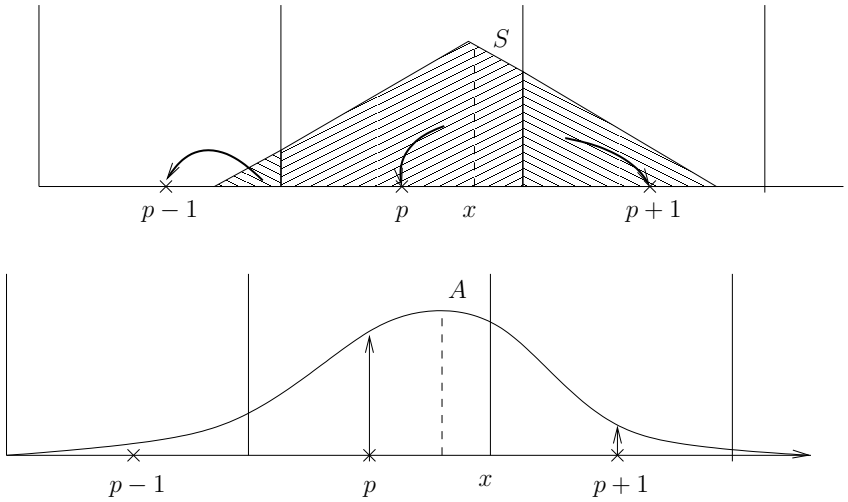


Figure 8.3: Particle cloud (top) and assignment function (bottom) of the triangular shaped cloud (TSC) interpolation scheme.

This triangular function can then be used as the cloud function of the next higher scheme, the 2^{nd}-order accurate TSC (triangular shaped cloud) scheme, and convolution with $\Pi\left(\frac{x}{h}\right)$ yields the new assignment function as shown in Fig. 8.3. TSC yields an interpolated field that is smooth in the value and continuous in the first derivative.
The next higher-order PQS scheme is smooth up to the second derivative and is of order 3. This hierarchy can be iterated as far as needed. The first 4 schemes are summarized in Table 8.1.

Scheme	Order	Support	Cloud	Assignment function	Field
NGP	0	1^d	δ	Π	Stepwise
CIC	1	2^d	Π	$\bigwedge = \Pi * \Pi$	Continuous piecewise linear
TSC	2	3^d	$\bigwedge$	$\Pi * \Pi * \Pi$	Continuous value and first derivative
PQS	3	4^d	$\bigwedge * \Pi$	$\Pi * \Pi * \Pi * \Pi$	Continuous value, first, and second derivative

Table 8.1: Summary of the first 4 members of the hierarchy of interpolation schemes and the order of the highest conserved moment in particle-to-mesh interpolation.

Since the result of an NGP interpolation is piece-wise constant, its first derivative is not defined (the gradient is infinite at mesh cell boundaries). Higher-order schemes yield successively smoother results with the CIC scheme having continuous values, the TSC scheme continuous values and first derivatives, and the PQS scheme continuous values as well as first and second derivatives.
It is important to always use an interpolation scheme that yields results that are smooth up to the degree of the highest derivative that occurs in the model. Otherwise, the numerical approximation of the derivative can diverge. If we want to simulate, e.g., diffusion, we have to evaluate a discretized second derivative. We thus have to use an interpolation scheme that provides results that are smooth up to and including the second derivative (such as PQS). The smoothness of an interpolation scheme can be increased using Richardson extrapolation without increasing the order of the highest conserved moment (Monaghan, 1985). This has the advantage of yielding sufficient smoothness in the interpolated result without requiring a larger support (computational cost). The M_4' scheme introduced in Section 5.2.2 provides continuous values as well as first and second derivatives (just like PQS). But it only conserves the moments up to and including the second in particle-to-mesh interpolation (like TSC). In mesh-to-particle interpolation, however, its order of convergence is 3.
Conservation of moments in particle-to-mesh interpolation can be best understood in the particle cloud interpretation, which reflects the re-distribution of each par-

ticle's extensive strength onto the surrounding mesh nodes. Mesh-to-particle interpolation of the intensive values stored on the mesh is best understood in the assignment function interpretation. Here, no extensive strength is re-distributed, but the interpolation weights are determined by evaluating the assignment function of the interpolation scheme at a certain distance. This does, in general, not conserve the moments of the interpolated extensive quantity (but recall that we have to use the same interpolation kernel to go both ways). The error introduced by mesh-to-particle interpolation decreases with a certain power of h. This power (order of convergence) is usually chosen at least one larger than the order of convergence of the operator approximation. The interpolation error is thus not visible in the final result, as it is dominated (masked) by the operator error. The interpolation schemes presented here only conserve moments if the target (mesh or particles) of the interpolation is regularly spaced with some spacing h. Notice also that remeshing only requires particle-to-mesh interpolation and hence always conserves the moments.

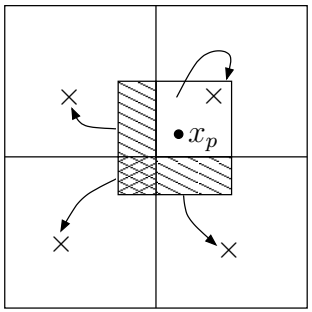
Attention:
While these interpolation kernels conserve successive moments of the interpolated field function, they do *not* conserve energy! A different set of kernels is available for that, but not discussed here. Also, these interpolation schemes only conserve moments when interpolating to *regular Cartesian* meshes. On irregular or non-uniform meshes, the moments are not conserved. In order to conserve moments also there, one would have to use different interpolation kernels for each mesh point (i.e., locally adapt the kernels to the resolution and geometry of the mesh).

In multiple dimensions

So far, we have only talked about interpolation in 1D. In $\mathbb{R}^d$, the kernels of this hierarchy are Cartesian products of the 1D kernels. We can thus compute the interpolation weights in each direction separately and then multiply them together, thus:

$$\begin{aligned} W(\underline{x}) &= W_1(x_1) \cdot W_2(x_2) \cdot \ldots \cdot W_d(x_d) \\ &= \Pi_{i=1}^{d} W_i(x_i)\,. \end{aligned}$$

Example 8.4.1 (CIC in 2D: areas)**.**



Chapter 9

Incompressible Flow

In this chapter:

- The governing equations for flow
- Flow simulations using particles (the vortex method)
- The algorithm for vortex methods
- Properties and characteristics of vortex methods

Learning goals:

- Have seen the governing equations for fluid flow
- Know by heart how circulation induces velocity
- Be able to implement a hybrid particle-mesh flow simulation
- Know the advantages and disadvantages of vortex methods compared to grid-based methods

In convective flow, the velocity field is implicitly defined by the physics of the flow and the forces applied to the fluid, and it is not an imposed given. Convective flow thus describes the natural motion of fluids such as liquids and gases driven by some external force and hence plays an important role on macroscopic length scales. After reviewing the governing equations of convective flow, we will extend our particle simulation framework to include also this case for incompressible flows. The main difference to advection is that we now have to compute the velocity field from the particle attributes at every time step. We will see how this can be efficiently done.

9.1 Governing Equations

In convective flow, the flow field is described by the velocity $\underline{v}(\underline{x},t)$, the pressure $p(\underline{x},t)$, and the density $\rho(\underline{x},t)$ (we called the density "concentration" before, but I will now switch to the terminology commonly used in fluid mechanics). All of these quantities are field functions of time and space. The governing equations are derived in an infinitesimal control volume from conservation of mass and impulse. Conservation of mass leads to the continuity equation (for the derivation see the self-test questions of lecture 5), conservation of impulse to the so-called Navier-Stokes equation. Together, these two equations govern the spatiotemporal dynamics of flow.

- Conservation of mass $\int \rho \, dV$: Continuity equation

$$\underbrace{\frac{\partial \rho}{\partial t}}_{\substack{\text{accumulation}\\ \text{of mass}}} + \underbrace{\nabla \cdot (\rho \underline{v})}_{=\underline{v}\cdot\nabla\rho+\rho\nabla\cdot\underline{v}} = 0$$

  advection of mass across boundary with advection velocity $\underline{v}$

  Diffusion is neglected, as one assumes that in a flow, advection is the dominant transport mechanism on the macroscopic scale. As there is no diffusion, the continuity equation then simply formulates the mass balance in the control volume with the only flow of mass due to advection across the boundary. It thus amounts to a simple advection equation (see Chapter 8).

- Conservation of impulse $\int \rho \underline{v} \, dV$: Navier-Stokes equation

$$\underbrace{\rho \frac{\partial \underline{v}}{\partial t} + \rho \underline{v} \cdot \nabla \underline{v}}_{\substack{\text{Lagrangian}\\ \text{derivative}\\ \text{of impulse}}} = \underbrace{\underbrace{-\nabla p}_{\text{pressure force}} + \underbrace{\mu \Delta \underline{v}}_{\text{viscous force}} + \underbrace{\rho f_v}_{\text{body force}}}_{\text{sources/sinks of impulse}} .$$

  The left-hand side gives the change of impulse felt by a Lagrangian fluid element per unit volume. According to Newton's law of mechanics, the temporal change of the impulse is given by the sum of all forces acting on the control volume. The first component on the right-hand side is the force per unit volume due to pressure differences across the control volume. If the pressure on two sides of the control volume differs, the volume feels a force that is opposed to the gradient. It is "sucked" to the side of lower pressure (or pushed away by the high pressure). The second force is the friction against neighboring control volumes. The quantity μ is the dynamic viscosity of the fluid (see Section 2.2). It is a constant material property that characterizes the "fluidity" of the fluid and it can be looked up in books or tables. Finally, ρf_v is the specific force that is applied to the fluid from external. If the fluid

is, e.g., experiencing gravity, it would be $f_v = -g\underline{e}_z$ with g the acceleration of gravity. Here, also other driving forces of the flow can be included.

We thus have two governing equations for the two unknown field quantities ρ and $\underline{v}$. The pressure will be of no importance, as we will see later.

9.2 Simulation using Particles: the Vortex Method

While the continuity equation is a compressible advection equation (without the diffusion part) that can easily be simulated by moving the particles and adapting their volumes, the Navier-Stokes equation is a bit more difficult. It is a non-linear (in $\underline{v}$) equation for the intensive properties ρ and $\underline{v}$ (subdividing a control volume leaves the velocity unchanged). While the corresponding extensive quantity of the density ρ is mass, the extensive quantity that corresponds to the velocity $\underline{v}$ has no useful conservation properties. In order to represent the dynamics of the Navier-Stokes equation on particles, we thus have to use a trick to introduce another extensive quantity. The trick is to take the curl of the entire equation, which converts the velocity $\underline{v}$ to the *vorticity* $\underline{\omega} = \nabla \times \underline{v}$. Vorticity has an associated extensive quantity, namely the *circulation* $\underline{\Gamma} = \int \underline{\omega}\, dV$, relating to the rotational energy stored in a fluid element. Circulation is a conserved quantity (Kelvin's theorem), which enables us to use our familiar control volume method for modeling and particles for the corresponding simulations. Taking the curl of the Navier-Stokes equation, we obtain:

$$\nabla \times \left(\rho \frac{\partial \underline{v}}{\partial t}\right) + \nabla \times (\rho \underline{v} \cdot \nabla \underline{v}) = \overbrace{-\nabla \times (\nabla p)}^{=0 \text{ (rule 12)}} + \nabla \times (\mu \Delta \underline{v}) + \nabla \times (\rho f_v)$$

$$\underbrace{\rho \nabla \times \frac{\partial \underline{v}}{\partial t} - \frac{\partial \underline{v}}{\partial t} \times \nabla \rho}_{\text{(rule 10)}} + \rho \nabla \times (\underline{v} \nabla \underline{v}) - \underline{v} \nabla \underline{v} \times \nabla \rho = \mu \nabla \times (\Delta \underline{v}) + \rho \nabla \times f_v \,.$$

This equation can be simplified by restricting ourselves to incompressible flows. In incompressible flows, the velocity field is divergence-free, thus $\nabla \cdot \underline{v} = 0$ (see Sec. 8.3). This is equivalent to saying that the density does not vary, thus $\rho = \not{\rho}$. This is a good approximation for most liquids as well as for gases that move at velocities much smaller than the speed of sound (e.g., air in the lungs). Setting $\rho = \not{\rho}$, and dividing the whole equation by ρ, we obtain:

$$\nabla \times \frac{\partial \underline{v}}{\partial t} + \nabla \times (\underline{v} \nabla \underline{v}) = \nu \nabla \times (\Delta \underline{v}) + \nabla \times f_v \,.$$

The quantity ν is the *kinematic viscosity* of the fluid, defined as $\nu = \mu/\rho$. It is the ratio of viscous forces to inertial forces and defines the diffusion constant for the impulse transport due to internal friction in the fluid. Expanding the second term and grouping the curls of the velocity field, we find:

$$\frac{\partial}{\partial t}(\nabla \times \underline{v}) + \underline{v} \cdot \nabla(\nabla \times \underline{v}) + (\nabla \times \underline{v}) \cdot \nabla \underline{v} = \nu \Delta(\nabla \times \underline{v}) + \nabla \times f_v \,.$$

Substituting the vorticity $\underline{\omega} = \nabla \times \underline{v}$ leads to the reformulated governing equation:

$$\frac{\partial \underline{\omega}}{\partial t} + \underline{v} \cdot \nabla \underline{\omega} + \underbrace{\underline{\omega} \cdot \nabla \underline{v}}_{\text{vortex stretching}} = \underbrace{\nu \Delta \underline{\omega}}_{\substack{\text{Vorticity diffusion due to} \\ \text{viscous friction}}} + \nabla \times f_v \,,$$

the Navier-Stokes equation in vorticity form. The first two terms of this governing equation for the vorticity field $\underline{\omega}$ are the material derivative. We can thus easily formulate the equation in Lagrangian form:

$$\boxed{\frac{D\underline{\omega}}{Dt} + \underline{\omega} \cdot \nabla \underline{v} = \nu \Delta \underline{\omega} + \nabla \times f_v} \,. \tag{9.1}$$

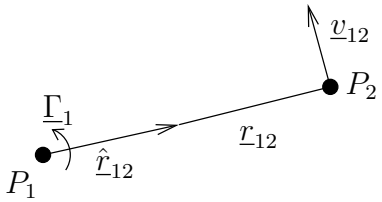
Notice that by definition: $\nabla \cdot \underline{\omega} = \nabla \cdot (\nabla \times \underline{v}) \equiv 0$ (rule 11).

The Navier-Stokes equation in vorticity form still describes the same physical system as the incompressible Navier-Stokes equation in velocity-pressure form, since all we did was applying the same mathematical operations to both sides of the equation. Notice, however, that the pressure term has vanished. This is natural, since in an incompressible fluid the pressure is not an independent quantity and can always be computed from the velocity field. Also notice that the Eulerian form of the Navier-Stokes equation is non-linear in $\underline{v}$, whereas the above Lagrangian formulation is linear. The non-linearity in the system is gone! This might seem surprising or impossible, but there are in fact no "physical" non-linearities in incompressible flows. The mathematically non-linear form of the velocity-pressure form of the Navier-Stokes equation comes from the fact that this formulation is also valid for compressible flows, which are indeed non-linear (see equation before assuming incompressibility). The above vorticity formulation is only valid for incompressible flows and is hence linear. This allows, e.g., using the superposition principle for solving problems. Due to its general formulation, the velocity-pressure form allows also compressible solutions. When simulating incompressible flows, much effort needs to be devoted to projecting them out again at the end. The vorticity formulation does not require this.

Neglecting for now the vortex stretching term, the incompressible Navier-Stokes equation in vorticity form is an incompressible advection-reaction-diffusion equation for the vorticity $\underline{\omega}$. It can thus be simulated using our existing particle framework, where here the particles carry circulation as their extensive property and are advected with velocity $\underline{v}$. But how to determine $\underline{v}$?

Computing $\underline{v}$ on the particles

The circulation that the individual particles carry can be thought of as a rotational energy that is associated with the local vorticity. Recall from Section 3.2 that the curl of a field is the vortex strength density. The circulation of a particle induces a rotational velocity field about that particle, as if the particle would spin around, whirling the fluid around it. This induces a tangential velocity on all other particles. Since every particle is doing this simultaneously, and the problem is linear, the total velocity field of the flow is given by the superposition of all these rotational components.



The velocity induced by the circulation $\underline{\Gamma}_1$ of particle 1 on particle 2 is:

$$\underline{v}_{12} = (\underline{\Gamma}_1 \times \underline{r}_{12}/||\underline{r}_{12}||)\,/||\underline{r}_{12}||^2\,.$$

The total velocity of particle 2 is then given by the influences from all other particles, thus:

$$\underline{v}_2 = \sum_{p=1}^{N\,2} \frac{\underline{\Gamma}_p \times \underline{r}_{p2}/||\underline{r}_{p2}||}{||\underline{r}_{p2}||^2}\,.$$

The resulting sum is known as the "Biot-Savart law", which also occurs in electromagnetism when evaluating Coulomb interactions.

Since this velocity has to be evaluated for all particles, and not only for particle 2, we have an N-body problem. For each of the N particles we have to sum the velocity contributions from all other $N-1$ particles, leading to a total of $O(N^2)$ interactions that are to be computed. Since the potential is not local, we cannot restrict the sum to the nearest neighbors as we could, e.g., in diffusion. Even particles far away contribute significantly to the velocity and must be accounted for. We thus have to solve the full N-body problem. It was this high computational cost that has long prevented the use of particle methods for flow problems. Since the mid-1980's, however, fast N-body solvers are available to reduce the computational cost to $O(N)$, albeit with a large pre-factor. These methods include the Barnes-Hut algorithm and the Fast Multipole Method (FMM). Alternatively, one can use a hybrid particle-mesh approach to render the computations more efficient. Hybrid particle-mesh simulations are also viable since the mid-1980's with the availability of moment-conserving interpolation schemes. We will follow this approach since the implementation of fast N-body solvers is involved and their computational performance is inferior to hybrid particle-mesh methods. Moreover, hybrid particle-mesh methods provide at least the same accuracy as fast N-body solvers and we anyway need a mesh for remeshing (see Section 8.4), so we can directly use the same mesh also for evaluating the long-range part of the operator, i.e., computing the velocity field, without needing any additional data structures.

Computing $\underline{v}$ on the mesh

In order to compute the velocity field on the mesh, we transform the problem to the corresponding PDE that can then be solved on the mesh in $O(N)$ time. Consider the definition of $\underline{\omega}$:

$$\nabla \times \underline{v} = \underline{\omega}\,.$$

Taking the curl of the whole equation, and applying compute rule 14 for differential operators, we obtain:

$$\begin{aligned} \nabla \times (\nabla \times \underline{v}) &= \nabla \times \underline{\omega} \\ \nabla(\nabla \cdot \underline{v}) - \Delta \underline{v} &= \nabla \times \underline{\omega} \qquad \text{(rule 14)} \quad (*)\,. \end{aligned}$$

So far, we have only used the Navier-Stokes equation in deriving our particle simulation scheme. Now it is time to look also at the continuity equation. Expanding it using compute rule 7 for differential operators, we have:

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{v}) = 0 \\ &\frac{\partial \rho}{\partial t} + \underline{v} \cdot \nabla \rho + \rho \nabla \cdot \underline{v} = 0 \qquad \text{(rule 7)}\,. \end{aligned}$$

Since we have already made the assumption of incompressibility when deriving the vorticity form of the Navier-Stokes equation, we can also simplify the continuity equation by setting $\rho = \not{\rho}$, yielding:

$$\begin{aligned} \rho \nabla \cdot \underline{v} &= 0 \\ \nabla \cdot \underline{v} &= 0\,. \end{aligned}$$

The continuity equation for incompressible flows thus reduces to the condition that the velocity field has to be divergence free. That's why the two definitions of incompressibility are equivalent (i.e., linked through conservation of mass).

Using $\nabla \cdot \underline{v} = 0$ in equation $(*)$, we find

$$\boxed{\Delta \underline{v} = -\nabla \times \underline{\omega}}\,. \tag{9.2}$$

This is a Poisson equation that links $\underline{\omega}$ and $\underline{v}$. Solving this equation for $\underline{v}$ on the mesh thus allows computing the flow velocity field from the vorticity field. Notice that this velocity equation implicitly contains the condition $\nabla \cdot \underline{v} = 0$. All solutions will thus automatically be divergence free. This is in contrast to grid based methods, where additional effort has to be devoted to ensuring divergence-freeness of the solution.

9.3 Algorithm

The present hybrid particle-mesh method to simulate incompressible flows is called "vortex method" since it is based on the vorticity form of the Navier-Stokes equation. For the compressible case, other particle methods are available, such as smoothed particle hydrodynamics (SPH) or moving least-squares (MLS), to numerically approximate the solution of the *compressible* Navier-Stokes equation. Since in biology flow velocities close to the speed of sound do not frequently occur, we limit our discussion to incompressible flows.
Simulating incompressible convection amounts to simulating advection-reaction-diffusion of vorticity (Eq. 9.1) with a velocity field that is determined by solving a Poisson equation (Eq. 9.2). The simulation algorithm thus proceeds as follows:

Initialize particles at locations $\underline{x}_p^0$ with circulations $\underline{\Gamma}_p^0$.
Do the time steps $n = 0, \ldots, T-1$:

- interpolate $\underline{\omega}$ from the particles onto the mesh
- solve $\Delta \underline{v} = -\nabla \times \underline{\omega}$ for $\underline{v}$ on the mesh using a fast Poisson solver (e.g. multigrid or FFT[1] that can be found in most scientific computing libraries. For M mesh points, the time complexity of a multigrid solver is $O(M)$ and that of FFTs is $O(M \log M)$. FFTs provide the exact solution (to machine precision), whereas multigrid algorithms provide an approximate solution. Multigrid algorithms can efficiently be parallelized on large multi-processor machines, whereas the scalability of FFTs is limited due to the global communication that is necessary.)
- interpolate $\underline{v}$ from the mesh back to the particles $\Rightarrow \underline{v}_p^n$
- compute vortex stretching $\underline{\omega} \cdot \nabla \underline{v}$ on the mesh using, e.g., finite differences
- interpolate the vortex stretching result from the mesh to the particles as a change of vorticity
- compute vorticity diffusion $\nu \Delta \underline{\omega}$ (isotropic and homogeneous) on the particles using PSE (or RW) (alternatively this can also be computed on the mesh and interpolated back), add this vorticity change to the one from the stretching term, and update the particle circulation. Also add the curl of the body force, $\nabla \times f_v$, if present. $\Rightarrow \underline{\Gamma}_p^{n+1}$
- move (advect) the particles with velocity $\underline{v}_p^n$ and update their positions, leaving the particle volumes unchanged since advection is incompressible. $\Rightarrow \underline{x}_p^{n+1}$
- remesh if needed. If remeshing is done at each time step, the mesh-particle interpolation of the quantities computed on the mesh can be skipped and directly new particles generated. Generate particles only at mesh nodes where $|\underline{\omega}| > 0$.

end time step.

Compared to pure particle methods that use Fast Multipole Methods, hybrid particle-mesh approaches have the following advantages:

- easy treatment of the vortex stretching term on the mesh
- no loss of accuracy since the interpolation kernels are moment-conserving and remeshing is needed anyway
- they are orders of magnitude faster than fast multipoles (both are $O(N)$, but FMM have a larger pre-factor)
- they are efficiently parallelizable (FMM requires a global tree data structure)

9.4 Properties of Vortex Methods

In the above algorithm, we do not need any new subroutines to simulate incompressible flows. All we need are the advection-reaction-diffusion solver from the last chapters, a mesh-based Poisson solver (that can be found in any standard scientific computing library), and the moment-conserving interpolation schemes.

Vortex methods have a number of unique properties:

- Particles are only needed where $|\underline{\omega}| > 0$ ($\Rightarrow$use an epsilon cutoff after remeshing!). This makes the method inherently adaptive. In mesh-based methods, a grid is usually required throughout the computational domain and it does not track the flow.
- There is no linear CFL condition, but only the Lagrangian CFL condition (see Sec. 8.2.1).

[1] Solving the Poisson equation with FFT (exact to machine precision):

$$\begin{array}{ccccc} \frac{\partial^2 u}{\partial x^2} = f & \xrightarrow{\mathcal{F}} & (ik)^2 \hat{u} = \hat{f} \\ f & \xrightarrow{\mathcal{F}} & \hat{f} \\ & & \downarrow \frac{1}{(ik)^2} \\ u & \xleftarrow{\mathcal{F}^{-1}} & \hat{u} \end{array}$$

- The governing equation for $\underline{\omega}$ is linear, whereas the Navier-Stokes equation for $\underline{v}$ and p is non-linear. This is a direct effect of the Lagrangian formulation.
- Divergence-freeness of the velocity field is automatically guaranteed, such that only physical solutions are possible. This largely eliminates the need for projection schemes that are required in some mesh-based methods.
- The implementation of vorticity boundary conditions is difficult. In complex geometries we need, e.g., immersed boundaries, Brinkman penalization, Boundary Element Methods, or vortex sheets in order to satisfy the boundary conditions. This is more involved than the simple boundary condition handling that is possible in mesh-based methods.

Chapter 10

Waves

In this chapter:

- The governing equations for waves
- The general wave solution and properties of waves
- Definitions of wave-related terms
- The behavior of waves at boundaries
- Standing waves and oscillations
- Simulating waves using particles

Learning goals:

- Know the governing equation and its general solution by heart
- Be able to define wave terms and know how they are interrelated
- Know how waves are reflected at boundaries and be able to explain this using the method of images
- Understand oscillations as standing waves
- Be able to solve the wave eqution using particle methods

After having seen how to model and simulate diffusion, reaction, advection, and convection processes, we now consider waves as the last of the biologically relevant transport phenomena discussed in this course. But there is one fundamental difference to the phenomena considered so far: Waves transport energy are not associated with any transport of mass. One has to be careful not to confure waves with traveling fronts in reaction-diffusion systems as discussed in Chapter 7. Biological phenomena that transport mass, such as Calcium waves in muscle cells or cAMP waves in cell signaling, are usually traveling reaction-diffusion fronts and not waves. Nevertheless, they are frequently termed "waves" in the biological literature. Care is needed not to postulate the wrong model then. Waves are also intimately realted to the concept of oscillations, as we will see in this chapter.
In biological systems, the following types of waves frequently play a role:

- electro-magnetic waves in neurons
- elastic waves in membranes, tissues, and bones
- acoustic pressure waves in the air

In addition, there can of course also be other waves not listed here.

10.1 Governing Equation

The governing equation for waves has the same right-hand side as the diffusion equation, but a second-order time derivative on the left-hand side:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u(\underline{x}, t)\,.$$

This is the general wave equation in n dimensions. Everything that is described by this equation is by definition considered a wave. In most practical applications, waves travel along a certain direction. We are thus often interested in the one-dimensional wave equation formulated along the direction x of propagation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \qquad (*)\,.$$

Another very important case are spherically symmetric waves originating from a point source. Also in this case the wave equation becomes one-dimensional with the only space coordinate being the radial distance r:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial r^2} + \frac{2}{r} \frac{\partial u}{\partial r} \right)\,.$$

This equation follows by expressing the Laplace operator in spherical coordinates.

10.2 General Solution

The one-dimensional wave equation can be solved analytically. This provides the important advantage of being able to understand the general behavior of the solutions in all cases (numerical simulations only give information about the solution in one particular situation). Numerical simulations then often directly implement the analytical solution.

There are several ways of analytically solving the wave equation. Here, we use Fourier transforms to do so. This is along the same lines as the use of Fourier transforms to solve the Poisson equation as discussed in the previous chapter. In

general, Fourier transforms reduce the order of all mathematical operations by one: multiplications in real space become additions in frequency space, convolutions and derivatives become multiplications, integrals become divisions, ODEs become algebraic equations, and PDEs become ODEs. Frequency space methods are thus very powerful for both numerical and analytical solution of equations. While the Fourier transform is the simplest frequency space transformation, there are also others such as the Laplace transform (for dissipative systems) or the z-transform (for discrete systems). The Fourier transform of $(*)$ in x is:

$$c^2(ik)^2U(k,t) - \frac{\partial^2 U(k,t)}{\partial t^2} = 0\,,$$

where U is the Fourier-transformed field u, i.e. the spectrum of spatial frequencies in u, i is the imaginary unit, and k the frequency. The temporal evolution of each Fourier mode (frequency) is thus governed by the following ODE (not a PDE any more!):

$$-c^2k^2U - U''(t) = 0\,.$$

This ODE is linear, homogeneous, and has constant coefficients. It can thus be solved for each k analytically and the general solution is:

$$U(k,t) = C_1e^{ickt} + C_2e^{-ickt}\,.$$

Determining the values of the integration constants C_1 and C_2 requires initial conditions that specify the initial wave form and the initial speed of the wave as:

$$\begin{aligned} U(k,0) &= F(k) \\ U'(k,0) &= G(k)\,. \end{aligned}$$

The integration constants then are:

$$\begin{aligned} U(k,0) &= C_1 + C_2 = F(k) \\ U'(k,0) &= ickC_1 - ickC_2 = G(k) \end{aligned}$$

$$\begin{aligned} C_1 &= \frac{1}{2}\left(F + \frac{1}{ick}G\right) \\ C_2 &= \frac{1}{2}\left(F - \frac{1}{ick}G\right)\,. \end{aligned}$$

Substituting these constants into the general solution yields the final, analytical solution of the wave equation in frequency space:

$$U(k,t) = \frac{1}{2}\left(F + \frac{1}{ick}G\right)e^{ickt} + \frac{1}{2}\left(F - \frac{1}{ick}G\right)e^{-ickt}\,.$$

The solution in real space is found by inverse Fourier transform. For this backward transform we need the Fourier translation identity:

$$\mathcal{F}\{f(a(x+b))\} = \frac{1}{a}F\left(\frac{k}{a}\right)e^{ibk}\,.$$

Terms of the form $e^{ickt}F(k)$ in our analytical solution correspond to the right-hand side of the Fourier translation identity with $a = 1$ and $b = ct$. We can thus directly transform these two term back as:

$$\begin{aligned} \mathcal{F}^{-1}\{e^{ickt}F(k)\} &= f(x+ct) \\ \mathcal{F}^{-1}\{e^{-ickt}F(k)\} &= f(x-ct)\,. \end{aligned}$$

The terms containing G are similar, but contain in addition a division by the wave number (ik). Divisions by the wave number in frequency space correspond to integration in real space. We can thus transform these terms back as:

$$\begin{aligned} \mathcal{F}^{-1}\left\{\frac{1}{c}\frac{1}{ik}Ge^{ickt}\right\} &= \frac{1}{c}\int_{-\infty}^{x}\mathcal{F}^{-1}\{Ge^{ickt}\}dt \\ &= \frac{1}{c}\int_{-\infty}^{x} g(\tau + ct)\,d\tau\,. \end{aligned}$$

The integral can be further simplified using the substitution $z = \tau + ct$, yielding:

$$\frac{1}{c}\int_{-\infty}^{x+ct} g(z)\,dz\,.$$

For the back-transform of the second G term, we find analogously:

$$\mathcal{F}^{-1}\left\{-\frac{1}{c}\frac{1}{ik}Ge^{-ickt}\right\} = -\frac{1}{c}\int_{-\infty}^{x-ct} g(z)\,dz\,.$$

The first term amounts to an integral over the initial wave velocity from $-\infty$ to $x+ct$. From this, the second term subtracts the integral from $-\infty$ to $x-ct$. What remains thus is the integral from $x-ct$ to $x+ct$, such that the analytical solution in real space reads:

$$u(x,t) = \frac{1}{2}f(x+ct) + \frac{1}{2}f(x-ct) + \frac{1}{2c}\int_{x-ct}^{x+ct} g(z)dz\,,$$

with initial conditions: $u(x,0) = f(x)$
$u_t(x,0) = g(x)$.

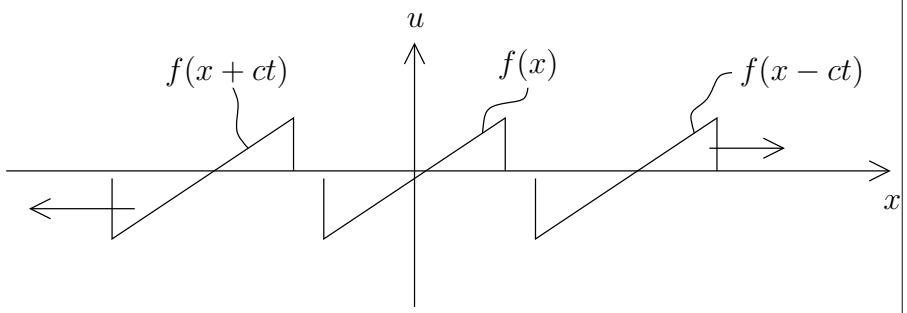
This solution can be interpreted as follows: the function f is the wave form. Half of it then travels along the x axis (the direction of propagation) with speed c, the other half travels in the opposite direction with the same speed. The integral is over the entire part of the domain that the waves have already visited, i.e. from $x-ct$ to $x+ct$. This region is called the *domain of dependence* of the wave and it comprises all points in space where the wave has already been "felt". The function g is the

rate at which the values of f change, i.e. the velocity (in time) of the deflection of the wave at any fixed position x. By definition, a wave preserves its shape as it travels. In order for this to be possible, the total increase and total decrease in deflection have to be equal, i.e. the area under f is conserved as the wave travels. As we integrate over the entire domain of dependence, the integral has to vanish and we find the final general solution of the wave equation as:

$$\boxed{u(x,t) = \frac{1}{2}f(x+ct) + \frac{1}{2}f(x-ct)}\,.$$

The general solution of the one-dimensional wave equation thus is the superposition of a wave of shape f traveling to the left and to the right along the direction of propagation. Both partial waves travel with the same speed c:

$$\underbrace{f(x+ct)}_{\text{to the left}} + \underbrace{f(x-ct)}_{\text{to the right}}$$



10.3 Properties of Waves

Knowing the analytical solution of the wave equation allows us to discuss some general properties of waves. The most important observation is that a wave travels without deforming. The initial spatial distribution (deflection) is given by the function $f(x)$. After a time Δt, the very same spatial distribution is found at position $\pm c\Delta t$. In energy-transporting waves, the actual material particles (molecules, etc.) are not advected, but undergo oscillatory deflections, which give the "visual impression" of a traveling wave. This is analogous to waves on water, where the water is not advected, but the undulations of the water surface create the impression of a traveling wave. Depending on the direction of deflection of the material particles, we distinguish *longitudinal waves* and *transversal waves*.

In longitudinal waves, the particles are deflected in the same direction as the wave propagates. The oscillatory motion of the particles and the direction of wave propataion are thus parallel. This is illustrated in Fig. 10.1. Each particle periodically oscillates around its center position in x. The phase shift between particles creates the impression of a traveling density wave. At any given time t, particles are more dense in some regions of space and less dense in others. This density fluctuation travels according to the wave equation. Figure 10.1 shows the locations of particles in a 1D example over time t. Over a full periode τ of the oscillations, a given density minimum (highlighted by the arrow) travels a distance that is equal to the wavelength λ.

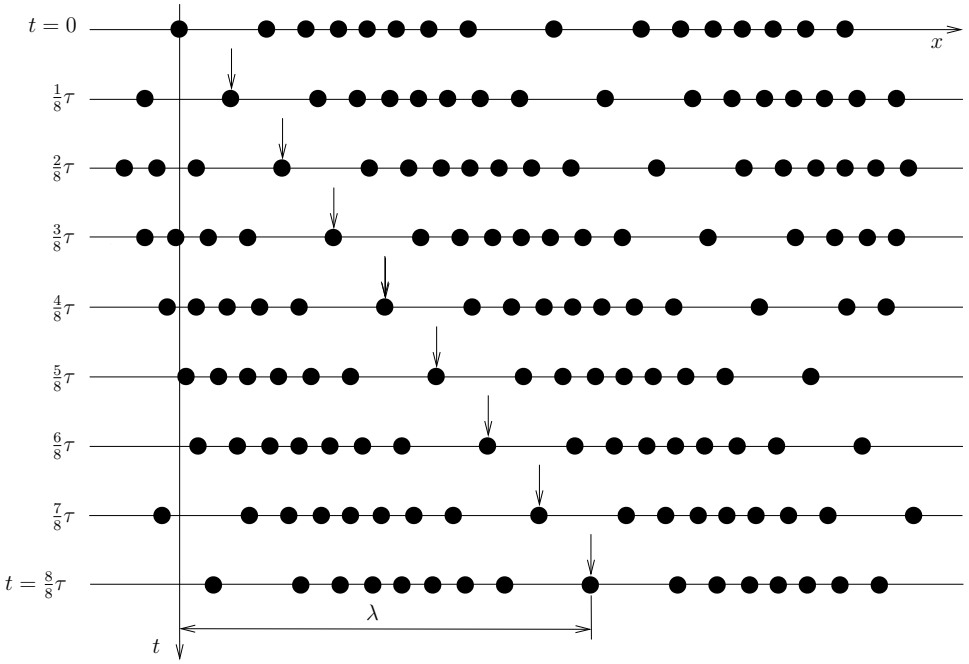
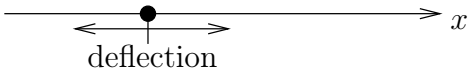


Figure 10.1: Longitudinal wave: the particles oscillate in the direction of wave propagation x. Each particle periodically oscillates around its center position. The phase shift between neighboring particles creates the impression of a traveling density wave.

Example 10.3.1 (sound waves in air)**.**
Sound waves in air are pressure waves. The air molecules oscillate in the direction of sound propagation. This causes a traveling density wave because air is compressible at these speeds. Sound waves thus are longitudinal waves.



In transversal waves, the deflection of the particles is orthogonal to the direction of wave propagation. (Picture for example the hands of spectators in the stadium, doing "the wave". People do not actually displace, but the coordinated up-down motion of their hands creates the impression of a traveling wave.) Transversal waves are illustrated in Fig. 10.2 on the example of a sine wave. Even though the physical particles do not move in x, but oscillate only in y, the phase shift between particles creates the impression of a traveling waveform. The figure shows the wave form at different times t over a full periode τ of the oscillations. It is clearly visible that the wave form behaves according to the wave equation. The arrow marks a particular maximum of the wave form, the distance between consecutive maxima is equal to the wavelength λ.

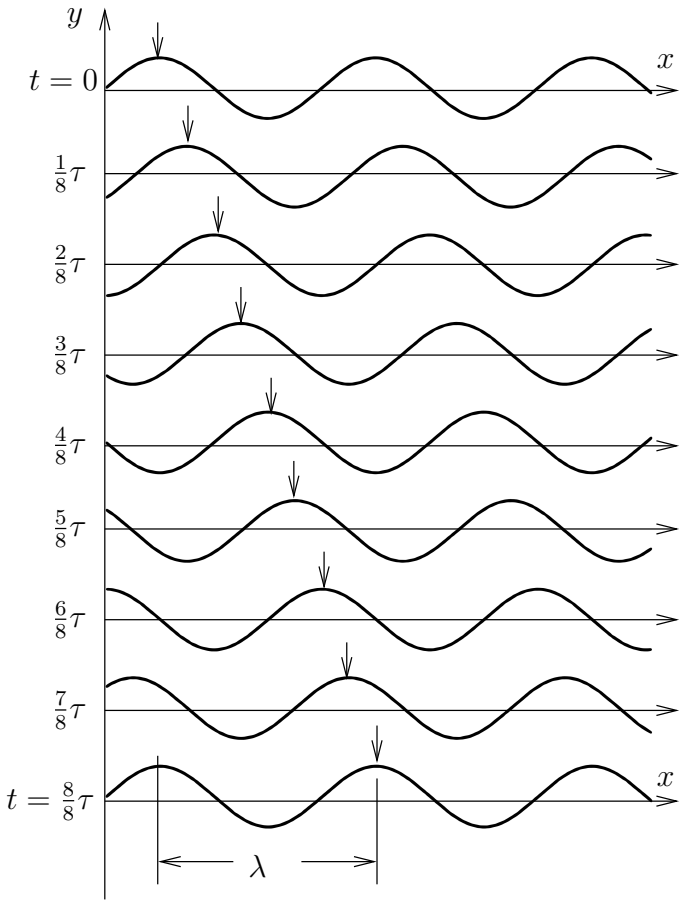
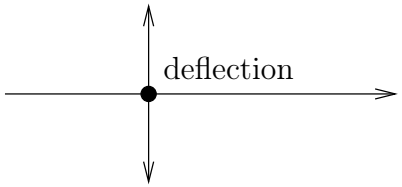


Figure 10.2: Transversal wave: the particles oscillate in a direction perpendicular to the wave's direction of propagation. Even though the particles do not move in x when they oscillate, the phase shift between neighboring particles creates the impression of a traveling wave.

Example 10.3.2 (electromagnetic waves)**.**
In electromagnetic waves, the electric and magnetic fields oscillate in directions perpendicular to the direction of wave propagation. In addition, the electric and magnetic field vectors are orthogonal to each other such that they span the plane normal to the direction of wave propagation.



10.4 Definitions

We define a few commonly used terms and expressions pertaining to waves:

Wave front: The wave front is the location where the temporal oscillations of all particles are in phase, i.e., $\{x : u(x,t) = K \text{ for fixed } t\}$). For isotropic media, where the wave propagation speed c is constant and does not depend on the location x, the wave fronts of point sources are:

- concentric circles in 2D
- concentric spheres in 3D

Ray: The ray of a wave is the path along which the wave travels. The ray is always perpendicular to the wave front at every point in space and time.

Phase velocity: The phase velocity is the velocity with which an observer would have to travel in order to always see the same function f. It is the velocity c in the governing equation.

Wavelength: The wavelength is the spatial period of the wave form f. It is usually denoted by the symbol λ.

Frequency: The frequency ν is the number of oscillations the particles undergo per unit time, which is equal to the number of maxima in $u(x,t)$ that travel through a fixed point x per unit time. The frequency and wavelength are thus linked to the phase velocity as $\boxed{c = \lambda\nu}$.

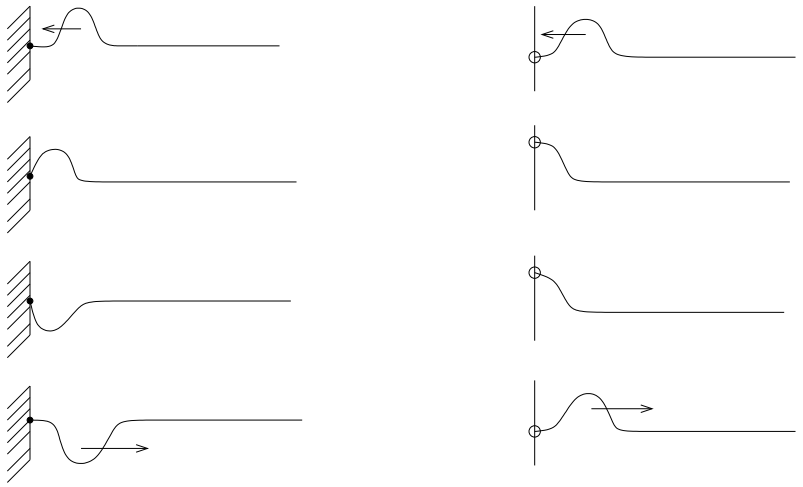
Amplitude: The amplitude A is the maximal deflection of the wave, i.e. the maximum absolute value of $f(x)$.

Intensity: The intensity of a wave is related to the amount of energy that is stored in the wave. It is given by the square of the amplitude, thus: $I = A^2$.

Harmonic wave: A harmonic wave is characterized by a sinusoidal wave form, thus $f(x) = A\sin(kx + \varphi)$.

Spherical wave: A spherical wave propagates spherically symmetric in radial direction, thus: $u(r,t) = \frac{1}{2r}f(r - ct) + \frac{1}{2r}f(r + ct)$. The damping factor $\frac{1}{r}$ that is not present in linear 1D waves is needed for conservation of energy. As the wave front propagates outward from the point source, its surface area grows with r^2. The energy that is stored in the wave is given by the intensity times the wave front area. The intensity thus has to decrease as $\frac{1}{r^2}$ with increasing distance from the source. This translates to a $1/r$ decay in amplitude. This quadratic decay in intensity with distance from the source is, by the way, the reason why radio stations can not be heard arbitrarily far, or light becomes dimmer farther away from the candle.

10.5 Boundary Conditions



Dirichlet: rope anchored in the wall. Neumann: rope end slides along pole.

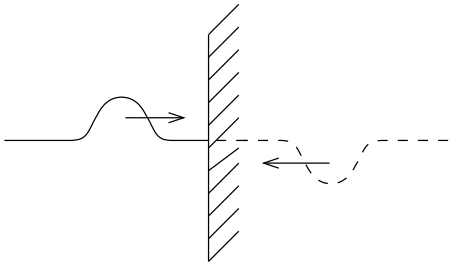
So far we have only considered waves traveling through open space ad infinitum without ever being disturbed by a boundary. This is of course not realistic, and we will now see how waves behave when they hit a boundary. There are two types of boundaries: absorbing ones and reflecting ones. In many applications, the boundary condition is a mixture of these two types, characterized by a reflectivity coefficient and an absorbance coefficient. The two parts can be treated separately and then superimposed as a weighted sum since the wave equation is linear. If a wave hits an absorbing boundary, it disappears. This is the same as imagining the wave to "run through" the boundary and continue behind it, never coming back. At a reflective boundary, the wave is reflected. Reflective boundaries can be either Dirichlet or Neumann boundaries (cf. Sec. 5.4). Dirichlet boundaries model hard (solid) walls where the wave is reflected with a phase shift of π, and Neumann

boundaries model soft (mobile) walls where the wave is reflected without phase shift.

10.5.1 The method of images

Reflective boundary conditions can be implemented and intuitively understood using the method of images (cf. Sec. 5.4). We imagine the wave to travel through the boundary without feeling its presence. We further imagine that another wave emerges from behind the boundary (i.e. crosses over from the other side). This will form the reflected wave. Since the wave equation is linear, the two waves can be superimposed by adding their deflections everywhere.

In order to satisfy $u(\text{boundary}, t) = 0$, the wave emerging from behind the boundary has to be a mirrored version of the incident wave. Then the two waves always have the same deflection magnitude at the boundary, but with opposite sign. This is where the phase inversion comes from.



For a homogeneous Neumann boundary condition, the emerging wave has to be an exact copy of the indicent wave, without phase inversion. Then the two waves always have opposite gradients at the boundary.

10.6 Standing Waves and Oscillations

In open space, a wave travels along its direction of propagation without changing its shape. If a wave is reflected at a boundary, however, it will travel back into the domain, where it is superimposed with the indicent wave. The final wave we see is the sum of the incident wave and the reflected wave.

Consider a harmonic incident wave $u_A = A\cos(kx - \omega t)$ with phase velocity $c = \frac{\omega}{k}$ that is reflected at a Dirichlet wall. The reflected wave is

$$u_R = \overset{\text{phase jump}}{-} A\cos(kx + \omega t)$$

and travels in the opposite direction. In the domain, we thus see the superposition

$$u = u_A + u_R = \underbrace{2A\sin kx}_{\text{space}}\underbrace{\sin\omega t}_{\text{time}},$$

where we have used the trigonometric identity $\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}$. The first sine factor of the sum of the two waves is a space-dependent amplitude, whereas the second factor is a purely temporal oscillation. The resulting wave thus appears stationary with all maxima and minima remaining fixed in space! This is called a *standing wave* and amounts to apparent spatiotemporal oscillations. Standing waves only exist with Dirichlet walls, not with Neumann boundaries.

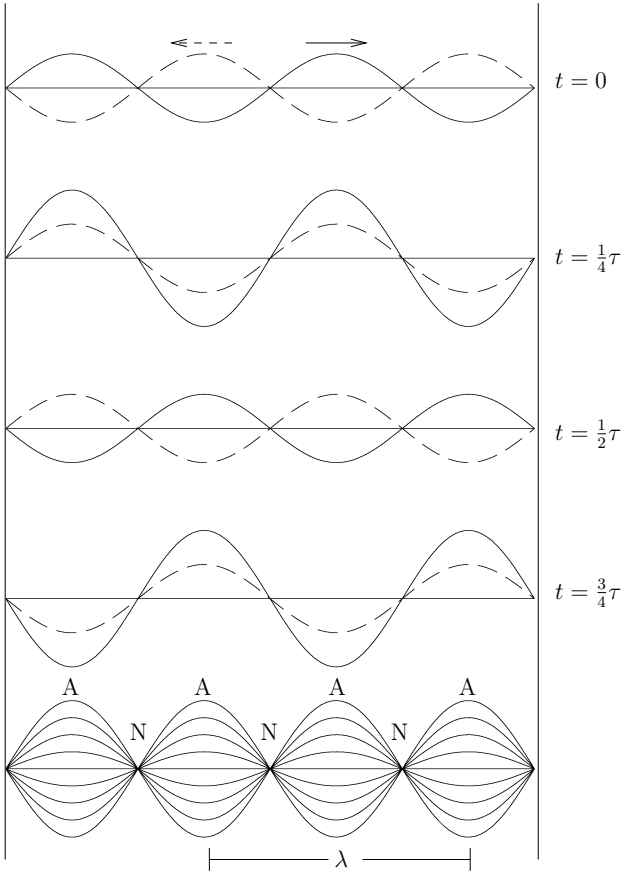


Figure 10.3: A wave bouncing back and forth between two Dirichlet walls that are a distance of $n\lambda/2$ apart for integer n leads to a standing wave with group velocity zero. At times $t = 0$ and $t = \frac{1}{2}\tau$, the incident wave and the reflected wave are shifted by exactly π in phase. They thus cancel each other and the amplitude of the resulting sum wave is zero everywhere. At times $t = \frac{1}{4}\tau$ and $t = \frac{3}{4}\tau$, the wave and its reflection are in phase and the amplitude of the sum wave is double the amplitude of the incident wave. This gives rise to a standing wave that does not seem to move and has an amplitude of zero at the nodes (N) and of $2A$ at the maxima (A).

If a wave is trapped between two Dirichlet walls, then it keeps bouncing back and forth between them, giving rise to a permanent standing wave if the distance between the two walls (i.e. the length of the domain) is an integer multiple of half the wavelength, thus $n\frac{\lambda}{2}$ with integer $n > 0$ (Fig. 10.3). Observing oscillations in an experimental system thus already tells us that domain length must be $n\frac{\lambda}{2}$ for some positive integer n. This is the case for the MinC-MinD system in *E. coli*, and also for the growth factor gradients in *Drosophila* embryos.

If the domain is not of length $n\frac{\lambda}{2}$, the zeros and maxima of the resulting wave move with a *group velocity* (velocity of the wave packets) that is $< c$. The group velocity increases with increasing deviation of the domain length from $n\frac{\lambda}{2}$. As the domain length approaches $n\frac{\lambda}{2}$ for the next integer n, the group velocity decreases until the wave stands still again.

Example 10.6.1 (Wheels)**.** Sometimes in movies, the wheels of cars seem to spin backward. This is the case if the frame rate of the movie is not an integer multiple of the angular velocity of the wheel. What we see in the movie is the apparent group velocity, which can be negative, instead of the phase velocity (the true angular velocity of the wheel).

10.7 Simulation using Particles

So far, we have focused on the one-dimensional wave equation with constant and homogeneous phase velocity, its analytical solution, and the properties of the solutions. If we want to solve the full n-dimensional wave equation, or waves with inhomogeneous or anisotropic phase velocities,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u \,,$$

then we have to resort to numerical simulations. Since the spatial part of the wave equation is identical to that of the diffusion equation, we can simply discretize Δu using the isotropic, homogeneous PSE scheme. The only difference to simulating diffusion is that we now have to use a 2^{nd} order time integrator such as, e.g., a velocity Verlet algorithm.

10.7.1 Numerically evaluating the general solution

Frequently, we are not interested in a full n-dimensional simulation of the wave equation, but we rather want to simulate the propagation and superposition of waves. This can be done by numerically evaluating the general analytical solution of the wave equation. In a one-dimensional domain of solution, this amounts to (linear) advection of the wave form $f(x)$ (supported on computational particles, not material particles!) with constant velocity c. All this requires is moving the particles on which the function u is represented. If we want to simulate waves in two- or three-dimensional spaces, we also have to keep track of the direction of propagation, i.e. the ray vector. When a wave is reflected at a boundary, the direction of propagation changes. We thus need a ray-tracing algorithm and then advect the particles along this ray.

10.7.2 Discretization of arbitrary differential operators on particles

While the right-hand side of the wave equation in Cartesian coordinates is identical to the isotropic, homogeneous diffusion operator, the radially symmetric wave equation also contains a term $\partial u/\partial r$. This can also be discretized numerically using PSE, since the PSE scheme is not limited to diffusion operators. It can be applied to any differential operator of any order (div, grad, ...). Let

$$L^{\underline{\beta}} = \frac{\partial^{|\underline{\beta}|}}{\partial x_1^{\beta_1} \partial x_2^{\beta_2}, \ldots, \partial x_d^{\beta_d}} \in \mathbb{R}^d$$

with $\underline{\beta} = (\beta_1, \beta_2, \ldots, \beta_d)$ and $|\underline{\beta}| = \sum \beta_i$ be an arbitrary (even mixed) differential operator of order $|\underline{\beta}|$. The PSE approximation (operator approximation on particles) for $L^{\underline{\beta}} f(\underline{x})$ then is:

$$L_h^{\underline{\beta}} f(\underline{x}_p) = \frac{1}{\epsilon^{|\underline{\beta}|}} \sum_{q=1}^{N} V_q (f(\underline{x}_q) \mp f(\underline{x}_p)) \eta_{\epsilon}^{\underline{\beta}} (\underline{x}_p - \underline{x}_q)\,,$$

where the kernel $\eta_{\epsilon}^{\underline{\beta}}(\underline{x}) = \frac{1}{\epsilon^d} \eta^{\underline{\beta}} \left(\frac{\underline{x}}{\epsilon}\right)$ is chosen according to Eldredge et al., J. Comput. Phys. 180, 2002. This is exactly the form of the general integral operator approximation presented in Section 5.2.1. The sign in the sum is negative for even orders $|\underline{\beta}|$ (e.g. the Laplacian) and positive for odd orders $|\underline{\beta}|$ (e.g. the gradient), in order to kill the correct terms in the Taylor expansion. Near boundaries, special one-sided kernels are available in order to satisfy the boundary conditions. Since all of these generalized PSE kernels are local, we can use cell list or Verlet list to evaluate them in $O(N)$ time.

We now give a set of example kernels. They give an impression of what such generalized PSE kernels look like and they can also directly be used in simulations to evaluate first derivatives on particles. We denote by

$$\eta^{(\beta_1, \beta_2, \ldots, \beta_d)}$$

the kernel that approximates the β_1^{th} derivative in the x_1-direction, the β_2^{th} derivative in the x_2-direction, and so on. We then have the following kernels (taken from the paper by Eldredge et al.) for different orders of accuracy:

One-dimensional first derivatives, full space

$$\eta^{(1)}(x) = \frac{x}{\sqrt{\pi}} e^{-x^2} \times \begin{cases} (-2), & \text{second order,} \\ (-5 + 2x^2), & \text{fourth order,} \\ (-\frac{35}{4} + 7x^2 - x^4), & \text{sixth order,} \\ (-\frac{105}{4} + \frac{63}{4} x^2 - \frac{9}{2} x^4 + \frac{1}{3} x^6), & \text{eighth order.} \end{cases}$$

One-dimensional first derivatives, left sided

$$\eta^{L,(1)}(x) = \frac{x}{\sqrt{\pi}} e^{-x^2} \times \begin{cases} (-4), & \text{first order} \\ (-16 + 8x^2), & \text{second order} \\ (-40 + 44x^2 - 8x^4), & \text{third order} \\ (-80 + 144x^2 - 56x^4 + \frac{16}{3} x^6), & \text{fourth order} \end{cases}$$

(Right-sided kernels are identical to left-sided ones.)

One-dimensional second derivatives, full space

$$\eta^{(2)}(x) = \frac{1}{\sqrt{\pi}} e^{-x^2} \times \begin{cases} (4), & \text{second order} \\ (10 - 4x^2), & \text{fourth order} \\ (\frac{35}{2} - 14x^2 + 2x^4), & \text{sixth order} \\ (\frac{105}{4} - \frac{63}{2} x^2 + 9x^4 - \frac{2}{3} x^6), & \text{eighth order} \end{cases}$$

Two-dimensional first derivatives, full space

$$\eta^{(1,0)}(\underline{x}) = \frac{x_1}{\pi} e^{-|\underline{x}|^2} \times \begin{cases} (-2), & \text{second order} \\ (-6 + 2|\underline{x}|^2), & \text{fourth order} \\ (-12 + 8|\underline{x}|^2 - |\underline{x}|^4), & \text{sixth order} \\ (-20 + 20|\underline{x}|^2 - 5|\underline{x}|^4 + \frac{1}{3}|\underline{x}|^6), & \text{eighth order} \end{cases}$$

(The $(0,1)$ derivative is approximated using the same kernels with the x_1 factor replaced by x_2.)

Two-dimesional first derivatives, left sided

$$\eta^{L,(1,0)}(\underline{x}) = \frac{x_1}{\pi} e^{-|\underline{x}|^2} \times \begin{cases} (-4), & \text{first order} \\ (-20 + 8|\underline{x}|^2), & \text{second order} \\ (-60 + 52|\underline{x}|^2 - 8|\underline{x}|^4), & \text{third order} \\ (-140 + 196|\underline{x}|^2 - 64|\underline{x}|^4 + \frac{16}{3}|\underline{x}|^6), & \text{fourth order} \end{cases}$$

(Again, these kernels are identical to their right-sided counterparts, and they can be adapted for use in approximating the $(0,1)$ derivative by replacing x_1 by x_2.)

Two-dimensional Laplacian, full space

$$\eta^{\text{lap}}(\underline{x}) = \frac{1}{\pi} e^{-|\underline{x}|^2} \times \begin{cases} (4), & \text{second order} \\ (12 - 4|\underline{x}|^2), & \text{fourth order} \\ (24 - 16|\underline{x}|^2 + 2|\underline{x}|^4), & \text{sixth order} \\ (40 - 40|\underline{x}|^2 + 10|\underline{x}|^4 - \frac{2}{3}|\underline{x}|^6), & \text{eighth order} \end{cases}$$

Kernels for any other derivatives or differential operators in any dimensions and for any order of accuracy can be systematically derived using the scheme given by Eldredge et al.

Chapter 11

Partial Differential Equations (PDE)

In this chapter:

- Definition and classification of PDEs
- The superposition principle for linear PDEs
- Analytically solving PDEs using Fourier transforms
- Analytically solving PDEs by separation of variables
- Analytically solving PDEs using the method of characteristics

Learning goals:

- Be able to classify a given PDE with respect to its type and order
- Know what an ill-posed problem is
- Be able to attribute a linear PDE of second order to the proper sub-class
- Know what side conditions are needed to solve linear, second-order PDEs from each sub-class
- Be able to use the methods of Fourier transforms and separation of variables to solve simple PDEs

Partial differential equations play a central role when modeling spatiotemporal systems on the continuum scale and they are of key importance in many fields of physics, engineering, and biology. All governing equations that we have considered so far were partial differential equations, and the particle simulation method we have developed is a general numerical solution strategy for PDEs.
We therefore conclude our little journey of spatiotemporal modeling and simulation methods by looking at PDEs from a more general, mathematical point of view and address questions like: How is a PDE formally defined? What types and classes of PDEs exist? How can they be solved analytically? We give a high-level overview of the topic and describe the three most important methods to analytically solve PDEs.
We do, however, not study arbitrary PDEs, but focus on those classes of PDEs that have a real-world meaning, i.e., that are dimensionally homogeneous and obey the laws of thermodynamics. Examples of such PDEs include those derived from conservation laws (as in the previous chapters), transport theorems (such as the Reynolds transport theorem), and kinetic equations (for chemical reactions).

11.1 Definition

Definition 11.1.1. A partial differential equation (PDE) is an equation that relates partial derivatives of an unknown function $u : \Omega \to \mathbb{R}^n$ in multiple variables. Ω is an open subset of $\mathbb{R}^d$; $d > 1$.

This is in contrast to ordinary differential equations (ODE) that relate total derivatives of a function of only one single variable. If we erase the word "partial" in the above definition, and set $d = 1$, we thus obtain a definition for ODEs.

11.2 Classifications of PDEs

In order to gain an overview of the diversity and commonalities of PDEs, we start by giving the standard "taxonomy". PDEs can be classified according to a number of properties, including:

by order: The order of a PDE is defined as the highest occurring degree (in any variable) of the derivatives in the equation.

Example 11.2.1. The diffusion equation is second order since the highest derivative is of degree 2 and occurs in the variable x.

PDEs of order >4 hardly occur in modeling practice. The most important (physical) PDEs are second order. There are, therefore, special sub-classes defined for second-order PDEs (see below).

by type:

linear PDEs only contain linear combinations of the unknown function and its derivatives.

nonlinear PDEs also contain non-linear combinations (e.g., products) of the unknown function or its derivatives. In the class of non-linear PDEs, we further distinguish:

quasi-linear PDEs that are linear in the highest occuring derivative over all terms.

Non-linear PDEs are a difficult topic since no closed theory exists for this class of equations. They are notoriously difficult to solve and numerical simulations frequently suffer from instabilities and ill-posedness. Due to the lack of theory, the existence of solutions is often unclear, which hinders the development of better numerical simulation methods.

homogeneous PDEs where all terms (summands) contain the unknown function or one of its derivatives.

inhomogeneous PDEs where at least one summand is independent of the unknown function and its derivatives.

constant-coefficient PDEs where all pre-factors of the unknown function and its derivatives are independent of the independent variables (e.g., space or time).

varying-coefficient PDEs where the coefficients themselves are functions of at least one of the independent variables (e.g., space or time).

by solvability: (classification due to Hadamard)

solution existence: PDEs for which a solution *exists*. Solution existence is mostly proven by showing that the assumption that no solution exists leads to a contradiction.

solution uniqueness: PDEs that have only one, *unique* solution. If a solution is found, one knows that it is the only possible solution.

solution stability: In PDEs with *stable* solutions, small perturbations in the initial or boundary conditions only lead to small (bounded) variations in the solution.

PDEs that do not fulfill all 3 of the above conditions are called "ill-posed" and are hard or impossible to solve numerically.

Example 11.2.2 (deconvolution as an ill-posed problem)**.** Deconvolution is a central problem in image processing (undoing the blurring introduced by the microscope), signal processing, and linear systems theory. Although it is not a PDE, we still use it here as a simple illustrative example of an ill-posed problem. In deconvolution, the task is to find a function g such that $g * K = f$ for a given function f. If the kernel K contains low frequencies in its spectrum, computing the deconvolution is unstable because we divide (in frequency space) by numbers that are close to zero. Small changes in f (or noise!) are thus amplified and lead to arbitrarily large deviations in g.

Example 11.2.3 (In order to develop some intuition, we give a few examples of classifications of PDEs with respect to their order and type. Classification with respect to solvability is much more involved and we will not consider it here)**.**

1. $\frac{\partial u}{\partial t} = D\Delta u \quad \Rightarrow 2^{\text{nd}}$ order, linear, homogeneous, with constant coefficients (homogneous isotropic diffusion equation)

2. $\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u \quad \Rightarrow$same as above (wave equation)

3. $\Delta u = f \quad \Rightarrow 2^{\text{nd}}$ order, linear, inhomogeneous, with constant coefficients (Poisson equation)

4. $\frac{\partial u}{\partial t} - 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \quad \Rightarrow 3^{\text{rd}}$ order, quasi-linear, homogeneous, constant coefficients (Korteweg-deVries equation describing waves on shallow waters)

5. $\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} - \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2 = f \quad \Rightarrow 2^{\text{nd}}$ order, non-linear, inhomogeneous, constant coefficients (Monge-Ampère equation describing surface shapes of given curvature f)

6. $\Delta\Delta u = 0 \quad \Rightarrow 4^{\text{th}}$ order, linear, homogeneous, constant coefficients (plate equation describing mechanical waves in thin plates and membranes)

7. $\frac{\partial u}{\partial t} = \nabla \cdot (D(x)\nabla u) \quad \Rightarrow 2^{\text{nd}}$ order, linear, homogeneous, varying coefficients (inhomogeneous, anisotropic diffusion equation. Notice that even though the physical process that this equation models is called *inhomogeneous diffusion*, the mathematical form of the PDE is *homogeneous*.)

8. $a(x)\frac{\partial u}{\partial t} + b(x)\frac{\partial u}{\partial x} = g \quad \Rightarrow 1^{\text{st}}$ order, linear, inhomogeneous, varying coefficients (geographic population dynamics)

11.2.1 Sub-classes of linear PDEs of second order

Linear PDEs of second order occur most frequently when modeling real-world systems in physics, engineering, or biology. For this class of PDEs, there exists, therefore, a finer classification into sub-classes. These sub-classes are very useful when numerically simulating a second-order linear PDE in the computer because PDEs of different sub-classes require different simulation algorithms. The classificaton into sub-classes is solely based on the homogeneous part of the equation. For inhomogeneous PDEs, the inhomogeneous terms are simply removed before determining the sub-class as they do not influence sub-class membership.

General form of the homogeneous part

For every linear, second-order PDE, the homogeneous part can be written as a linear combination of second-order, first-order, and zeroth-order derivatives of the unknown function, thus:

$$\sum_{i,j=1}^{d} a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d} b_i \frac{\partial u}{\partial x_i} + cu = 0$$

with symmetric coefficients $a_{ij}(\underline{x}) = a_{ji}(\underline{x})$, and coefficients $b_i(\underline{x})$ and $c(\underline{x})$. The symmetry of the coefficients a_{ij} implies that all Eigenvalues of the matrix (a_{ij}) are real. If a PDE contains asymmetric coefficients in the second-order derivatives, they can always be symmetrized by splitting the term into two equal halfs, since partial derivatives commute (i.e., their order does not matter), e.g.:

$$3\frac{\partial^2 u}{\partial x_1 \partial x_2} = \frac{3}{2}\frac{\partial^2 u}{\partial x_1 \partial x_2} + \frac{3}{2}\frac{\partial^2 u}{\partial x_2 \partial x_1}\,.$$

Based on the (symmetric) matrix $A = (a_{ij})$ of the second-order coefficients, linear second-order PDEs are then classified into:

Elliptic Equations

In elliptic PDEs all Eigenvalues λ_i of A are strictly positive (>0). This means that the matrix A is positive definite. Equations with a negative definite A can always be made elliptic by replacing A with $-A$.

Hyperbolic Equations

In a hyperbolic PDE, all Eigenvalues λ_i of A are positive, except one. The matrix A thus has exactly one $\lambda_i < 0$ and all $(d-1)$ other $\lambda_i > 0$.

Parabolic equations

Parabolic PDEs are those that contain second-order derivatives in all variables except one, and where all Eigenvalues λ_i of A are >0 (elliptic A). Parabolic equations can be written in the general form:

$$\sum_{i,j=1}^{d-1} a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \frac{\partial u}{\partial x_d} + cu = 0\,.$$

Remarks

- For PDEs with varying coefficients the sub-class may depend on the point in space. They can, e.g., be elliptic in some regions of space, but become hyperbolic in others.
- The sub-classification of a PDE is invariant to coordinate transformations.
- Certain numerical simulation methods are limited to specific sub-classes of PDEs. This is for example the reason why simulating the wave equation is fundamentally different from simulating diffusion, reactions, or flows.
- In order to solve an elliptic PDE, we need to specify boundary conditions ("boundary value problem").
- Solving parabolic PDEs requires both boundary conditions and initial conditions for the value of u ("initial boundary value problem").
- Hyperbolic PDEs need boundary conditions and initial conditions for both the value of u and for its derivative u' (see, e.g., the wave equation).

Example 11.2.4 (We give examples of the sub-classification of linear second-order PDEs, along with the corresponding system matrix)**.**

1. $\left.\begin{array}{rcl} \Delta u &=& 0 \\ \Delta u &=& f \end{array}\right\}$ $\rightarrow$ elliptic (Laplace/Poisson eq.) $\quad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ $\Rightarrow A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ $\Rightarrow \lambda_i = (1,1) > 0.$

2. $\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u$ $\quad \rightarrow$ hyperbolic (wave equation) $\quad A = \begin{pmatrix} c^2 & 0 & 0 \\ 0 & c^2 & 0 \\ 0 & 0 & -1 \end{pmatrix}$

3. $\frac{\partial u}{\partial t} = D\Delta u$ $\quad \rightarrow$ parabolic (diffusion equation); the left-hand side is the only non-second-order derivative and the right-hand side is elliptic with matrix
$$A = \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix}$$
and Eigenvalues $\lambda_i = (D, D) > 0$ for $D > 0$. For negative diffusion constants D, the equation has unstable solutions and therefore is ill-posed, because it violates thermodynamics (the entropy then spontaneously decreases over time!).

11.3 The Superposition Principle

The superposition principle is valid for all linear, homogeneous PDEs and is a very powerful tool in modeling and simulation. It allows one, for example, to decompose a solution into elementary solutions or to combine found solutions to new ones. The superposition principle is based on the following theorem:

Theorem 4. Are $u_0, u_1, \ldots$ solutions of the same linear homogeneous PDE (for the same homogeneous boundary and initial conditions), then any linear combination

$$u = \sum_i \alpha_i u_i; \quad \text{for any } \alpha_i \in \mathbb{R}$$

is also a solution.

The superposition principle is of central importance in practical modeling and simulation. It is due to this theorem that we could simply add the incident and the reflected waves when discussing boundary conditions for the wave equation. Since both the incident wave and the reflected wave are solutions of the wave equation, the superposition principle guarantees that also their sum is a valid solution. The same is also the reason why the method of images for boundary conditions in simulations works. Another application of the superposition principle we have seen when discussing smooth particle function approximations where the particles carry mollification kernels. The superposition principle guarantees that if the kernels are valid basis functions (solutions to the governing equation, such as Gaussians for the diffusion equation), then also their weighted sum over all particles is a valid solution. The coefficients α_i then correspond to the particle strengths ω_p.
The superposition principle is also used to analytically solve inomogeneous PDEs. There, a frequently used method is to first find a set of solutions for the homogeneous part (the so-called "basic solutions") and then try to find a linear combination of them such that also the inhomogeneous part of the equation is fulfilled (the "particular solution"). For 2^{nd}-order inhomogeneous PDEs, the basic solutions u_i frequently are sines and cosines. The particular solution can thus efficiently be found using the Fourier transform.

11.4 Solution by Fourier Transform

As already seen in the previous chapter, the Fourier transform is a powerful tool to solve PDEs, both analytically and numerically. This is based on the fact that in frequency space the order of all mathematical operations is reduced by one. We have already used this technique when describing fast Poisson solvers on the mesh in hybrid particle-mesh simulations in section 9.3 and to solve the wave equation analytically in section 10.2. We will now formalize this technique and also give a list of useful and frequently used compute rules for the Fourier transform. The general solution strategy for both numerical and analytical solutions is to:

1. transform the PDE to frequency space,
2. solve the resulting ODE (if possible), and
3. transform the solution back to real space.

In order to compute the forward and backward Fourier transforms analytically, the following compute rules might be useful (for an example of use see section 10.2). Capital letters denote the Fourier transforms (i.e., spectra) of the corresponding lower-case functions and $\xrightarrow{\mathcal{F}}$ denotes the Fourier transform. The variable x is the real-space location and k the frequency in Fourier space. We then have:

$$\begin{aligned}
f(a(x+b)) &\xrightarrow{\mathcal{F}} \frac{1}{a} e^{ikb} F\left(\frac{k}{a}\right) \\
e^{ixb} f(x) &\xrightarrow{\mathcal{F}} F(k-b) \\
f(x) * g(x) &\xrightarrow{\mathcal{F}} F(k)G(k) \\
\partial f / \partial x &\xrightarrow{\mathcal{F}} (ik)F(k) \\
\int_{-\infty}^{x} f(\tilde{x}) d\tilde{x} &\xrightarrow{\mathcal{F}} \frac{1}{ik} F(k) \\
1 &\xrightarrow{\mathcal{F}} \delta(k) \\
af(x) + bg(x) &\xrightarrow{\mathcal{F}} aF(k) + bG(k) \quad \Rightarrow \text{linearity of } \xrightarrow{\mathcal{F}}
\end{aligned}$$

These compute rules can help determine the analytical Fourier transform (or its inverse) of a PDE.

11.5 Solution by Separation of Variables

Another powerful technique to solve PDEs analytically is by separation of variables. This, however, only works for homogeneous, linear PDEs (can also be used to find the basic solutions when solving an inhomogeneous PDE) for which the variables are multiplicatively separable. While the first prerequisite is easy to check, the separability of variables is not always obvious and in practice we simply try and see whether it works. A PDE in the two variables x and t is separable if (and only if) its solution $u(x,t)$ can be written as $X(x)T(t)$. This can be used to analytically solve the PDE as follows:

1. *assume* that the solution u is multiplicatively separable,
2. substitute the product form into the PDE,
3. (try to) separate the variables by algebraic manipulations,
4. solve the corresponding ODEs analytically and multiply their solutions.

Similar to the method of Fourier transforms, separation of variables also tries to reduce the PDE to ODEs that can then perhaps be solved. While the method of Fourier transforms works for both analytical and numerical solution, separation of variables is a purely analytical technique.

Example 11.5.1 (Diffusion equation in 1D)**.** We illustrate the use of the method of separation of variables to analytically solve the diffusion equation in 1D. The governing equation is:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}\,.$$

We assume that the solution $u(x,t)$ can be written as the product $X(x)T(t)$ of two unknown function X and T. Substituting this assumption into the PDE yields:

$$u(x,t) = X(x)T(t) \qquad \Rightarrow X(x)\dot{T}(t) = DX''(x)T(t)\,,$$

where the superset dot denotes the (full) dervative with respect to t and a prime a full derivative with respect to x. Dividing the entire equation by $X(x)T(t)$ separates the variables as:

$$\underbrace{\frac{\dot{T}}{T}}_{\text{only depends on t}} = \underbrace{D\frac{X''}{X}}_{\text{only depends on } x}\,.$$

The only way the two sides can be equal for all x and t is for them to be constant, thus:

$$\frac{1}{D}\frac{\dot{T}}{T} = \frac{X''}{X} = -\lambda^2 = \text{constant}\,.$$

The name of the constant $(-\lambda^2)$ is arbitrarily (but cleverly) chosen. Setting both sides of the equation independently equal to $-\lambda^2$, we find the two ODEs:

$$\begin{cases} \dot{T} = -\lambda^2 TD & (1) \\ X'' + \lambda^2 X = 0 & (2) \end{cases}$$

that are both linear, homogenous, and with constant coefficients. They can thus be solved analytically and we obtain the solutions:

$$\begin{aligned} (1) \quad &\Rightarrow \quad T(t) = Ae^{-\lambda^2 Dt} \\ (2) \quad &\Rightarrow \quad X(x) = B_1\sin(\lambda x) + B_2\cos(\lambda x)\,. \end{aligned}$$

Multiplying the solutions for $X(x)$ and $T(t)$, we find the set of basic solutions for the product $u(x,t) = X(x)T(t)$ as:

$$u(x,t) = \left(C_1\sin(\lambda x) + C_2\cos(\lambda x)\right)e^{-\lambda^2 Dt}\,.$$

In order to determine the values of the two integration constants C_1 and C_2, we would need to prescribe a boundary condition and an initial condition (the diffusion equation is parabolic; see above!).

11.6 Solution by the Method of Characteristics (Optional Material)

Even though hyperbolic PDEs can also be solved using Fourier transforms (see chapter 10.2 for an example), there is a special technique particularly for hyperbolic equations. This technique is called the method of characteristics. "Characteristics" are curves in space-time that correspond to the trajectories that the particles follow if their spatiotemporal dynamics is governed by the hyperbolic PDE in question. For the wave equation, the characteristics are the rays (directions of propagation) of the waves. Re-writing a hyperbolic PDE along its characteristics renders it solvable. Finding the symbolic expressions for the characteristics can be quite involved. For PDEs of only two variables, however, a general recipe exists and the method of characteristics always works. We first give this general recipe and then apply it to the example of the 1D wave equation. The method consists of the following steps:

1. Write the hyperbolic PDE in the general form

$$a(x,y)\frac{\partial^2 u}{\partial x^2} + 2b(x,y)\frac{\partial^2 u}{\partial x \partial y} + c(x,y)\frac{\partial^2 u}{\partial y^2} = e(x,y)\,.$$

2. Formulate the associated quadratic PDE:

$$a(x,y)\left(\frac{\partial z}{\partial x}\right)^2 + 2b(x,y)\frac{\partial z}{\partial x}\frac{\partial x}{\partial y} + c(x,y)\left(\frac{\partial z}{\partial y}\right)^2 = 0\,.$$

3. Determine λ and μ as the two solutions of the quadratic algebraic equation $ax^2 + 2bx + c = 0$ and re-write the associated quadratic PDE as:

$$\left(\frac{\partial z}{\partial x} - \lambda(x,y)\frac{\partial z}{\partial y}\right)\left(\frac{\partial z}{\partial x} - \mu(x,y)\frac{\partial z}{\partial y}\right) = 0\,.$$

4. Since the original PDE is hyperbolic, μ and λ are real and distinct. We thus get two linear PDEs of first order:

$$\begin{cases} \dfrac{\partial z}{\partial x} - \lambda(x,y)\dfrac{\partial z}{\partial y} = 0 \\ \dfrac{\partial z}{\partial x} - \mu(x,y)\dfrac{\partial z}{\partial y} = 0\,. \end{cases}$$

5. Solve these first-order linear PDEs analytically. The two solutions $v(x,y) = \not{c}$ and $w(x,y) = \not{c}$ are the characteristics of the original PDE.

6. Write the original PDE in the new coordinates along the characteristics, thus apply the coordinate transform:

$$\begin{aligned} \xi &= v(x,y) \\ \eta &= w(x,y) \qquad \Rightarrow \text{"canonic form"}. \end{aligned}$$

7. Solve the canonic form using the Riemann integration method.

8. Transform the solution back to original variables.

Example 11.6.1 (1D Wave equation)**.** We perform the individual steps of the above recipe for the 1D wave equation in order to determine its general analytic solution.

1. The general form of the wave equation in 1D is:

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0\,.$$

   This corresponds to the generic form above with constant coefficients $a = 1$, $b = 0$, $c = -c^2$, and variables $x = t$ and $y = x$.

2. The associated quadratic PDE is:

$$\left(\frac{\partial z}{\partial t}\right)^2 - c^2 \left(\frac{\partial z}{\partial x}\right)^2 = 0\,.$$

3. The quadratic algebraic equation in this case is linear and reads $x^2 - c^2 = 0$. The two solutions thus are: $\lambda = -c$, $\mu = c$.

4. Using these solutions, we can re-write the associated quadratic PDE as:

$$\left(\frac{\partial z}{\partial t} + c\frac{\partial z}{\partial x}\right)\left(\frac{\partial z}{\partial t} - c\frac{\partial z}{\partial x}\right) = 0\,.$$

5. The two linear first-order PDEs corresponding to the two factors in parentheses thus are:

$$\begin{cases} \dfrac{\partial z}{\partial t} + c\dfrac{\partial z}{\partial x} = 0 \\ \dfrac{\partial z}{\partial t} - c\dfrac{\partial z}{\partial x} = 0\,. \end{cases}$$

6. Solving these two equations yields the characteristics

$$\begin{aligned} v(x,t) &= x - ct = \not{c} \\ w(x,t) &= x + ct = \not{c}\,. \end{aligned}$$

7. Transforming the original PDE into the new coordinates $\xi = x - ct$ and $\eta = x + ct$ (i.e., writing it along the characteristics), it becomes:

$$4c^2 \tilde{u}_{\xi\eta} = 0\,.$$

8. Solving this canonical form using Riemann integration yields the solution

$$\tilde{u}(\xi,\eta) = f(\xi) + f(\eta)\,.$$

9. Transforming back to original variables by substituting the expressions for the characteristics, we find the final general solution:

$$u(x,t) = f(x - ct) + f(x + ct)\,.$$

This is the same solution that we have already found in section 10.2 using Fourier transforms. In contrast to the Fourier solution, however, there is no spurious integral term (that is zero anyway). This is because the method of characteristics is closer to physical reality (characteristics have the real-world meaning of trajectories). The integral term was, thus, an artifact of the use of Fourier transforms.

Appendix A

Answers

Answer 1. For a spherical droplet, both curvature radii are identical, thus $R_1 = R_2 = R$, and the pressure difference between the inside and the outside of the droplet due to surface tension is $\Delta p = \frac{2\sigma}{R}$.

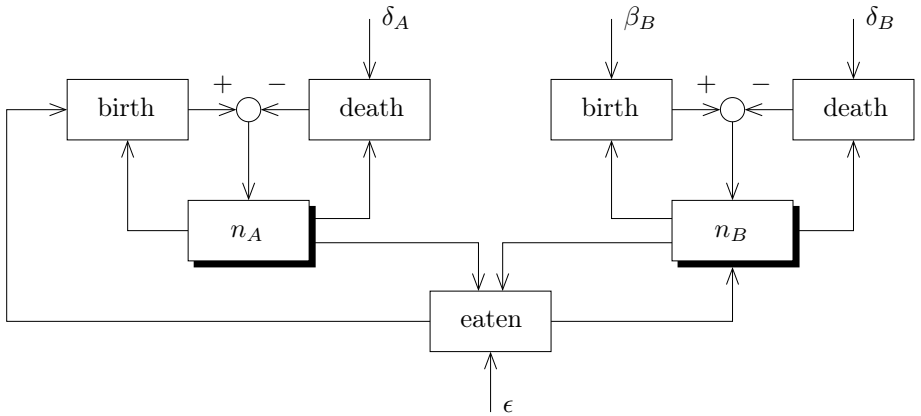
Answer 2. The maximum pressure difference is attained when the membrane is maximally curved in both directions, i.e. $R_1 = R_2 = 20\,\mathrm{nm} = 2 \cdot 10^{-8}\,\mathrm{m}$. With the given value of σ, the pressure difference due to surface tension thus is $3 \cdot 10^6\,\mathrm{N/m^2}$. Since 1 Pascal is equal to 1 $\mathrm{N/m^2}$, this amounts to about 30 bar. Across a cell membrane, we can thus sustain pressure differences of almost 30-fold atmospheric pressure! This is needed in order for the cell to balance the immense osmotic pressure from the large differences in salt concentration between the intra-cellular and extra-cellular spaces. Only thanks to this huge surface tension cells do not burst under the osmotic pressure.

Answer 3. Each term in the Navier-Stokes equation has dimensions $\frac{M}{L^2T^2}$. The Navier-Stokes equation is thus dimensionally homogeneous and fulfills Theorem 1.

Answer 4. The model equations for the temporal dynamics of the ecosystem are:

$$\frac{dn_A}{dt} = \beta_A(\epsilon n_A n_B) n_A - \delta_A n_A$$
$$\frac{dn_B}{dt} = \beta_B n_B - \delta_B n_B - \epsilon n_A n_B\,,$$

where n_A is the number of animals of species A and n_B the number of animals of species B. The birth rate $\beta_A(\cdot)$ is a function of the argument in parentheses in order to model the effect of nutrition on proliferation. The corresponding causality diagram is:



Note that all inputs to the causality diagram need to be causally independent. β_A is thus not an input as it depends on $\epsilon n_A n_B$!

The above solution is not the only one possible. Other valid solutions would, for example:

- formulate the equations in terms of population densities instead of absolute numbers, or
- explicitly represent the multiplication $n_A n_B$ as a node in the causality diagram.